Evaluating humanness in language models

Dissertation
zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

vorgelegt von

Clayton Greenberg

Saarbrücken, 2023

ISBN: 9798884356313
Imprint: Independently published

Abstract

Advances with language models, systems that predict upcoming words in context, have enabled an era in which people sometimes cannot distinguish between human-written and artificially created text. Perplexity, the simplest and most popular way to evaluate the quality of a language model, rewards any pattern captured by the system as long as it robustly constrains the upcoming possibilities. By capturing patterns that humans do not use, optimizing a language model for minimal perplexity could trigger a divergence between the most probable text and the most human-like text.

In this thesis, I argue that this divergence has happened for state-of-the-art language models. Part I characterizes the kinds of knowledge captured by language models. First, I present three novel language model architectures whose neural connections were inspired by human behavior. Then, I discuss novel morphology- and sentiment-based paradigms that capture human knowledge quantitatively. Part II establishes several methods for evaluating language models by comparison against human behavior measures. I consider the suitability and potential confounds for offline ratings and two paradigms of online reading times: eye-tracking and G-Maze. Then, I use a novel dataset of G-Maze response times to show computational and linguistic evidence of the divergence.

Kurzzusammenfassung

Fortschritte bei Sprachmodellen (LMs) - Systeme, die aus dem Kontext heraus nachfolgende Worte vorhersagen - haben dazu geführt, dass Menschen manchmal nicht mehr zwischen von Menschen geschriebenem und künstlich erzeugtem Text unterscheiden können. Perplexität (PPL), die einfachste und beliebteste Methode zur Bewertung der Qualität eines LM, belohnt jedes vom System erfasste Muster, solange es die kommenden Möglichkeiten stark einschränkt. Durch die Erfassung von Mustern, die Menschen nicht verwenden, könnte die Optimierung eines LM hinsichtlich minimaler PPL zu einer Divergenz zwischen dem wahrscheinlichsten Text und dem menschenähnlichsten Text führen.

In dieser Arbeit wird argumentiert, dass diese Divergenz bei modernen LMs aufgetreten ist. Teil I charakterisiert die Arten von Wissen, die von LMs erfasst werden. Zuerst werden drei neue LM-Architekturen beschreiben, deren neuronale Verbindungen von menschlichem Verhalten inspiriert wurden. Danach werden neuartige morphologie- und sentiment-basierte Paradigmen diskutiert, die menschliches Verhalten quantitativ erfassen. In Teil II werden mehrere Methoden entwickelt, die LMs durch Vergleich mit menschlichen Verhaltensmaßen bewerten. Diskutiert werden die Eignung und mögliche Störfaktoren für Offline-Bewertungen und zwei Paradigmen von Online-Lesezeiten: Eye-Tracking und G-Maze. Ein neuartiger Datensatz der G-Maze-Antwortzeiten wird dazu verwendet, um rechnerische und sprachliche Beweise für die Divergenz zu liefern.

Acknowledgment

To my advisor, Dietrich Klakow: I wrote five lines of MATLAB code to get your attention. Plenty of unexpected things happened after this, notably: deriving least squares regression spontaneously during your interview, approximating "success" as having enough equations and numbers to fill a 30 minute meeting, and seeing that you booked my office hours when we had not touched base in a while. But all those years ago when I wrote that code, I did it because I felt that I had found a professor with the perfect mixture of perspective, persistence, and patience to see me through a doctorate. I was right. Every day, I feel lucky that you were my advisor. Giving me the support and time to do this on my terms was the greatest gift I could have asked for. Thank you, immensely, for everything.

To my mentor, Vera Demberg: It still amazes me how quickly, and seemingly effortlessly, you can take some scientific idea, agenda, or issue to a deeper level. Thank you for recognizing, within a few minutes, my Pi Day reference in my qualifying exam email announcement. Thank you for the four other brief moments that resulted in four chapters in this thesis. Thank you for the ordinary and fun moments, too.

Thank you to Chris Callison-Burch for offering local support as I completed this work.

Thank you to my brilliant collaborators: Matthew Crocker, Heiner Drenhaus, Youssef Oualil, Asad Sayeed, Anna Schmidt, Xiaoyu Shen, Les Sikos, Mittul Singh, and Michael Wiegand. It was a pleasure getting to know you, working with you, and living life with you.

Thanks to our student assistant Adam Kusmirek who trained several language models used in Chapter 11, maintained the graphical user interface for our language modeling software, and extended the software itself to support neural language models.

"This work was funded by the German Research Foundation (DFG) as part of the Collaborative Research Center on Information Density and Linguistic Encoding (IDeaL, SFB 1102)." But beyond the mandated affiliation line, thank you to Elke Teich, speaker of the SFB and my SFB mentor; thank you to the Language Model Users Group for countless insights and inspirations, thank you to Alessandra Zarcone, for wisdom and lasagne; thank you to my fellow inaugural PhD representatives David Howcroft and Ekaterina Kravtchenko; thank you to our sensational coordinators, Marie-Ann Kühne and Patricia Borrull Enguix. I am so proud to have been part of this astute, passionate, and loyal community of scholars.

I acknowledge the Cluster of Excellence on "Multimodal Computing and Interaction" (MMCI) for printing several posters about the work contained in this thesis.

Thank you, Robert Fieldhouse, for exploring `transformers` with me.

Thank you, Andre Schmeißer and Steffen Türk, for helping with translating my abstract.

Thank you to my friends for enriching my life, especially Yoav Binoun, for letting me give a best man speech in English at his wedding…to be comprehended by 10% of the guests.

I dedicate this work to my family: Linda Greenberg, Fred Greenberg, and Aaron Garrett. You are my everything.

Co-authoring

In Chapter 1, Section 1.1.1 came from Singh, Greenberg, and Klakow (2016), and a little of Section 1.1.2 came from Oualil et al. (2016a,b). All other parts are unpublished. I claim that all scientific writing and intellectual contributions of this chapter are mine and do not reflect the opinions of my co-authors.

Chapter 2 is based on Singh, Greenberg, and Klakow (2016). I was not involved in implementing this model. However, I wrote the majority of the paper based on an unpublished report. As such, most of the words are mine. My individual contributions to this project were scientific writing, mapping the mathematics of the model to my own theory that words should each have their own decay profile, and linguistic analysis. I claim that, in all, this amounts to 50% of the paper.

Chapter 3 is based on Oualil et al. (2016a). I did not implement the model. Youssef Oualil wrote the majority of the paper. My individual contributions to this project include development of the architecture i.e. deciding which connections to include in the network and development of my own theory that the components of a context vector should be partitioned according to the distance of the history, with smaller regions for more distant history. Additionally, my individual contributions include developing programmatic pipelines for evaluation of word similarity / relatedness, and editing of scientific writing. I claim that, in all, this amounts to 50% of the paper.

Chapter 4 is based on Oualil et al. (2016b). I did not implement the model. Youssef Oualil wrote the majority of the paper. My individual contributions to this project were development of a methodology that quantifies the extent to which a document representation has shifted sentence-by-sentence, interpretation of the results from this methodology, recognizing and explaining the interactions among components of an LSRC node, and linguistic analysis. I claim that, in all, this amounts to 25% of the paper.

Chapter 5 contains Singh et al. (2016). I had a secondary role in implementing the model. Mittul Singh wrote the majority of the publication. My individual contributions were linguistic analysis, especially the challenges of adapting the pipeline to languages from different families, interpretation of results, and editing of scientific writing. I claim that, in all, this amounts to 50% of the original publication. In Section 5.5, I report a follow-up study that was solely my work and not part of the publication. I 100% wrote and submitted a workshop paper based on this follow-up study. The workshop accepted the paper, but then the workshop was cancelled without publishing its proceedings.

Chapter 6 is based on Wiegand et al. (2018). My contributions were confined to discussions of the pipeline and recommendations for fine-tuning word embeddings for the task. I claim that, in all, this amounts to 25% of the paper, especially as there were no other student authors on this paper.

I wrote all text before the first section of each chapter to situate the chapter within this thesis. These texts do not appear in the publications.

Co-authoring for Part II

Chapter 7 is based on Shen et al. (2017). I did not collect or process the data. While Xiaoyu Shen wrote a majority of the publication, I substantially rewrote it for this thesis chapter. My individual contributions to this project were experimental design, interpretation of results, and scientific writing. I claim that, in all, this amounts to 50% of the original publication.

Chapter 8 has not yet been published elsewhere. It was solely my work.

Chapter 9 has not yet been published elsewhere. It was solely my work.

Chapter 10 contains some of Sikos et al. (2017). Specifically, in Section 10.1, “my coauthors” represents Les Sikos and Heiner Drenhaus, supervised by Matthew Crocker. Given that we published in a psycholinguistics venue, we agreed that Les Sikos should be first author because he ran the human experiment. I also did not develop the stimuli. Section 10.2.1 describes parts of Sikos et al. (2017) that were solely my work, which I claim was 50% of the original paper, especially as there were no other student authors on this paper. The other sections of Chapter 10 are solely my work and unpublished.

Chapter 11 contains a small amount of Sikos et al. (2017), as well. I manually reordered, processed, and analyzed the data following the publication. I also re-produced all figures on my own. All language model analyses were solely my work.

Chapter 12 has not yet been published elsewhere. It was solely my work.

List of Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
BERTd	German BERT model released by DBMDZ
BPE	Byte-Pair Encoding
BPTT	Back-Propagation Through Time
CCNN-LSTM	Character-based Long-Short Term Memory
CDLM	Custom Decay Language Model
ENF	Embedding Not Found
FFD	First Fixation Duration
FNN	Feedforward Neural Network
FOFE	Fixed-size Ordinally Forgetting Encoding
FPRT	First Pass Reading Time
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
KN	Kneser-Ney
LBL	Log-Bilinear
LDA	Latent Dirichlet Allocation
LM	Language Model
LSA	Latent Semantic Analysis
LSRC	Long-Short Range Context
LSTM	Long-Short Term Memory, LSTMd for dropout version
LTCB	Large Text Compression Benchmark
ME	Maximum Entropy
mlm	mixed linear model
nonpost	Non-predictive context, post-nominal modification
nonpre	Non-predictive context, pre-nominal modification
NoP	Number of Parameters
OOV	Out Of Vocabulary
POS	Part Of Speech
PPL	Perplexity
predpost	Predictive context, post-nominal modification
predpre	Predictive context, pre-nominal modification
PTB	Penn Treebank
RNN	Recurrent Neural Network
slm	simple linear model, no mixed effects
SRNN	Sequential Recurrent Neural Network
SVM	Support Vector Machines
SWordSS	Sub-Word Similarity-based Search
TFT	Total Fixation Time
unk	the unknown token
WD	word-dependent
WI	word-independent

Contents

List of Figures

List of Tables

Part I

Characterizing knowledge in language models

Chapter 1

Introduction: language models assign numbers to words

On November 30, 2022, OpenAI released a chatbot called ChatGPT, and the media response in the months that followed *seemed* to harken a revolution in artificial intelligence. Teachers worried they would not be able to assign essays for homework anymore because ChatGPT could write them. Workers whose work involves substantial writing reached a new level of worry about the security of their jobs. Geoffrey Hinton, the "Godfather of AI", said in a televised interview[1], "I think in five years' time [ChatGPT] may well be able to reason better than us."

But a contingent of the academic community, myself included, reacted not with shock, but confusion. The technology behind ChatGPT was not new. In some form, it has been driving automatic speech recognition (Katz, 1987), machine translation (Brown et al., 1990), dialogue systems, autocomplete, information retrieval, question answering, and other language technologies for decades. The first chatbot, ELIZA, was developed in the 1960s. And the specific model architecture underlying ChatGPT, the Generative Pre-trained Transformer (GPT) made its debut four years prior (Radford et al., 2018). In short, the world took notice because an online *tool* made a reasonably state-of-the-art language model intuitively accessible, not because the language model itself was singularly revolutionary.

Despite their current fame and power, a language model to me is still a program that takes some text as input and outputs numbers, maybe a number for each word but at least one number for the whole text. These numbers represent the probability of the current word, or sentence, or paragraph, etc., given its context. The context is usually some representation of the words that came before. Perhaps cyclically, the numbers that come out of a language model are both its product *and* its evaluation. If the language model receives a suitable amount of real text, that text "occurred", so it should have as high a probability as possible.

[1] `https://www.cbsnews.com/news/geoffrey-hinton-ai-dangers-60-minutes-transcript/`

It is useful to imagine probability in this case as a limited resource, so an increase in the probability of one text would require a corresponding decrease in the probability of another text that may or may not occur in the future.

Research in language modeling essentially comes down to how to represent that context. In theory, a language model could use any kind of information to refine its predictions about the next word. It is also easy to imagine that humans would practically be able to use this information, too, but they would not be able to use all the kinds of information that could conceivably be encoded in a language model. For example, if there were some statistical pattern that only arises after ten lifetimes' worth of exposure to a language, a human would not be able to use that in producing or comprehending language. Assuming that such patterns exist and assuming language models would be able to capture and use them appropriately, the prevailing way of evaluating language models would incentivize them to become less human-like[2] as their predictions become more accurate.

I call this the *language model divergence*. In this thesis, I claim that state-of-the-art language models have begun using information that humans cannot and do not use to refine their predictions, and so the language model divergence has occurred. I show some kinds of information that language models are good at learning and some kinds that humans must have access to, but language models did not seem to capture. Then, by operationalizing humanness as the ability to explain measurements of human behavior, I put forward evidence that the language model divergence has occurred and advocate for transitioning towards using measurements of human behavior to evaluate language models in the future.

1.1 A brief history of language modeling, in three parts

I assert that two monumental advances triggered three distinct eras in the history of language modeling. The first, the introduction of neural networks for language modeling, brought language models out of somewhat niche use and into widespread deployment. The second advance, the Transformer language model, in several variants, triggered the availability of language models that could generate somewhat convincingly human-like text.

1.1.1 *n*-grams and long distance dependencies

The first language models represented context in arguably the simplest way possible: enumeration. That is, they counted all of the observed continuations of a context and set the probabilities accordingly. But, the number of contexts to track grows exponentially with the context size, so these models commonly used a Markov assumption: cut off the context so it only contains the $n-1$ most recent words. Thus, the model only considers those plus the current word, n words in all, at one time. This is called an n-gram language model. Of

[2]"Less human-like", in the current sense, means using a larger proportion of information that humans cannot or do not use.

course, contexts of size $n-1$ could still be missing from the training data, which motivated a plenitude of smoothing methods to prevent unexpected probabilities of zero. For example, Kneser-Ney smoothing (Kneser and Ney, 1995), which considers the diversity of contexts for a word, became a standard.

n-grams were difficult to outperform for a very long time (Rosenfeld, 2000), as they are mathematically optimal aside from the influence of distant context. My coauthors and I, in Singh, Greenberg, and Klakow (2016), investigated how long a word in the context maintains influence over the future as more and more other words intervene. Put another way, as long as a word remains influential, statistical properties of the future, such as probabilities of words, are *dependent* upon it. This motivates the present use of the term "long-distance dependency", but note that this is not exactly the same kind of long-distance dependency in language commonly studied in linguistics. For those, context influences the grammaticality or acceptability of a future word rather than its probability. For example, the *subject* of a sentence, regardless of however many words intervene or the grammatical number of intervening nouns, *determines* the grammatical number of the verb.

To quantify information in dependencies of long distances, my coauthors and I used a variant of pointwise mutual information. Specifically, for a given pair of words (w_1, w_2) separated over a distance d, this examines the ratio of the actual co-occurrence rate to the statistically predicted co-occurrence rate: $c_d(w_1, w_2) = \frac{\Pr_d(w_1,w_2)}{\Pr(w_1)\Pr(w_2)}$. A value greater than 1 shows it is more likely that the word w_2 follows w_1 at a distance d than otherwise expected according to the unigram frequencies of the two words. My coauthors generated Figure 1.1, an example variation of this correlation for pronouns with the distance d on the English Gigaword corpus (Graff and Cieri, 2003).

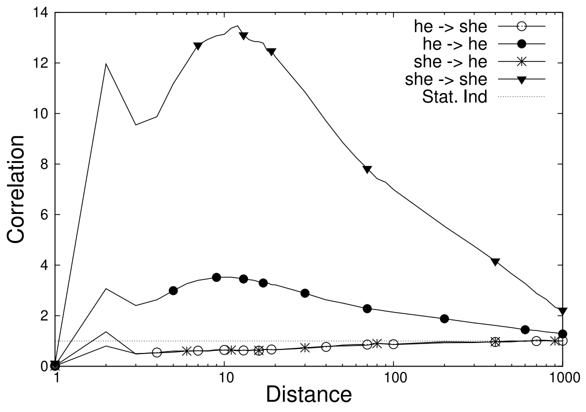


Figure 1.1: Variation of word triggering correlations for pronouns over large distances.

As I wrote in the publication, seeing another "she" about twenty words after seeing a first "she" is more than 13 times more likely than seeing a "she" in general. A similar, but interestingly weaker, observation can be made for the word "he". Note also that "she" somewhat suppresses "he" and vice versa, and these cross-correlations, although negative, are still informative for a prediction system. In summary, Figure 1.1 demonstrates that plenty of word triggering information is spread out over long-distance dependencies that is typically beyond the reach of n-gram language models.

1.1.2 Neural network language models

As neural networks for language modeling are able to capture much more of the triggering information described in Section 1.1.1, they produced substantially better predictions. This was mainly because they encode words as vectors in a continuous space. Vector addition does not increase the dimensionality of the vectors, so representing words as vectors means that any number of them can be combined without increasing the number of parameters in the model. Bengio et al. (2003) proposed a Feedforward Neural Network (FNN) for language modeling, as a variant of n-gram language models. This FNN encoded the words in the entire n-gram (context and current word) as vectors, combined them, and then computed the probability from the combined vector. This approach was very successful and has been shown to outperform a mixture of different other models (Goodman, 2001a), and to improve speech recognition performance significantly (Schwenk and Gauvain, 2005).

To "overcome" this constraint in which only n words are considered at a time, since as shown in Section 1.1.1, triggering can persist beyond 1000 words, Mikolov et al. (2010, 2011c) proposed and implemented a toolkit for a Recurrent Neural Network (RNN) language model. In such a model, there are two inputs at every time step: the current word and the context vector from the previous time step. In theory, this allows information about the context to cycle in the network indefinitely. However, in practice, my coauthors and I found that information rarely persists in such a network for more than eight words (Oualil et al., 2016b).

Sundermeyer, Schlüter, and Ney (2012) originally pitched their improvement to the RNN, called a Long-Short Term Memory (LSTM) network, as a way to address the RNN-specific issues of vanishing and exploding gradients. To train a model that feeds the context vector back into the network as an input, it is common practice to propagate errors backward through those recurrent connections. Essentially, the training process "unrolls" the network. By unrolling this way, there is a substantial chance that the gradient becomes too large to be usable as it would dominate the previous values for the parameters that are being iteratively updated (exploding gradient), or that it becomes 0, meaning that it does not provide a direction to update the parameters and improve the model (vanishing gradient). As the field of language modeling adopted LSTMs, with their separate input, output, and forget gates, this architecture was hailed as especially appropriate because these gates can explicitly control the longevity of context information in the network.

RNNs, and even more so for LSTMs, require orders of magnitude more parameters and training iterations before their predictions are suitably accurate. It is especially problematic that the number of parameters grows super-linearly with the size of the vocabulary (the list of possible words available to occur at each position in the utterance). Also, utilizing classes to speed up training for RNNs leads to some instability in language model quality. Therefore, my coauthors and I in Singh, Greenberg, and Klakow (2016) proposed, implemented, and evaluated the Custom Decay Language Model (CDLM), a non-recurrent architecture designed to have the minimal power (number of parameters) necessary to capture triggering relationships of the kind discussed in Section 1.1.1.

To combine the benefits of FFNs and RNNs, my coauthors and I in Oualil et al. (2016a) proposed, implemented, and evaluated the Sequential Recurrent Neural Network (SRNN) language model. Specifically, this architecture took $n + 1$ inputs at each time step: the current word represented as a continuous vector, the preceding $n - 1$ words represented *individually* as continuous vectors, and a single recurrent vector meant to capture the context before those $n - 1$ words. The network contained lateral connections that allowed the word representations to specialize in a particularly linguistically-motivated way.

As a different approach that took advantage of the still somewhat recent advent of the LSTM, my coauthors and I in Oualil et al. (2016b) modified the internal structure of an LSTM node to include one additional recurrence, making the Long-Short Range Context (LSRC) Language Model. This allowed two, rather than one, vector representations of the context: the short context vector and the long context vector. Although this model had the linguistically desirable quality that its long context vector changes over time much more gradually, allowing some information to persist throughout an utterance of arbitrary length, the prediction accuracy of LSRC ended up being roughly comparable to an LSTM with dropout. Dropout is an incredibly successful strategy for machine learning in which nodes are temporarily removed from the network according to a probabilistic function, encouraging redundancy among the different nodes. There is a plausible analogy between dropout and the refractory period after a (bio) neuron has fired.

1.1.3 Transformers

Somewhat related to the SRNN, the Transformer (Vaswani et al., 2017) includes individual context vectors for the "recent" context, although the notion of "recent" expanded to about 512 words given advances with machine learning on graphics processing units (GPUs). A Transformer has components of two kinds, encoder and decoder, although a single system might contain multiple encoders or decoders. The encoder creates these individual context vectors for the various positions in the 512 most recent words. Then, it combines them using weighted addition. There are a number of ways to set the weights, but the most complicated, successful, and commonly used is an LSTM mechanism called self-attention. The values in the context vectors determine the weights. As such, the encoder considers the words themselves when "deciding" what parts of the context to "attend" to, and weights the context words accordingly.

A decoder component begins with the already combined context vector and, using self-attention over the components of that context vector, generates probabilities for the next words. If the decoder is being used to generate language, the selected word at each time step becomes an input for the next. This is called auto-regression. Note then that the input begins as a sequence, leaves the encoder as a single representation, and then leaves the decoder as a sequence. As such, a Transformer is a sequence-to-sequence model.

Next, the Bidirectional Encoder Representations from Transformers (BERT) language model (Devlin et al., 2018) commonly contains several encoder components (layers). The unique idea for this architecture was to input an entire text, perhaps a few sentences, into the model at one time, and then use the words both before and after the current word to refine the model's predictions. More specifically, the self-attention part can pass back and forth (bidirectional) over the text to set the weights for the combined context vector. So, such a model is only trained to see complete texts and predict a temporarily masked word at each position.

As a final architecture, the Generative Pre-trained Transformer (GPT) language model (Radford et al., 2019) commonly contains several decoder components. While GPT has possibly bi-directional self-attention over the final representation of context, it contrasts with BERT in that the information remains strictly sequential (auto-regressive). Therefore, this model does something much simpler to create a context vector, similar to an RNN, but it does more computation than BERT on that context vector to generate predictions. BERT and GPT form a contrasting pair of Transformers to evaluate. While BERT has access to the future and lacks auto-regression, GPT lacks access to the future and has auto-regression.

In summary, language modeling research in the recent years yielded several appealing neural network architectures, many of which requiring outrageous amounts of data, processing power, and time to train and use from scratch. Further, a language technology application, especially in industry, is not exactly incentivized to go through the trouble of changing its language model architecture unless that change could have a substantial benefit to the users. Therefore, it really matters how academia evaluates language models because engineers use those evaluations to make decisions. Ideally, an appropriate authority could set up a test, large and comprehensive enough to enable statistical inference and specific to the language technology application, which holds all other variables constant and evaluates the performance of multiple architectures on the downstream application itself. For a litany of reasons, including finding the proper venue for such results, having resources to conduct the test, and considering that the other variables do not stay constant, this is not feasible.

So, the field persists with intrinsic evaluation. Language models associate a conditional probability with each word and these probabilities are, by definition, real numbers between 0 and 1. Multiplying many numbers in that interval could have underflow issues. So this, as well as naturalness of the term in psycholinguistics literature, motivates a switch from probability to *surprisal* (Hale, 2001), the negative logarithm of the probability value. With surprisals, the minimum value is 0, corresponding to a probability of 1 (certain, or completely not surprising), and there is no upper bound as the probability approaches 0 (nearly impossible, or very surprising). Then, all of the individual surprisals can be summed to get a joint surprisal. The arithmetic mean of the surprisals is called the *average surprisal* or *cross entropy*. And, the exponentiation of the cross entropy, which undoes the logarithm that transformed probability into surprisal, is called *perplexity*. Since the minimum surprisal is 0, the minimum average surprisal is 0, and the corresponding minimum perplexity is 1. Just as with surprisal, perplexity has no upper bound. If multiple language models each compute the average surprisal for a large test corpus, these average surprisals, and therefore, perplexities, correspond inversely with how accurate the predictions were for each language model. That is, if the language model predicted a word well, it assigned a low surprisal to it. This leads to the intuitive interpretation of perplexity: the average number of equally probable words among which the language model is choosing at each position in the utterance.

This, combined with the fact that perplexity correlates incredibly precisely with word error rate for automatic speech recognition (Klakow and Peters, 2002), means that most language model evaluation stops at perplexity. But, as discussed before, perplexity evaluation only rewards the most probable text. After a language model divergence, the most probable text is not the best text. So, what can be used to evaluate language models in the place of perplexity?

1.2 Measures of human behavior

My position is that perplexity evaluation should be succeeded by correlation with some human behavior measurement. The first, perhaps obvious, choice is to ask humans for introspective judgements / ratings of humanness on a Likert (1932) scale. But this has two potentially severe shortcomings. First, there needs to be something to rate, ideally something that does not require specialized knowledge to evaluate. Sampling from a language model meets this ideal, but perhaps does not capture the essence of a language model's predictions.

The second potential shortcoming is that introspection is highly unnatural. Humans do not often interact consciously with their knowledge of language. So, a more faithful application of language knowledge is having humans complete a task that requires language knowledge, measuring some associated behavior, and then evaluating humanness indirectly based on the relationship between the language model predictions and the human behavior. Since language models make predictions about upcoming words, experimenters can ask humans to do the same explicitly. When completed by humans, this is called a Cloze task (Taylor, 1953). Specifically, the part that can be measured is which words humans suggest as good continuations of a text segment. Note that context can be very constraining, as in "Before mailing the letter, I affixed a postage ____", or not very constraining, as in "They ____". Since language models cannot "break" when they encounter something unpredictable, they must make less extreme predictions than those suggested by the distributions of Cloze responses. Smith and Levy (2011) confirmed a dissociation between Cloze responses and language model predictions, as well as between Cloze responses and human reading times.

In contrast to introspection and Cloze, perhaps the best way for measurements to capture human knowledge of language is via online comprehension tasks, which require humans to interact with language in real time. The most pronounced behaviors of this sort occur when the linguistic stimuli are aberrant, whether that arises from phonology / morphology, syntax, semantics, pragmatics, or some combination thereof. Brainwaves (electroencephalography) are especially promising as a way to measure human responses to aberrant stimuli. In particular, the N400 is traditionally associated with semantic violations (Kutas and Hillyard, 1980) and the P600 is traditionally associated with syntactic violations (Osterhout and Holcomb, 1992). But then again, these measures focus on detecting when something goes wrong. They do not say much about the case in which processing proceeds normally.

There are a number of measures of “normal” language processing during reading. Witzel, Witzel, and Forster (2012) compared three main ways to measure reading times. First, eye-tracking measurements involve where, when, and for how long the eyes fixate on text. This gives a window to what words the human is currently processing and how difficult it is to recognize and integrate the new information with the existing information. Second, in self-paced reading (Just, Carpenter, and Woolley, 1982), humans proceed word-by-word through a text by pressing a button. Third, in a Maze (Forster, Guerrera, and Elliot, 2009), humans navigate a series of binary forced choices. At each choice in an L-Maze, one word is a valid continuation of the sentence and the other is not a valid word. In a G-Maze, one word is a valid continuation of the sentence and the other is a valid word that cannot continue the sentence grammatically. Finally, the grandly named Index of Cognitive Activity is the “number of times per second that an abrupt discontinuity in the pupil signal is detected.” It is believed to be related to cognitive effort, as opposed to time (Marshall, 2002).

1.3 Structure of the thesis

This thesis on the language model divergence has two parts. First, it was essential to complete some work in novel architectures and look for connections to humanness on that level. So, in Part I, I present three novel architectures that I already introduced in Section 1.1.2. They are the Custom Decay Language Model (Chapter 2), the Sequential Recurrent Neural Network Language Model (Chapter 3), and the Long-Short Recurrent Context Language Model (Chapter 4). Then, I do one analysis of language model “knowledge” at the morphological level (Chapter 5) and one at the sentiment level (Chapter 6).

Then, in Part II, I use various measures of human behavior to evaluate language models. Chapter 7 presents a study on direct introspection into the humanness of language models. Chapter 8 gives the basic framework for how I obtain data from a language model so that this data can be evaluated against human behavior. Once in place, I explore the surface relationship between surprisal, word frequency, and word length, to ensure the results of my humanness studies are not confounded. Then, in Chapter 9, I evaluate language models against English eye-tracking data, English G-Maze data, and German eye-tracking data. Since language models explained more variance in the G-Maze data than in the eye-tracking data, I report a new dataset of G-Maze response times that my colleagues created and perform a computational analysis of it that shows definitive evidence for the language model divergence (Chapter 10). Through a battery of tests for systematic differences designed into the new G-Maze data, I show linguistic evidence for the divergence as well (Chapter 11).

Finally, I conclude in Chapter 12 by conjecturing that the divergence will continue to widen, and I offer an outlook for the future of language modeling.

Chapter 2

Language models can mimic human burstiness

As introduced in Chapter 1 and Singh, Greenberg, and Klakow (2016), there are triggering effects in corpora spanning at least 1000 words. It is perhaps striking that auto-triggering curves, in which a word becomes more likely to occur again after a first occurrence, and cross-triggering curves, in which a word becomes more likely to occur after a different word occurs, vary widely among words and word pairs. The strength of these curves and common linguistic knowledge have led to the rise of the term "bursty" as applied toward human language. The idea is that if a topic appears once, perhaps "language models" as an example for this thesis, the topic appears many more times, i.e. it "bursts".

Upon inspecting the triggering curves, I contributed the idea to make a language model that could minimally learn decay profiles (triggering curves) that are *custom* to each word. This is how I coined the name for the novel language model architecture presented in this chapter: the Custom Decay Language Model (CDLM).

2.1 Introduction and Background

Before CDLMs, scientists proposed that several models, such as the cache-based LM (Kuhn and De Mori, 1990), skip models (Guthrie et al., 2006; Momtazi, Faubel, and Klakow, 2010), and recurrent neural network language models (RNNLMs) (Mikolov et al., 2011c) capture triggering over large contexts. But, from checking a few examples for language generated by these language models, my coauthors and I could see that language models seldom captured cross-triggering. Also, just to capture auto-triggering adequately, they often required a prohibitively large number of parameters as the vocabulary size scales. CDLMs were specifically built to capture long range dependencies while growth in number of parameters remained sub-linear in vocabulary size. They outperformed large-context-size skip models, whose parameters were not constrained this way. Additionally, CDLMs showed a more robust variation of performance metric against the variation of meta-parameters than RNNLMs. My coauthors conducted a study of the sparseness of word representations over different context sizes, and I helped to interpret the results.

Skip models Skip models enumerate dependencies like n-grams, but allow wildcards (skips) at specific positions. This technique in combination with distance-specific smoothing methods spans larger context sizes and reduces the sparseness problem. However, the number of parameters still grow by $O(V^2)$ (where V is the vocabulary size) as the context size grows, making them computationally inefficient. In theory, the skip sizes could be binned or quantized, but this would still require the model architect to set arbitrary boundaries. The CDLM architecture occupied, in its time, a nice position with respect to all of these issues. My coauthors and I gave it the minimal number of parameters that it needed to capture the triggering and it still had representational power more along the lines of a neural network than the skip models that preceded it.

For the experiments, my coauthors built skip models by combining unified-smoothing 3-grams and distance 2-grams, which extended the range. Previously, such a combination had been shown to outperform state-of-the-art smoothed n-gram LMs Singh and Klakow (2013).

RNNLMs RNNLMs provided impressive performance gains when compared to other contemporary language models. Through recurrence, the context size for these models is essentially infinite, or at least, formally unconstrained. This makes them especially suitable for long range dependencies. However, training RNNLMs can be slow, especially because the output must be normalized for each word in the vocabulary. Hierarchical softmax and related procedures that involve decomposing the output layer into classes can help with this normalization Goodman (2001b). Unfortunately, using classes for normalization complicates the training process, since it creates a particularly volatile metaparameter. This can be observed in Figure 2.1, where even for small variation in classes, RNNLMs show unstable variation in perplexity.

I trained many RNNLMs in this time using a popular implementation by Mikolov et al. (2011c). This implementation builds networks that require $H^2 + 2HV + HC$ parameters, where H is the number of hidden units and C is the number of normalization classes. An increase by one in the hidden layer size increases the number of parameters linearly in vocabulary size ($O(V + H + C)$).

2.2 Custom Decay Language Models

To build CDLMs, my coauthor Mittul Singh was inspired by log-linear language models, which have sub-linear growth in the number of parameters with context size (Klakow, 1998).

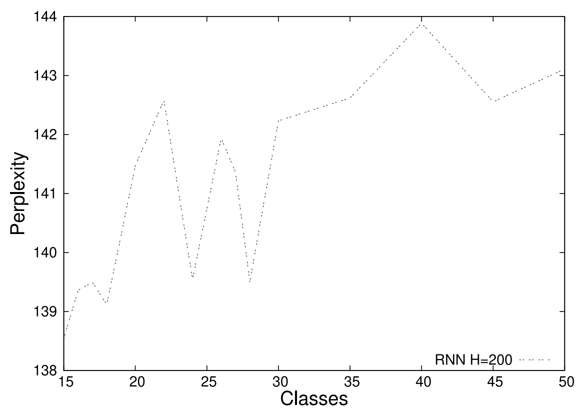


Figure 2.1: Variation of perplexity against the number of classes for a RNNLM with 200 hidden nodes.

The model architecture consisted of two parts: a log-linear model and an n-gram model. For a history of size M, the n-gram part looks at the first $N-1$ ($N < M$) predecessor words and the log-linear part captures the triggering information stored in distances d in the range $[N, M)$. Given the string of words $\{w_{i-M+2}, \cdots, w_{i-1}, w_i, w_{i+1}\}$ where $h = \{w_{i-M+2}, \cdots, w_i\}$, and supposing that $N = 3$, CDLM can be defined as :

$$
\begin{aligned}
P(w_{i+1}|h_i) = \frac{1}{Z(h_i)} &\times P_{3\text{-}gram}(w_{i+1}|w_{i-1}, w_i) \\
&\times \exp(E^{w_{i+1}} v_{w_{i-2}} + \sum_{k=i-N+2}^{i-3} E^{w_{i+1}} T_k v_{w_k}) \qquad (2.1)
\end{aligned}
$$

where i is the position in the document, $P_{3\text{-}gram}$ is a standard 3-gram LM and v_{w_k} is the vector representation of the word at a distance k from the word to be predicted in a C-dimensional, continuous, dense latent space ($C < V$). Here, the dimensions of C can be understood as "classes" capturing latent semantic information in the data.

E^{w_i} refers to a column of the emission matrix E, which weighs the word vectors v_{w_k} to predict the next word. Such a matrix can be thought of as an interpretation function for the current latent state of the model. These latent states exist in the same space as the word vectors. Presumably, some words are closer to this state than others. In this way, the latent states represent semantic concepts that the E matrix can translate into words.

The model also includes a distance specific transition matrix T_k to take word vectors from one distance-based latent space to another. More directly, the T_k matrices control the decay of each word within the latent state. Since the T_k are matrices, as opposed to scalars, which would provide a uniform decay, and as opposed to vectors which would provide a class-based decay, the shape of the decay function is *custom* to each word, which is why this model is named the Custom Decay Language Model.

This setup allows the model to constrain the number of parameters, as each time a word is added to the latent state, only the T_k matrix needs to be updated. Apart from the $O(V^3)$

parameters required to construct the trigram, it needs $O(VC)$ parameters to train the E matrix and the word vectors v_{w_k}, and it needs $O(C^2)$ parameters for training the T_k matrices. In all, CDLM parameters increase sub-linearly with V.

As shown in the last line of Equation 2.1, the model log-linearly combines $T_k v_{w_k}$ at each context position to form a long-distance predictor of the next word. This approach, though inspired by skip models, is more customizable as it allows the exponent parameters to include matrix based formulations and not be constrained only to single values like skip models. Though the exponential element captures the latent / topical information well, the effects are too subtle to capture many simple short-distance dependencies (sparse sequential details). In order to make the model richer in sparse sequential details, my coauthors log-linearly combined the long-distance component with an n-gram LM. In order to estimate the parameters E, v_{w_k} and T_k, my coauthors used the stochastic gradient descent algorithm and minimize the training perplexity of CDLM.

2.3 Language modeling experiments

Corpus My coauthors trained and evaluated the language models on the Penn Treebank as preprocessed by Charniak (2001) using the traditional divisions of the corpus: sections 0-20 for training (925K tokens), sections 21-22 for validation (73K tokens), and sections 23-24 for testing (82K tokens). Despite its final vocabulary of 9,997 words and overall small size, this particular version of the Penn Treebank was a standard for evaluating perplexities of novel language models Mikolov et al. (2010); Cheng et al. (2014). The small size makes training and testing faster, but also makes demonstrating differences in performance more difficult. Presently, this corpus is not used as much for language model evaluation because there is an expected ceiling: the perplexity may not fall much more than it already has.

Experimental Setup My coauthors constructed perplexity experiments as a way to gauge the amount of "knowledge" the CDLMs acquired during training.

In order to establish the most competitive baselines, the RNNLMs trained in the experiment were optimized for number of classes. Recall that these classes just aid the normalization process, as opposed to CDLM classes, which form a very integral part of the model. If classes were overhauled from the RNNLM altogether, training would take much longer, but the perplexity results would be slightly lower. My coauthors found that 15 classes optimized perplexity values for RNNLMs with 50 and 145 hidden nodes, and 18 classes optimized perplexity values for RNNLMs with 500 nodes. These models were trained using the freely available RNNLM toolkit, version 0.4b, with the `-rand-seed 1` and `-bptt 3` arguments.

The n-gram models used were trained with SRILM (Stolcke, 2002). They were a unified smoothing 3-gram (*UniSt*) and an interpolated modified Kneser-Ney 5-gram (*KN*). The *KN* model was trained with the following arguments: `-order 5 -gt2min 1 -gt4min 1 -gt3min 1 -kndiscount -interpolate`.

CDLM uses the unified-smoothing 3-gram as the short-distance dependency component of its model and the long-distance (exponential) element of the model considers up to five words after the trigram.

The learning rate (η) adaptation scheme was managed by the adaptive gradient methods Duchi, Hazan, and Singer (2011). After optimizing on the development set, η was fixed to 0.1 and the dimensionality of the latent space C was fixed at 45.

While building CDLMs, my coauthors first trained a CDLM $M = 4$ and reused its constituent parameters E and v to build CDLM $M = 5$, only updating T_k while training. This process iterated up to $M = 8$.

2.4 Results and Discussion

2.4.1 CDLM robustness analysis

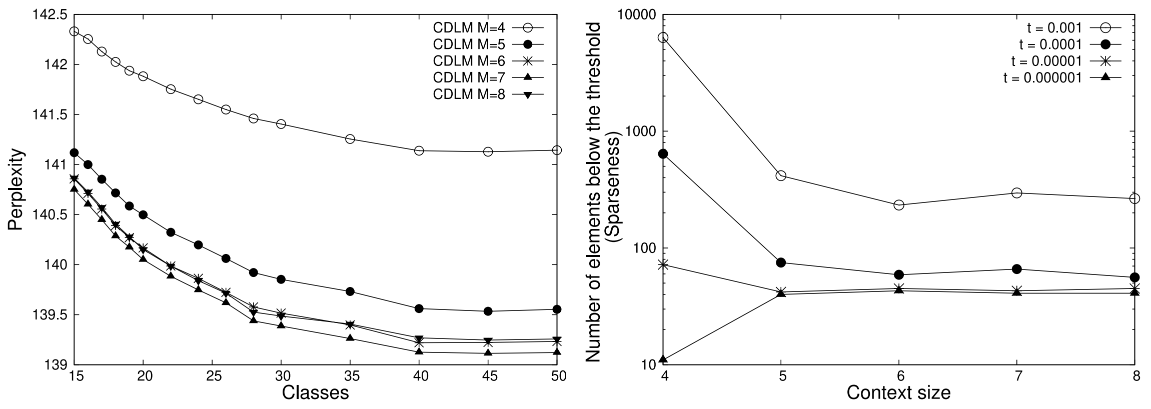


Figure 2.2: (Left) Perplexity versus number of classes (C) in CDLM. (Right) Sparseness of CDLM's transformed word space ($T_l v_{w_l}$) measured at different threshold (t) versus its context size. M represented the long-distance history size.

CDLMs showed robust variation of perplexity with changes in classes, as shown in Figure 2.2. The perplexity values decreased monotonically with increasing classes, as expected since each increase in class created more parameters that can be tuned. Note that moving from $M = 4$ to $M = 5$ doubled the number of T_k matrices, which caused the large perplexity drop.

Along with the robustness shown by CDLMs, the log-linear formulation of CDLMs allowed a follow-up study on the sparseness of the transformed word space matrices represented by $T_k v_{w_k}$ for different distances. My coauthors measured sparseness by counting the matrix entries below a given threshold. By this measure, a more sparse matrix will have large number of entries below the threshold than a less sparse matrix. My coauthors plotted the variation of the sparseness for $T_k v_{w_k}$ matrices for different thresholds against the context size of CDLM in Figure 2.2. In most cases, my coauthors and I observed that as the context size increased, the transition matrices had fewer entries below the threshold, making them less sparse. Therefore, my coauthors and I claimed that this matrix formulation alleviated the sparseness problem and also allowed the exponent part to capture latent information.

LM	Range	Hidden	PPL	NoP
UniSt	3	-	162.1	2.0M
Skip	4	-	160.0	4.1M
	5		155.8	
	6		154.4	
	7		153.6	
	8		153.2	
KN	5	-	141.8	3.2M
RNNLM	∞	50	156.5	1.0M
		145	139.3	2.9M
		500	136.6	10.3M
CDLM	4	45	141.1	2.9M
	5		139.5	
	6		139.2	
	7		139.1	
	8		139.2	
KN+CDLM	$5+4$	45	137.2	6.1M
	$5+5$		135.7	
	$5+6$		135.2	
	$5+7$		134.9	
	$5+8$		134.9	
KN+RNNLM	$5+\infty$	50	120.3	4.2M
CDLM+RNNLM	$7+\infty$	$45+50$	120.2	3.9M

Table 2.1: Test set perplexity (PPL) and total number of parameters (PAR) for each LM.

2.4.2 Perplexity results

Table 2.1 presents a comparison of CDLM with different language models on the basis of their total numbers of parameters and their perplexities. As shown, skip models (*skip*) outperformed the unified smoothing 3-gram (*UniSt3*) as they had more parameters and hence, encoded more information over large distances.

CDLM outperformed *UniSt3* because of spanning larger context size and greater number of parameters at its disposal. *CDLM45* also outperformed the *Skip* models. In fact, increasing the context size of *Skip* to eight words obtained a perplexity of 153.2, which was still less than the CDLM perplexity of 141.1 for a context size of four words. Also, *Skip* required 4.1 million parameters which was more than a third greater than those required to build the seven-word CDLM. Also, *CDLM* was able to perform better than *KN* with fewer number of parameters. When combining *CDLM* with *KN*, increasing the context size for CDLM obtained progressively-better performance than *KN*.

The experiments further compared CDLMs with RNNLMs. An RNNLM with 145 hidden nodes had about the same number of parameters as *CDLM* and performed 0.1 perplexity points worse than *CDLM*. Increasing the hidden units for RNNLM to 500 led to the best performing RNNLM. This came at a cost of using a lot of parameters. To produce better performing language models with fewer parameters, my coauthors constructed an RNNLM with 50 hidden units, which when linearly combined with *CDLM* (*CDLM+RNNLM*) outperformed the best *RNNLM* using less than half as many parameters. It even nominally outperformed the combination of *KN* and *RNNLM* using fewer parameters, but this difference was likely not significant. Combinations of the three different LMs did not result in any large improvements, suggesting that there was redundancy in the information spread over these three types of language models.

Finally, my coauthors and I observed that the increase in parameters did not always lead to better performance, for example, while comparing a *Skip* model with *CDLM*. This increase can be attributed to the richer formulation of *CDLM*. Increasing parameters for *CDLM+KN* did not also lead to a better performance against the fewer-parameters *CDLM+RNNLM*. My coauthors and I also observed this for *CDLM+KN* and *KN+RNNLM*. In this case, my coauthors and I suspected that the lower performance was due to CDLM's lack of recursive connections which form an integral part of RNNLMs. But it was noteworthy that CDLMs, which are not recurrent, captured much of the long-distance information that the recurrent language models did, as evidenced by the perplexity experiments.

2.5 Conclusion

In this chapter, I presented the Custom Decay Language Model (CDLM), inspired by skip models' log-linear technique of dividing context into smaller bigrams and then recombining them. In contrast with skip models, CDLMs used a richer formulation by employing a matrix-based exponentiation method to capture long range dependencies. CDLMs used an n-gram model to capture the short range regularities.

My coauthors and I observed perplexity improvements for CDLMs even when compared to skip models with larger range and Kneser-Ney 5-grams. Additionally, the CDLMs that my coauthors tested used fewer parameters than the larger range skip models and Kneser-Ney 5-grams. CDLMs provided a rich formulation for language modeling where the growth of number of parameters was constrained.

Fortunately for language model research, but perhaps unfortunately for CDLMs, language model size is not the concern that it was in 2016. For free, in the cloud, I routinely use open-source language models with billions of parameters. Language models larger than this exist, but are less often free to use. In short, the property that CDLMs minimally captured triggering is no longer the impressive feat that it was. I assert, without computationally verifying, that today's large language models capture just as much or more helpful triggering information. However, it is worth noting that for low-resource language modeling, a smaller language model with fewer connections might be an asset. CDLMs would be a good choice to model triggering in such a case.

Chapter 3

Language models can make specialized meaning representations of words

The core idea behind Sequential Recurrent Neural Networks (SRNNs) (Oualil et al., 2016a) was to combine the benefits of a feedforward neural network (FNN), which captures short-range context very well, with a recurrent neural network (RNN), which captures long-range context very well. This network architecture, no longer minimal like the Custom Decay Language Model, allowed continuous representations for different positions in the history to contextualize each other. In addition to presenting state-of-the-art perplexity results for the time, this work argued that through lateral connections / contextualization, word embeddings would become more specialized toward syntactic, rather than semantic relationships.

3.1 Introduction

Contrary to FNN, recurrent models such as RNN and LSTM predict the next word based only on the current word and the context representation. Therefore, they lose information about word position rather quickly and cannot model short range dependencies as well as FNN and n-grams. For example, English has position-dependent patterns such as "he $*$ he" ("he said he", "he mentioned he",...). The position of "he" is essential for making the right prediction in this case, and the recurrent models are not designed to encode that. Rather, they are better for smooth incremental updates and hence for longer range dependencies.

This chapter proposes a novel approach that models short range dependencies like FNN and long range dependencies like RNN. In particular, the hidden layers combine explicit encoding of the local context and a recurrent architecture, which allows the context information to sequentially evolve in the network at the projection layer. In the first step, the word representation are enhanced using the context information. This step maps the word representations from a universal embedding space into a context-based space. Then, the system performs the next word prediction as it is typically done in FNN. The learning of the network weights uses the Back-Propagation Through Time (BPTT) algorithm similarly to RNN. The main difference here is the additional network error resulting from the additional sequential connections. Perplexity experiments showed that learning of word-dependent sequential connections can substantially improve the performance of the SRNN network.

3.2 Neural network language models

The following two sections report the mathematics behind neural network language models in general and also the specific architecture of the SRNN. While I did not derive the mathematical results or implement the model, I include these sections because working through them, with the help of my coauthors, helped me form a basis of understanding on how complex language models worked. In addition, I participated actively in interpreting the results. Finally, I highlight Section 3.5, which was my dedicated work during this project.
The goal of a language model is to estimate the probability distribution $p(w_1^T)$ of word sequences $w_1^T = w_1, \dots, w_T$. Using the chain rule, this distribution can be expressed as

$$p(w_1^T) = \prod_{t=1}^{T} p(w_t \mid w_1^{t-1}) \tag{3.1}$$

The rest of this section shows how FNN and RNN are used to approximate this probability distribution.

3.2.1 Feedforward neural networks

Similarly to n-gram models, FNN uses the Markov assumption of order $n-1$ to approximate $p(w_1^T)$ according to

$$p(w_1^T) \approx \prod_{t=1}^{T} p(w_t \mid w_{t-n+1}^{t-1}) \tag{3.2}$$

Subsequently, each of the terms involved in this product, i.e. $p(w_t \mid w_{t-n+1}^{t-1})$, is estimated, separately, in a single bottom-up evaluation of the network according to

$$P_{t-i} = X_{t-i} \cdot U, \qquad i = n-1, \dots, 1 \tag{3.3}$$

$$H_t = f\left(\sum_{i=1}^{n-1} P_{t-i} \cdot V_i\right) \tag{3.4}$$

$$O_t = g(H_t \cdot W) \tag{3.5}$$

X_{t-i} is a one-hot encoding of the word w_{t-i}, whereas the rows of U encode the continuous word representations (i.e. embeddings). Thus, P_{t-i} is the continuous representation of the word w_{t-i}. W and $V = [V_1, \ldots, V_{n-1}]$ are the network connection weights, which are learned during training in addition to U. Moreover, $f(\cdot)$ is an activation function, whereas $g(\cdot)$ is the softmax function. Figure 3.1 (left) shows an example of an FNN with a fixed context size $n{-}1 = 3$ with a single hidden layer.

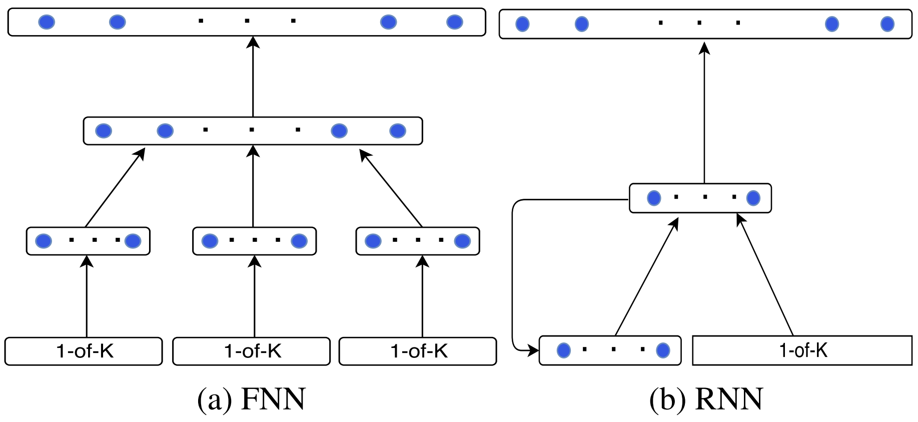


Figure 3.1: FNN versus RNN architecture.

3.2.2 Recurrent neural networks

An RNN attempts to capture the complete history in a context vector h_t, which represents the state of the network and evolves in time. Therefore, it approximates $p(w_1^T)$ according to

$$p(w_1^T) \approx \prod_{t=1}^{T} p(w_t \mid w_{t-1}, h_{t-1}) = \prod_{t=1}^{T} p(w_t \mid h_t) \tag{3.6}$$

RNN evaluates this distribution similarly to FNN. The main difference occurs in Equations 3.3 and 3.4 which are combined into

$$H_t = f(X_{t-1} \cdot U + H_{t-1} \cdot V) \tag{3.7}$$

Figure 3.1 (right) shows an example of a standard RNN. The next section will show how an RNN can be extended to explicitly model short range dependencies through additional sequential connections.

3.3 Sequential Recurrent Neural Network

The main difference between an RNN and an FNN is the context representation. More precisely, The context layer H_t of an FNN is estimated based on a fixed context size i.e. the last $n{-}1$ words, whereas in an RNN, H_t is constantly updated (at each time iteration) using only the last word and context at time $t{-}1$.

3.3.1 The SRNN neural architecture

My coauthors and I developed SRNN as an architecture which captures short range dependencies over the last $n{-}1$ word positions as it is done in FNN, and the long range context through recurrence, similarly to RNN. The design of this structure is motivated by the inefficiency of RNN to model position dependent patterns, which are particularly frequent in conversational speech. I assert that an RNN loses information about word position quickly and therefore cannot efficiently model short range dependencies. FNN and n-gram models, however, are designed as position-dependent models, which deal only with short-term context. Extending RNN structure to represent the short term history explicitly as it is done in FNN will help improve the modeling of short range context, as it will allow the network to capture any residual / additional context information that may be present in the past $i = t{-}n + 1, \ldots, t{-}2$ time iterations but which may have been lost during the last context update, which is based only on the last word at $t{-}1$ (see illustration in Figure 3.2). In the worst case scenario, the context information will be simply redundant and is expected not to harm the performance. The rest of this section introduces the mathematical formulation of this approach.

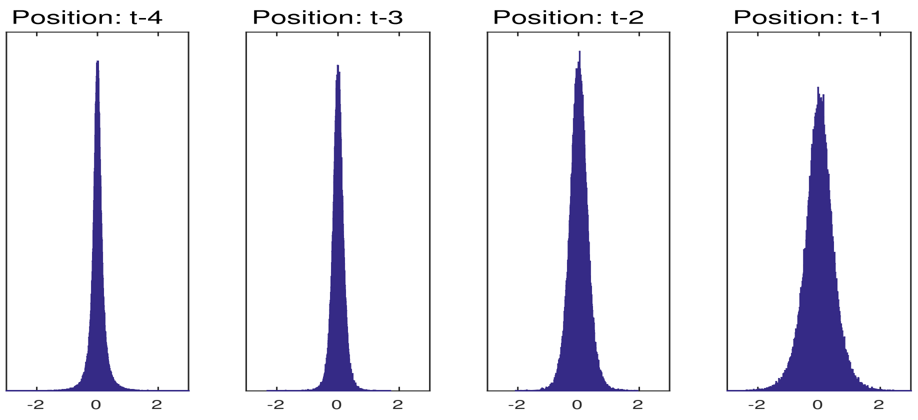


Figure 3.2: Histograms of the projection-to-hidden weights V_1, V_2, V_3, and V_4 (see Figure 3.3) for each of the 4 word positions of an SRNN ($n = 5$) trained on LTCB. These histograms show that the magnitude of the weights decays with the word position (from $t-1$ to $t-4$) but does not nullify. Thus, the model successfully captured some good short-range dependencies.

The Sequential Recurrent Neural Network (SRNN) approximates $p(w_1^T)$ according to

$$p(w_1^T) \approx \prod_{t=1}^{T} p(w_t \mid w_{t-n+1}^{t-1}, h_{t-n+1}) = \prod_{t=1}^{T} p(w_t \mid h_{t-n+2}^{t}) \tag{3.8}$$

Thus, as shown in Equation 3.8, the SRNN architecture explicitly represents the history over the last $n{-}1$ word positions in the manner of an FNN (Equation 3.2), but enhances the actual word representations using the recurrent context information, which propagates sequentially within the network. Furthermore, setting the context to a one word history window ($n = 2$) in Equation 3.8 leads to the RNN approximation in Equation 3.6. Therefore, the SRNN approach can be seen as an extension of the standard RNN to explicitly model and capture short range context.

The additional sequential connections allow the context information to propagate from the past to the future within the network. These connections can be defined as a Word-Independent (WI) recurrence vector, which fixes the amount of context information allowed to propagate in the network, as they can be designed as Word-Dependent (WD) vectors. In this case, each word will have its own context weight vector, which will typically learn which context "neurons" are relevant for that particular word and therefore scales each context unit accordingly.

The network evaluation is performed similarly to FNN, the main difference occurs in Equation 3.3, which becomes in the case of the word-independent model (WI-SRNN)

$$P_{t\text{-}i} = f_s(X_{t\text{-}i} \cdot U + C \odot P_{t\text{-}i\text{-}1}), \qquad i = n{-}1, \ldots, 1 \tag{3.9}$$

as it becomes in the case of the word-dependent model (WD-SRNN)

$$P_{t\text{-}i} = f_s(X_{t\text{-}i} \cdot U + C_{w_{t\text{-}i}} \odot P_{t\text{-}i\text{-}1}), \quad i = n{-}1, \ldots, 1 \tag{3.10}$$

where $f_s(\cdot)$ is an activation function and $\odot$ is the element-wise product operator. C is the word-independent recurrence weight vector, whereas $C_{w_{t\text{-}i}}$ is the word-dependent context weight corresponding to word $w_{t\text{-}i}$.

The SRNN model replaces the universal word embeddings at the projection layer of an FNN by context-dependent word embeddings. More particularly, both Equations 3.9 and 3.10 show that each word representation is enhanced using the context information before proceeding to the next word prediction. Therefore, this particular step could be seen as a transformation from the universal embedding space into a context-dependent space with a better discrimination of words.

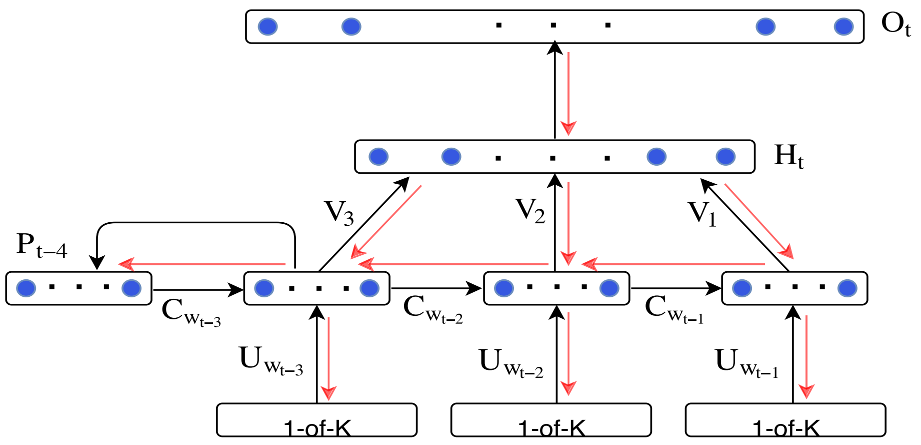


Figure 3.3: Sequential Recurrent Neural Network architecture. The red arrows show the error propagation during training (this figure does not include BPTT).

Figure 3.3 shows an example of an SRNN with three additional sequential connections ($n{-}1 = 3$) and a single hidden layer. As shown, this is a general architecture that includes different networks. In particular, setting $C = [0, \dots, 0]$ and $f_s(x) = x$ results in the classical FNN architecture, whereas setting $n = 2$ leads to a standard RNN with a diagonal recurrence matrix and an additional non-recurrent layer. Moreover, setting C to a fixed value in $[0, 1]$ and $f_s(x) = x$ leads to the Fixed-size Ordinally-Forgetting Encoding (FOFE) (Zhang et al., 2015) architecture, which was proposed to uniquely encode word sequences.

3.3.2 SRNN training

The parameters to train for an SRNN are the word embeddings U, the project-to-hidden connection weights $V = [V_1, \dots, V_{n-1}]$, the hidden-to-output connection weights W and the context weight vector C for the WI model, or $C = [C_1^t, \dots, C_K^t]^t$, where K is the vocabulary size, for the WD model. In this case, each word w in the vocabulary will be characterized by two learnable vectors, namely, the continuous representation (embedding) U_w and the context weight C_w. Similarly to RNN, the parameter learning of an SRNN architecture follows the standard Back-Propagation Through Time (BPTT) algorithm. The main difference occurs at the projection layer, where the additional error vectors resulting from the sequential connections should be taken into account (See example or error propagation in Figure 3.3) before unfolding the network in time.

3.4 Experimental setup

My coauthors evaluated SRNN models on two different benchmark tasks. The first set of experiments was conducted on the Penn Treebank (PTB) corpus using the standard division (e.g. Mikolov et al., 2011c; Zhang et al., 2015): sections 0-20 are used for training while sections 21-22 and 23-24 are used for validation and testing. The vocabulary was limited to the top 10k most-frequent words while the remaining words were all mapped to the token `<unk>`. In order to evaluate how the SRNN approach scales to large corpora, my coauthors ran a set of experiments on the Large Text Compression Benchmark (LTCB) (Mahoney, 2011). This corpus is based on the *enwik9* dataset which contains the first 10^9 bytes of `enwiki20060303-pages-articles.xml`. My coauthors adopted the same training-test-validation data split and preprocessing from Zhang et al. (2015). All but the top 80k most frequent words were replaced by `<unk>`. Table 3.1 gives details about the sizes of these two corpora and the percentage of out-of-vocabulary (OOV) words that were mapped to `<unk>`.

	Train		**Dev**		**Test**	
Corpus	**Tokens**	**OOV**	**Tokens**	**OOV**	**Tokens**	**OOV**
PTB	930K	6.52%	74K	6.47%	82K	7.45%
LTCB	133M	1.43%	7.8M	2.15%	7.9M	2.30%

Table 3.1: Corpus size in number of tokens and OOV rate.

SRNN models are compared to different systems including the n-gram Kneser-Ney (KN) model and different feedforward and recurrent neural architectures. For feedforward networks, the baseline systems include

1. the FNN-based LM (Bengio et al., 2003)
2. Fixed-size Ordinally Forgetting Encoding (FOFE) approach, which was implemented as a feedforward sentence-based model. The FOFE results were obtained using the FOFE toolkit (Zhang et al., 2015). The results are reported for different context sizes ($n - 1 = 1, 2, 4$) and different numbers of hidden layers (1 or 2).

Regarding recurrent models, my coauthors and I compared SRNN models to

3. the full RNN (without classes) (Mikolov et al., 2011c)
4. a deep RNN (Pascanu et al., 2014), which investigates different ways of adding hidden layers to RNN
5. the LSTM architecture (Sundermeyer, Schlüter, and Ney, 2012), which explicitly regulates the amount of information that propagates in the network

3.4.1 PTB experiments

For the PTB experiments, the FNN, FOFE and SRNN architectures have similar configurations. That is, the hidden layer(s) size is 400 with all hidden units using the rectified linear unit i.e. $f(x) = \max(0, x)$, as an activation function, whereas the word representation (embedding) size was set to 200 for FNN, FOFE and LSTM and 100 for SRNN. The latter uses $f_s = \tanh(\cdot)$ as sequential activation function. The hidden layer size of RNN and LSTM were set to 400 and follow the original configuration proposed by Mikolov et al. (2011c) and Sundermeyer, Schlüter, and Ney (2012), respectively. My coauthors also used the same learning setup adopted in Zhang et al. (2015). Namely, the minibatch size for stochastic gradient descent was 200, the learning rate was initialized to 0.4, the momentum was set to 0.9, the weight decay was fixed to $4 \cdot 10^{-5}$ and the training was done in epochs. The weights initialization follows the normalized initialization proposed by Glorot and Bengio (2010). Similarly to Mikolov et al. (2010), the learning rate is halved when no substantial improvement in the log-likelihood of the validation data is observed. Then, training continued with seven more epochs while halving the learning rate after each epoch. The BPTT was set to 5 time steps. For both models, the context connection weights, C, were randomly initialized in $[0, 1]$. In order to compare to the FOFE approach, my coauthors and I also reported results where C was reduced to a scalar forgetting factor that is fixed at 0.7. This is denoted as WI-SRNN$*$ in the tables below. I report the results in terms of perplexity (PPL), Number of model Parameters (NoP), and the training speed, which is defined as the number of words processed per second (w/s) on a GTX TITAN X GPU.

3.4.2 LTCB experiments

The LTCB experiments used the same PTB setup with minor changes. My coauthors continued to use the same experimental setup used in Zhang et al. (2015). More precisely, these results were obtained without usage of momentum or weight decay whereas the mini-batch size was set to 400. The FNN and FOFE architectures contain 2 hidden layers of size 600 (or 400) whereas RNN and SRNN have a single hidden layer of size 600. In order to compare to Zhang et al. (2015), the forgetting factor C of WI-SRNN$*$ is fixed at 0.6.

3.5 Word embedding evaluation

in		**strictly**		**germany**	
U_w	C_w	U_w	C_w	U_w	C_w
into	at	solely	purely	italy	japan
throughout	on	rigidly	totally	france	russia
through	for	broadly	physically	britain	italy
during	their	purely	solely	switzerland	france
within	to	ostensibly	technically	england	spain

Table 3.2: Examples of top 5 similar words.

One of my individual contributions to this project was streamlining the process for obtaining similarity scores, relatedness scores, analogy completions, and nearest neighbors for various sets of word embeddings. Table 3.2 shows some word examples that I found with their top 5 cosine similarities for word embeddings U_w and Euclidean distance for context weights C_w. Spot-checking different embeddings led me to believe that the "universal" SRNN embeddings captured semantic (conceptual) similarities, while the context weights captured more syntactic (functional) similarities. For example, the universal embeddings had high cosines between related words like "car", "wheel", "drive", and "vehicular". The context weights were much more constrained with respect to part of speech, so the nearest words to "car" would have been more like "truck", "van", and "automobile".

3.6 Results

Architecture	**model**			**model+KN5**			**NoP**	**w/s**
$n-1=$	1	2	4	1	2	4	4	4
	1 Hidden Layer							
FNN	176	131	119	132	116	107	6M	24K
FOFE	123	111	112	108	100	101	6M	17K
WI-SRNN*	117	110	109	105	100	99	5M	13K
WI-SRNN	112	107	107	102	98	97	5M	11K
WD-SRNN	109	106	106	99	96	95	6M	10K
	2 Hidden Layers							
FNN	176	129	114	132	114	102	6M	22K
FOFE	116	108	109	104	98	97	6M	17K
WI-SRNN*	114	108	107	102	98	96	5M	11K
WI-SRNN	109	105	104	99	96	94	5M	10K
WD-SRNN	108	103	104	97	94	94	6M	9K
	Recurrent Models							
RNN		123			107		8M	21
Deep RNN		108			–		7M	–
LSTM		114			99		7M	8K

Table 3.3: Language model performance (PPL) on the PTB test set.

Table 3.3 shows the LMs evaluation on the PTB test set. My coauthors and I observed that SRNN models outperformed all other models using the lowest Number of model Parameters (NoP) among all configurations. This also includes other models that were reported in the literature, such as RNN with maximum entropy (Mikolov et al., 2011b), random forest LM (Xu and Jelinek, 2007), structured LM (Filimonov and Harper, 2009) and syntactic neural network LM (Emami and Jelinek, 2004). More particularly, SRNN with two hidden layers achieved a comparable performance to a mixture of RNNs (Mikolov et al., 2011a). My coauthors and I concluded that the explicit modeling of short range dependencies through sequential connections improved the performance. More precisely, the results show that increasing the history window (1, 2 and 4) improves the performance for all SRNN models. Table 3.3 also shows that using a fixed scalar forgetting factor (WI-SRNN∗) leads to a slight improvement over the FOFE approach, which is mainly due to the additional non-linear activation function f_s. Furthermore, the word-dependent (WD-SRNN) model slightly outperforms the word-independent model (WI-SRNN) but with a non-negligible increase in the number of parameters. Regarding the training speed, my coauthors and I concluded that training an SRNN model requires approximately twice the time needed for FFN and RNN, whereas it needs less time compared to LSTM.

Architecture	**model**			**NoP**
$n-1=$	1	2	4	4
KN	239	156	132	–
FNN [M∗200]-600-600	235	150	114	65M
FOFE [M∗200]-400-400	120	115	108	48M
FOFE [M∗200]-600-600	112	107	100	65M
WI-SRNN* [M∗200]-600	110	102	94	65M
WI-SRNN [M∗200]-600	85	80	77	65M
WD-SRNN [M∗200]-600	77	74	72	80M
RNN [600]-600		85		96M

Table 3.4: Language model performance (PPL) on the LTCB test set.

The LTCB results shown in Table 3.4 generally confirm the PTB conclusions. In particular, the SRNN models outperformed all other models while requiring comparable or fewer model parameters. Moreover, the WI-SRNN∗ model with a single hidden layer slightly outperforms FOFE (2 hidden layers). These results, however, show a larger improvement for the WD-SRNN model and for the increased window size (from 1 to 4) compared to the improvement obtained on the PTB. This is mainly due to the large amount of LTCB training data, which allows the model to learn richer WD context vectors.

3.7 Conclusion and outlook

This chapter presented the SRNN architecture, which captures short range dependencies using short history windows and models long range context through recurrent connections. At publication time, this model had state-of-the-art perplexity. Most likely by simultaneous innovation, SRNNs and Transformers share many properties. Specifically, the lateral connections in SRNNs had a similar function to attention in the encoder. The biggest difference was the separate handling of short and long range context in SRNN. Transformers can treat up to 2048 words as "short range" context. Some later approaches such as Transformer-XL (Dai et al., 2019) attempted to preserve the theoretically infinite context afforded by recurrent models like SRNN, but GPT models remain much more popular.

Chapter 4

Language models can be optimized for cohesion

When I presented the long-short range context (LSRC) language model (Oualil et al., 2016b), I mixed vinegar, representing an LSTM node, with baking soda, representing an additional recurrence. The point was that even LSTMs benefit from extra power and cleaner separation of long and short context. LSRC models maintain a fully separate state vector to capture short range context. As such, the long range part was free to adapt more slowly, mirroring how human language preserves cohesion. This third novel architecture marked a transition from expanded power gained from connections *between* nodes to power gained from additional complexity *within* nodes.

4.1 Introduction

In order to overcome the short context constraint and capture long range dependencies known to be present in language, Bellegarda (1998b) proposed to use Latent Semantic Analysis (LSA) to capture the global context, and then combine it with the standard n-gram models, which capture the local context. Similarly, Mikolov and Zweig (2012) showed that recurrent neural network (RNN)-based LM performance can be significantly improved using an additional global topic information obtained using Latent Dirichlet Allocation (LDA). In fact, although recurrent architectures theoretically allow the context to indefinitely cycle in the network, Le, Allauzen, and Yvon (2012) have shown that, in practice, this information changes quickly in the classical RNN (Mikolov et al., 2010) structure, and that it is experimentally equivalent to an 8-gram FNN. Despite its name, the Long-Short Term Memory (LSTM) network (Sundermeyer, Schlüter, and Ney, 2012) does not maintain separate long and short term memory states. Rather, it uses multiple gates to control the flow of information into, within, and out of its single state.

Motivated by the works in Bellegarda (1998b) and Mikolov and Zweig (2012), this chapter discusses one final novel neural architecture which explicitly models 1) the local (short) context information, generally syntactic, as well as 2) the global (long) context, which is semantic in nature, using two separate recurrent hidden states. These states evolve in parallel within an LSRC. In doing so, the LSRC architecture is particularly adapted to model natural languages that manifest local-global context information in their linguistic properties.

4.2 Short vs. long context language models

As with the analogous section in Chapter 3, the point of including this section is that this discussion is how I worked through the connections present in the most intricate, pre-Transformer language models. In particular, my coauthors and I decided to leave the LSTM equations in the publication for the express purpose of providing a comparison between an LSTM node and an LSRC node. For good measure, this section shows that mathematically and diagrammatically. Finally, my individual contribution for the LSRC project was the investigation of context range (Section 4.4).

The goal of a language model is to estimate the probability distribution $p(w_1^T)$ of word sequences $w_1^T = w_1, \dots, w_T$. Using the chain rule, this distribution can be expressed as

$$p(w_1^T) = \prod_{t=1}^{T} p(w_t \mid w_1^{t-1}) \tag{4.1}$$

This probability is generally approximated under different simplifying assumptions, which are typically derived based on different linguistic observations. All these assumptions, however, aim at modeling the optimal context information, be it syntactic and / or semantic, to perform the word prediction. The resulting models can be broadly classified into two main categories: long and short range context models. The rest of this section presents a brief overview of these categories with a particular focus on neural models.

4.2.1 Short range context

This category includes models that approximate $p(w_1^T)$ based on the Markov independence assumption of order $n-1$. That is, the prediction of the current word depends only on the last $n-1$ words in the history. In this case, $p(w_1^T)$ is approximated by

$$p(w_1^T) \approx \prod_{t=1}^{T} p(w_t \mid w_{t-n+1}^{t-1}) \tag{4.2}$$

The most popular methods that subscribe in this category are the n-gram models (Rosenfeld, 2000; Kneser and Ney, 1995) as well as the FNN model (Bengio et al., 2003), which estimates each of the terms involved in this product, i.e. $p(w_t \mid w_{t-n+1}^{t-1})$ in a single bottom-up evaluation of the network.

Although these methods perform well and are easy to learn, the natural languages that they try to encode, however, are not generated under a Markov model due to their dynamic nature and the long range dependencies they manifest. Relaxing this untrue assumption has led to extensive research to develop more suitable modeling techniques.

4.2.2 Long range context

Conventionally, n-gram LMs are not built to capture long range dependencies, although there is substantial word triggering across long distances as shown in Chapter 1. In order to model long-range dependencies and overcome the restrictive Markov assumption, recurrent language models have been proposed to approximate $p(w_1^T)$ according to

$$p(w_1^T) \approx \prod_{t=1}^{T} p(w_t \mid w_{t-1}, h_{t-1}) = \prod_{t=1}^{T} p(w_t \mid h_t) \tag{4.3}$$

In NN-based recurrent models, h_t is a context vector which represents the complete history, and modeled as a hidden state that evolves within the network.

Elman-type RNN-based LM

The classical RNN (Mikolov et al., 2010) estimates each of the product terms in Equation 4.3 according to

$$H_t = f(X_{t-1} + V \cdot H_{t-1}) \tag{4.4}$$
$$P_t = g(W \cdot H_t) \tag{4.5}$$

where X_{t-1} is a continuous representation , i.e. embedding, of the word w_{t-1}, V encodes the recurrent connection weights and W is the hidden-to-output connection weights. These parameters define the network and are learned during training. Moreover, $f(\cdot)$ is an activation function, and $g(\cdot)$ is the softmax function. The right panel of Figure 3.1 shows an example of the standard RNN architecture.

Theoretically, the recurrent connections of an RNN allow the context to indefinitely cycle in the network and thus, modeling long context. In practice, however, Le, Allauzen, and Yvon (2012) have shown that this information changes quickly over time, and that it is experimentally equivalent to an 8-gram FNN. This observation was confirmed by the experiments that reported in this chapter.

Long-Short Term Memory network

In order to alleviate the rapidly changing context issue in standard RNNs and control the longevity of the dependencies modeling in the network, the LSTM architecture (Sundermeyer, Schlüter, and Ney, 2012) introduces an internal memory state C_t, which explicitly controls the amount of information, to forget or to add to the network, before estimating the current hidden state. Formally, this is done according to

$$\{i, f, o\}_t = \sigma\Big(U^{i,f,o} \cdot X_{t-1} + V^{i,f,o} \cdot H_{t-1}\Big) \tag{4.6}$$

$$\tilde{C}_t = f(U^c \cdot X_{t-1} + V^c \cdot H_{t-1}) \tag{4.7}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{4.8}$$

$$H_t = o_t \odot f(C_t) \tag{4.9}$$

$$P_t = g(W \cdot H_t) \tag{4.10}$$

where $\odot$ is the element-wise multiplication operator, $\tilde{C}_t$ is the memory candidate, whereas i_t, f_t and o_t are the input, forget and output gates of the network, respectively. Figure 4.1 illustrates the recurrent module of an LSTM network. Tuning an LSTM model requires the training of the network parameters $U^{i,f,o,c}$, $V^{i,f,o,c}$ and W.

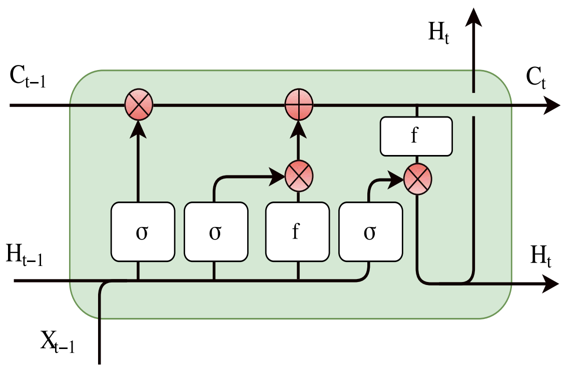


Figure 4.1: Block diagram of a recurrent node in an LSTM network.

Although LSTM models have been shown to outperform classical RNN in modeling long range dependencies, they do not explicitly model long / short context but rather use a single state to encode the global linguistic context.

4.3 Multi-span language models

The attempts to learn and combine short and long range dependencies in language modeling led to what is known as multi-span LMs (Bellegarda, 1998b). The goal of these models is to learn the various constraints, both local and global, that are present in natural language.

This is typically done using two different models, which separately learn the local and global context, and then combine their resulting linguistic information to perform the word prediction. For instance, Bellegarda (1998a) proposed to use Latent Semantic Analysis (LSA) to capture the global context, and then combine it with the standard n-gram models, which capture the local context, whereas Mikolov and Zweig (2012) proposed to model the global topic information using Latent Dirichlet Allocation (LDA), which is then combined with an RNN-based LM. This idea is not particular to language modeling but has been also used in other Natural Language Processing (NLP) tasks, e.g., Anastasakos, Kim, and Deoras (2014) proposed to use a local / global model to perform a spoken language understanding task.

4.3.1 Long-Short Range Context network

Following the line of thought in Bellegarda (1998a) and Mikolov and Zweig (2012), my coauthors and I put forward a new multi-span model, which takes advantage of the LSTM ability to model long range context while, simultaneously, learning and integrating the short context through an additional recurrent, local state. In doing so, the resulting Long-Short Range Context (LSRC) network is able to separately model the short / long context while it dynamically combines them to perform the next word prediction. Formally, this new model is defined as

$$H_t^l = f\Big(X_{t\text{-}1} + U_l^c \cdot H_{t\text{-}1}^l\Big) \tag{4.11}$$

$$\{i, f, o\}_t = \sigma\Big(V_l^{i,f,o} \cdot H_t^l + V_g^{i,f,o} \cdot H_{t-1}^g\Big) \tag{4.12}$$

$$\tilde{C}_t = f(V_l^c \cdot H_t^l + V_g^c \cdot H_{t-1}^g) \tag{4.13}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{4.14}$$

$$H_t^g = o_t \odot f(C_t) \tag{4.15}$$

$$P_t = g(W \cdot H_t^g) \tag{4.16}$$

Tuning an LSRC model requires the training of the local parameters $V_l^{i,f,o,c}$ and U_l^c, the global parameters $V_g^{i,f,o,c}$, and the hidden-to-output connection weights W. This can be done using the standard Back-Propagation Through Time (BPTT) algorithm, which is typically used to train recurrent networks.

The LSRC approach uses two hidden states, namely, H_t^l and H_t^g to model short and long range context, respectively. More particularly, the local state H_t^l evolves according to Equation 4.11 which is nothing but a simple recurrent model as it is defined in Equation 4.4. In doing so, H_t^l is expected to have a similar behavior to RNN, which has been shown to capture local / short context (up to 10 words), whereas the global state H_t^g follows the LSTM model, which is known to capture longer dependencies (see example in Figure 4.3). The main difference here, however, is the dependence of the network modules (gates and memory candidate) on the previous local state H_t^l instead of the last seen word $X_{t\text{-}1}$. This model is based on the assumption that the local context carries more linguistic information, and is therefore, more suitable to combine with the global context and update the LSTM, compared to the last seen word. Figure 4.2 illustrates the recurrent module of an LSRC network. It is worth mentioning that this model was not particularly developed to separately learn syntactic and semantic information. This may come, however, as a result of the inherent local and global nature of these two types of linguistic properties.

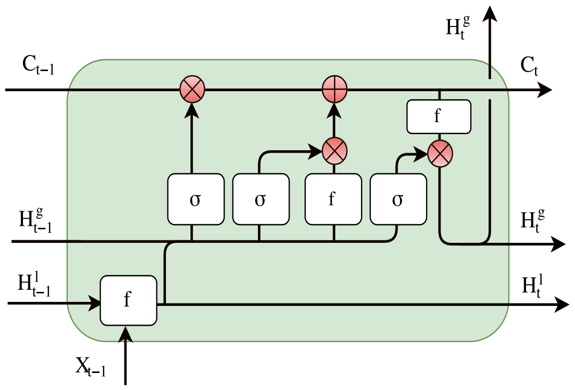


Figure 4.2: Block diagram of a recurrent node in an LSRC network.

4.4 Context range estimation

For many NLP applications, capturing the global context information can be a crucial component to developing successful systems. This is mainly due to the inherent nature of language, where a single idea or topic can span over few sentences, paragraphs or a complete document. LSA-like approaches take advantage of this property, and aim at extracting some hidden “concepts” that best explain the data in a low-dimension “semantic space”. To some extent, the hidden layer of LSRC / LSTM can be seen as a vector in a similar space. The information stored in this vector, however, changes continuously based on the processed words. Moreover, interpreting its content is generally difficult. As an alternative, measuring the temporal correlation of this hidden vector, i.e. the extent to which the vector changes over time, can be used as an indicator of the ability of the network to model short and long context dependencies. Formally, the temporal correlation of a hidden state H over a distance d is given by

$$c_d = \frac{1}{D}\sum_{t=1}^{D} SM(H_t, H_{t+d}) \tag{4.17}$$

where D is the number of tokens in the test corpus and SM is a similarity measure such as *cosine similarity*. This measure allows us to evaluate how fast the information stored in the hidden state changes over time.

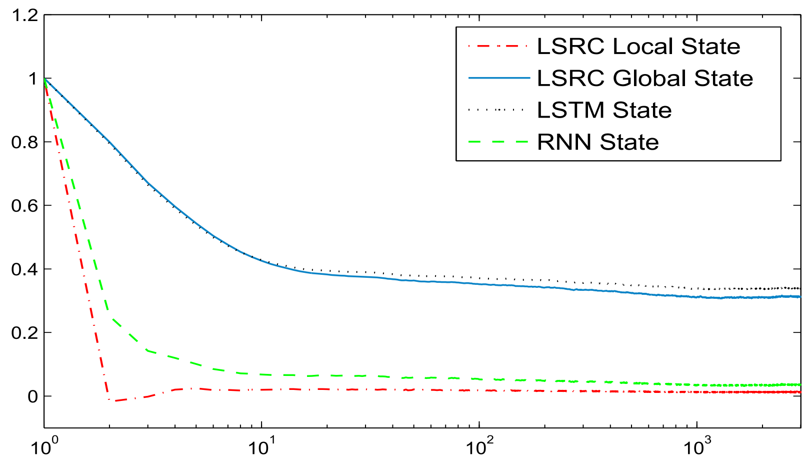


Figure 4.3: Temporal correlation of LSRC states in comparison to LSTM and RNN.

Figure 4.3 shows how the variation of this temporal correlation for the local and global states of the LSRC network in comparison to RNN and LSTM for various values of the distance d (up to 3000). This figure was obtained on the test set of the Penn Treebank (PTB) corpus, described in Section 4.5. The main conclusion that my coauthors and I drew from this figure is the ability of the LSRC local and global states (trained jointly) to behave in a similar fashion to RNN and LSTM states (trained separately). My coauthors and I concluded that the LSRC global state and LSTM are able to capture long range correlations, whereas the context changes rapidly over time in RNN and LSRC local state.

4.5 Experiments and results

4.5.1 Experimental setup

As in the two previous chapters, my coauthors evaluated the LSRC architecture on the Penn Treebank (PTB) and Large Text Compression Benchmark (LTCB) corpora. Details about the sizes of these two corpora can be found in Table 4.1.

Corpus	**Train**	**Dev**	**Test**
PTB	930K	74K	82K
LTCB	133M	7.8M	7.9M

Table 4.1: Corpus size in number of tokens.

My coauthors and I decided for this project to use just a single end sentence tag between each two consecutive sentences, rather than use a separate begin sentence tag[1]. Also, my coauthors used the same baseline models as discussed in Section 3.4.

[1]This explains the difference in the corpus size compared to the one reported in Zhang et al. (2015).

4.5.2 PTB experiments

For the PTB experiments, the FNN and FOFE models use a word embedding size of 200, whereas the hidden layer(s) size is fixed at 400, with all hidden units using the rectified linear unit, i.e. $f(x) = \max(0, x)$ as activation function. My coauthors also used the same learning setup adopted in Zhang et al. (2015). Namely, the mini-batch size for stochastic gradient descent algorithm was 200. The learning rate was initialized to 0.4, the momentum was set to 0.9, the weight decay was fixed at $4 \cdot 10^{-5}$, and the training was done in epochs. The weights initialization followed the normalized initialization proposed by Glorot and Bengio (2010). Similarly to Mikolov et al. (2010), the learning rate was halved when no substantial improvement of the validation data log-likelihood was observed. Then, training continued with seven more epochs while halving the learning rate after each epoch.

Regarding the recurrent models, my coauthors used $f = \tanh(\cdot)$ as the activation function for all recurrent layers, whereas $f = \text{sigmoid}(\cdot)$ was used for the input, forget, and output gates of LSTM and LSRC. The additional non-recurrent layer in D-LSRC, however, used the rectified linear unit activation function. The word embedding size was set to 200 for LSTM and LSRC whereas it was the same as the hidden layer size for RNN (result of the RNN equation 4). To illustrate the effectiveness of the LSRC model, the experiments show the results when the embedding size was fixed at 100, LSRC(100). The training used the BPTT algorithm for 5 time steps. Similarly to short context models, the mini-batch was set to 200. The learning rate, however, was set to 1.0 and the weight decay to $5 \cdot 10^{-5}$. The use of momentum did not lead to any additional improvement. Moreover, the data were sequentially without any sentence independence assumption. Thus, the recurrent models were still able to capture long range dependencies that existed beyond the sentence boundary.

Architecture	**model**			**model+KN5**			**NoP**
$n-1=$	1	2	4	1	2	4	4
KN	186	148	141	–	–	–	–
KN+cache	168	134	129	–	–	–	–
	1 Hidden Layer						
FNN	176	131	119	132	116	107	6.32M
FOFE	123	111	112	108	100	101	6.32M
	Recurrent Models (1 Layer)						
RNN	117			104			8M
LSTM (1L)	113			99			7M
LSRC(100)	109			96			6M
LSRC(200)	104			94			7M
	2 Hidden Layers						
FNN	176	129	114	132	114	102	6.96M
FOFE	116	108	109	104	98	97	6.96M
	Deep Recurrent Models						
D-LSTM (2L)	110			97			8M
D-RNN (3L)	108			–			6M
D-LSRC(100)	103			93			6M
D-LSRC(200)	102			92			7M

Table 4.2: Comparison of LSRC and other model perplexities (PPL) on the PTB test set.

Table 4.2 shows the perplexity evaluation on the PTB test set. To compare the model sizes, it also reports the Number of Parameters (NoP) for each of the models. My coauthors and I first observed that LSRC outperformed all other models for all configurations, in particular, RNN and LSTM. This observation includes other models that were reported in the literature, such as random forest LM (Xu and Jelinek, 2007), structured LM (Filimonov and Harper, 2009), and syntactic neural network LM (Emami and Jelinek, 2004). More particularly, my coauthors and I concluded that LSRC, with an embedding size of 100, achieved a better performance than all other models while reducing the number of parameters by $\approx 29\%$ and $\approx 17\%$ compared to RNN and LSTM, respectively. Increasing the embedding size to 200, which is used by the other models, significantly improved the performance with a resulting NoP comparable to LSTM. A t-test confirmed the significance of the improvements obtained over LSTM, which led to p-values $\leq 10^{-10}$.

The results of the deep models in Table 4.2 also show that adding a single non-recurrent hidden layer to LSRC can substantially improve the performance. In fact, the additional layer bridges the gap between the LSRC models with an embedding size of 100 and 200. The resulting architectures outperform the other deep recurrent models with a substantial reduction in the number of parameters (for the embedding size 100), and without dropout regularization, L_p, and maxout units or gradient control techniques from D-RNN.

My coauthors and I concluded from these experiments that the explicit modeling of short and long range dependencies using two separate hidden states improves the performance while reducing the number of parameters.

In order to show the consistency of the LSRC improvement over the other recurrent models, Figure 4.4 illustrates the variation of the models performance with respect to the hidden layer size. This figure shows that increasing the LSTM or RNN hidden layer size could not achieve a similar performance to the one obtained using LSRC with a small layer size (e.g. 300). It is also worth mentioning that this observation holds when comparing a 2-recurrent layers LSTM to LSRC with an additional non-recurrent layer.

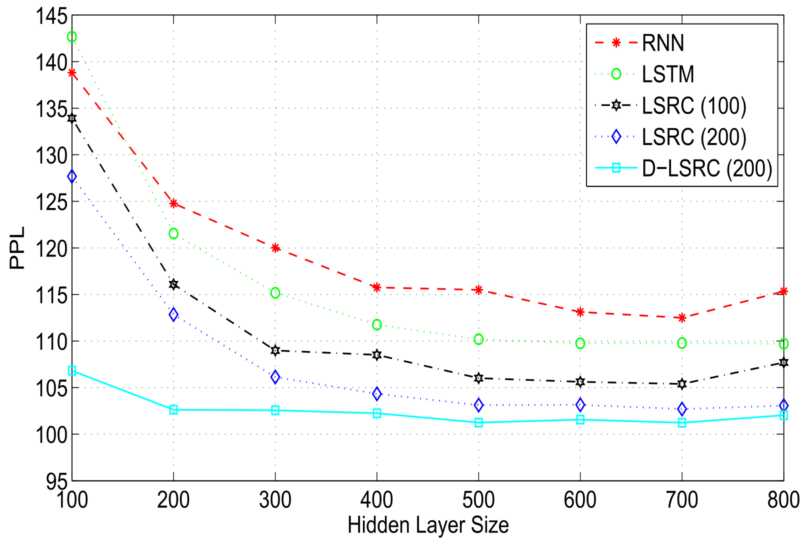


Figure 4.4: Perplexity of neural LMs versus hidden layer size on PTB.

4.5.3 LTCB experiments

Our LTCB experiments used the same PTB setup with minor modifications and followed exactly the same setup as Zhang et al. (2015). More precisely, these results were obtained without use of momentum or weight decay (due to the long training time required for this corpus), the mini-batch size was set to 400, the learning rate was set to 0.4 and the BPTT step was fixed at 5. The FNN and FOFE architectures use 2 hidden layers of size 600, whereas RNN, LSTM and LSRC have a single hidden layer of size 600. Moreover, the word embedding size was set to 200 for all models except RNN, which was set to 600.

The PTB and LTCB results clearly highlight the importance of recurrent models to capture long range dependencies for LM tasks. The training of these models, however, required large amounts of data to outperform short context models. This can be seen in the performance of RNN and LSTM on both corpora. My coauthors and I concluded from these results that the explicit modeling of long and short context in a multi-span model can lead to a substantial improvement over state-of-the-art models.

Architecture	**model**			**NoP**
$n-1=$	1	2	4	4
KN	239	156	132	–
KN+cache	188	127	109	–
FNN [M*200]-600-600	235	150	114	65M
FOFE [M*200]-600-600	112	107	100	65M
RNN [600]-R600		85		96M
LSTM [200]-R600		66		66M
LSTM [200]-R600-R600		61		69M
LSRC [200]-R600		63		66M
LSRC [200]-R600-600		59		66M

Table 4.3: Comparison of LSRC and other model perplexities (PPL) on the LTCB test set.

Table 4.3 reports results for an LSTM with 2 recurrent layers as well as for LSRC with an additional non-recurrent layer. The recurrent layers are marked with an "R".

The results shown in Table 4.3 generally confirm the conclusions drawn from the PTB experiments above. In particular, the proposed LSRC model largely outperformed all other models. In particular, LSRC clearly outperformed LSTM with a negligible increase in the number of parameters (resulting from the additional $200 \cdot 200 = 0.04M$ local connection weights U_l^c) for the single layer results. This improvement was maintained for deep models (2 hidden layers), where the LSRC model achieved a slightly better performance while reducing the number of parameters by $\approx 2.5M$ and speeding up the training time by $\approx 20\%$ compared to deep LSTM.

4.6 Conclusion and synthesizing discussion

This chapter considers the importance and ability of standard neural networks to encode long and short range dependencies for language modeling tasks. I argue that these models were not particularly designed to, explicitly and separately, capture these two sources of linguistic information. LSRC was a good alternative solution, taking advantage of the LSTM ability to capture long range dependencies and the RNN ability to encode a much shorter range of context. In doing so, this network is able to encode the short and long range linguistic dependencies using two separate network states that evolve in time. In experiments conducted on PTB and LTCB, LSRC models substantially outperformed different state-of-the-art neural network architectures, including LSTM and RNN, even when smaller architectures were used.

In work following the LSRC publication, I found experimentally that LSRC models consistently outperformed LSTM models without dropout. But if dropout was included, performance was comparable. It was interesting that an extra recurrent connection would have such paradoxical purposes in relating the three novel architectures in this section. Namely, recurrences were not present in CDLM, used for the long context in SRNNs, maintained the short context in LSRC models, and seemed comparable to probabilistically *deleting* information in a network without the extra recurrence. As such, it seemed ill-advised to invest too much energy in arguing for one architecture over another except in quite specialized use cases.

The most important limitation of this work is that perplexity experiments are not that useful for making sweeping inferences about the qualities of language models. For a host of reasons, a perplexity experiment never "proves" that one language model is better than another. The foremost reason is that the outcome of the experiment is so largely dependent on the training data, the testing data, hyperparameter settings, the choice of vocabulary, and further decisions that the experimenter makes such as when to stop training. Indeed, without spending perhaps an imprudent amount of time searching the hyperparameter space, it is difficult to know when the language model is "good enough". This problem has become more of an issue in the years following my publications. Currently, language models are so big that it is impractical to train them from scratch, so the hyperparameter space is not explored. Datasets are so large that scientists cannot know that much about what they contain, leading some, notably Bender and Gebru et al. (2021), to argue that scientists should only create corpora and language models "as large as can be sufficiently documented."

That said, I can still make recommendations for when to use different language models, provided that the standard of "proof" is lower for recommendations. Especially with the subsequent advent of transformer language models, I would recommend LSTM and its close variants (LSRC) mainly in the case that the data are too specialized or rare to support a Transformer model. Then, if the resources are still not sufficient, a CDLM might work. Given though that Transformer models tend to be in a different league in terms of model quality, it did not seem as important to practice experimental methods of language model evaluation, such as coreference resolution or batteries of subject verb agreement tests, on all three novel architectures in this thesis. I chose to use only LSRC as this model is closest to LSTMs, which enjoyed relative hegemony in the pre-Transformer era of language modeling research.

But the story of the development of these three novel architectures is still interesting and informs how long distance triggering patterns came to be commonly included in language models. The three architectures built on each other in that CDLM had the minimal power needed to capture long distance dependencies; SRNN added more power, achieved state of the art, and provided a mechanism for semantically and syntactically oriented embeddings from the same system; and LSRC developed to the point that there was actual data showing that information such as triggering information was remaining in the network as intended.

Chapter 5

Language models can use spelling to approximate the meanings of words

Byte-pair encoding (BPE), as formulated for compression by Gage (1994) and "popularized…as a tokenization scheme" by Sennrich, Haddow, and Birch (2016), has become somewhat of a standard in tokenization for language modeling (Zouhar et al., 2023). It essentially solves the problem of out-of-vocabulary tokens having undefined surprisal and allows the system to use sub-word information. However, even this is perhaps more brittle than the human capacity to handle novel words, since they can flexibly divide up words and make associations as they see fit. Sub-word similarity-based search (SWordSS), which my coauthors and I presented in Singh et al. (2016), aimed to sidestep at least the word division problem by exploring overlapping sub-word units of a fixed size. Therefore, no specialized knowledge of morphology was necessary. In my individual follow-up studies, I found, perhaps surprisingly, that even in some cases that a representation of the entire word can be obtained, it was better to use SWordSS on the word's spelling to approximate the word's meaning.

5.1 Introduction and background

Word embeddings have been successfully applied to many NLP tasks (Collobert and Weston, 2008; Collobert, 2011; Socher et al., 2011, 2012; Hermann and Blunsom, 2014; Bengio and Heigold, 2014; Yang et al., 2015), and these systems often achieved state-of-the-art performance. This success has been ascribed to embeddings' ability to capture regularities traditionally represented in core NLP features. Most of these embeddings were trained on large amounts of data, allowing them to have good coverage of the relevant vocabularies. However, embeddings often still cannot satisfactorily represent rare words, i.e. words with few occurrences in training data.

To generate useful embeddings for words too rare for standard methods, Luong, Socher, and

Manning (2013) and Botha and Blunsom (2014) leveraged the segmentation tool, `Morfessor` (Creutz and Lagus, 2005), while Cotterell, Schütze, and Eisner (2016) used morphological lexica to generate rare-word embeddings. In general, these methods added resource-based knowledge to their systems in order to form word vector representations, showing impressive performance gains over methods which did not address the rare words problem.

In contrast, Soricut and Och (2015) applied an automatic method to induce morphological rules and transformations as vectors in the same embedding space. More specifically, they exploited automatically-learned prefix- and suffix-based rules using the frequency of such transformations in the data and induced a morphological relationship-based word graph. Then, they searched over this graph for rules that best infer the morphology of the rare words. The embeddings were then estimated using these rare-word explaining rules. In this method, creating and tuning this morphological graph could lead to a high initial cost.

To overcome this cost and still be able to automatically induce rare word representations, my coauthors and I proposed a sub-word similarity-based search. This technique maps a rare word to a set of its orthographically-similar words and combines the embeddings of these similar words to generate the rare word's representation. These generated embeddings can then be combined with existing word embeddings to be applied in various tasks. The SWordSS method is distinctive in that it improves embeddings for rare words without requiring additional knowledge, data, or expensive computation.

In Section 5.3, I evaluate SWordSS embeddings on word similarity tasks. For further evaluation, in Section 5.4, I describe how my coauthor injected SWordSS embeddings into a language model and analyzed its perplexity on rare words over various corpora. In Section 5.5, I take a more quantitative look at how common it is for words to be rare and what proportion of words can be expected to see a benefit from this method. Finally, I summarize the findings and draw connections to more recent approaches in Section 5.6.

5.2 Inducing Rare Word Embeddings

Rare words form a large part of a language's vocabulary. Table 5.1 reports that large portions of the vocabularies for several languages in *Polyglot* (Al-Rfou', Perozzi, and Skiena, 2013) are words that occurred once or not at all in training data. Further, it is widely known that in English, roughly half of all tokens in a given corpus occur only once.

Language	V	RW	#ENF	**Coverage (%)**
German	36602	15715	13103	99.9
Tagalog	22492	10568	8407	98.1
Turkish	24840	13624	9555	99.0
Vietnamese	6423	1332	305	69.1

Table 5.1: Some statistics on corpora in several languages used for language modeling.

In this chapter, I define that a word is rare if it occurs zero or one time in a training corpus. Further, if it did not occur at all, it is considered *out of vocabulary*.

Increasing the size of the corpus generally does not reduce the proportion of rare words in the vocabulary because the distribution of words is Zipfian (Zipf, 1936, 1949). That is, there is an unavoidable long tail. Thus, it is essential to handle rare words properly to obtain good performance.

In the context of word embeddings-related tasks, training good word embeddings can incur huge computational costs (Al-Rfou', Perozzi, and Skiena, 2013). So, the SWordSS approach focuses on augmenting readily available embeddings sets rather than creating new ones from scratch. To increase the availability of resources for many languages, Al-Rfou', Perozzi, and Skiena (2013) released[1] pre-trained word embeddings for more than 100 languages. These pre-trained word embeddings, namely *Polyglot*, were constructed by applying the method outlined in Bengio et al. (2009) on Wikipedia texts, which vary in size from millions of tokens to a few billion tokens.

Among many available pre-trained word embeddings, Google released `word2vec` (Mikolov et al., 2013) based embeddings[2] trained on their English News dataset (about 100 billion tokens). The experiments described in this chapter applied both of these embeddings sets to jump start generating the rare word embeddings for different languages.

Statistics about the various language modeling corpora and word similarity tasks used in the experiments are shown in Table 5.1 and Table 5.2. These tables report the vocabulary size, number of rare words, and the number of words for which the embeddings were not found (ENF = Embedding Not Found) in the pre-trained embedding sets. For most corpora and evaluation tasks, ENFs form a large share of the vocabulary. So, SWordSS produced the embeddings for these words.

Dataset	V	#ENF	**Coverage (%)**
Rare Word (Luong, Socher, and Manning, 2013)	2951	1073	100
Gur65 (Gurevych, 2005)	49	4	100
Rare Word + GoogleNews	2951	173	100

Table 5.2: Some statistics on word similarity datasets used in the experiments.

[1]`https://sites.google.com/site/rmyeid/projects/polyglot`
[2]`https://code.google.com/archive/p/word2vec/`

For a given set of pre-trained embeddings with a finite vocabulary V_E applied to a task with vocabulary V_T and a finite set of given rare words $RW = \{w \mid w \notin V_E \wedge w \in V_T\}$, SWordSS applies the following steps:

Step 1: Map every word $w \in V_T$ to its sub-word features.

Although a word may be rare, substrings of that word are, in general, less rare. Hence, the process starts by breaking down each word $w \in V$ into its constituent N-sized sub-word units: $D_N(w)$. For example, given the sub-word size $N = 3$,

$$D_N(language) = \{lan, ang, ngu, gua, uag, age\}$$

In this chapter's experiments, N was set to 3. However, it remains to be seen how using differently sized sub-word units or even morphemes affects the performance of this method. Note that this procedure does not formally require that sub-word units be of equal length, so linguistically-sensible morphemes may be used if the resource is available for that language.

Step 2: Index $w \in V_T$ using its sub-word features.

Pre-trained sets of embeddings can have large coverage at the outset (for example, *Polyglot* has 100K words in its vocabulary). So, performing substring searches and comparisons can become quite computationally expensive. To speed up the search for sub-word units, my coauthor created an inverted index on words. For each $w \in V$, $D_N(w)$ was treated as a document and fed into a search engine based indexer. Lucene[3] (McCandless, Hatcher, and Gospodnetic, 2010) was used to index the words.

Step 3: Search the index for matches of $w' \in RW$.

Next, SWordSS breaks down the rare word $w' \in RW$ into its sub-word units ($D_N(w')$) and searches for $D_N(w')$ on the index. The program returns the top $K = 10$ results, denoted by $R^K(w')$. This contains words having similar sub-word units as w', hence, containing words which are sub-word similar to w'.

Step 4: For every $w' \in RW$, combine matched embeddings to generate its embedding.

To compute the SWordSS embedding of $w' \in RW$, the system computes the weighted average of embeddings (v) of the rare-word matches. A string similarity function S provides the weights such that

$$v_{w'} = \sum_{w:D_N(w) \in R^K(w')} S(w', w) \cdot v_w$$

The above method particularly hinges on the third step, which uses sub-word similarity of sub-word similar words to search for rare word alternatives, leading to the embedding combination in the fourth step. Hence, my coauthor named the technique **S**ub-**Word S**imilarity based **S**earch (**SWordSS**: pronounced swordz). The SWordSS embeddings ($\{v_{w'} : w' \in RW\}$) are used along with $\{v_w : w \in V_E\}$ to perform rare word-related tasks.

[3]`https://lucene.apache.org/`

For the fourth step, my coauthor and I both independent tried different string similarity functions (S), described in the list below, to average different embeddings of matches from the third step. These different similarity functions helped to provide a more morphologically-sensible scoring of matches and eventually set the weights for computing the final rare word embeddings.

Jaccard Index Jaccard (1912) computes the size of the character intersection over the size of the character union. Therefore, order of characters is not considered by this metric. Frequent characters such as vowels lead to uninteresting intersections, and short words could possibly suffer from an unfair floor.

Jaro Similarity Jaro (1989) considers the number of matching characters in corresponding positions and the number of transpositions detected. So, order of characters does matter for this metric. Insertions and deletions are treated similarly, and the frequency and length effects from Jaccard could also affect this metric.

Jaro-Winkler Similarity Winkler (1990) developed a variation on Jaro similarity and quite importantly, this metric deems two strings more similar if a short prefix (at the beginning of the strings) appears on both strings. This is especially relevant for this work because it benefits precisely the case in which two strings differ only in an inflectional suffix.

Most Frequent K Characters Similarity Seker et al. (2014) consider the counts of the top K characters in each string. Thus, if the "root morphemes" are long enough to create nontrivial count statistics, this metric may, too, favor a more linguistic similarity, but as before, shorter strings could have unwanted effects.

Subsequence Kernels Lodhi et al. (2002) create automatically-generated features based on sequences of characters within the strings to be compared. Therefore, those sequences that do not cross morpheme boundaries could be especially helpful for estimating morphological similarity.

Tversky Coefficient Tversky (1977) breaks down the union in the Jaccard Index, allowing different weights for the denominator intersection, those characters that only appear in the first string, and those characters that only appear in the second string. These metaparameters allow the metric some flexibility that the others do not have.

In experiments on rare word-related tasks, my coauthor and I mostly observed that using SWordSS leads to high coverage rates, also presented in Table 5.2 and Table 5.1. Whenever words w' resulted in zero matches, I removed them from the word similarity tasks and my coauthor substituted in random vectors for the language modeling tasks.

5.3 Correlations with human word similarity scores

Datasets I evaluated SWordSS embeddings on two word similarity tasks. The evaluation metric is the correlation between the human ratings of word pairs and the scores generated using embeddings. A good set of embeddings should yield a high correlation.

Specifically, I evaluated SWordSS embeddings on Luong, Socher, and Manning (2013)'s English Rare Words dataset with 2034 word pairs (Luong2034) and also evaluated these embeddings on a German word similarity task (Gurevych, 2005; Zesch and Gurevych, 2006) with 65 word pairs (Gur65).

Experimental Setup For the German word similarity task, I used only *Polyglot* word embeddings, which have 64 dimensions. For English, along with *Polyglot* word embeddings, I used the GoogleNews `word2vec` embeddings, which are 300-dimensional vectors.

As a baseline, I used the existing pre-trained word embeddings, which are compared to their augmented SWordSS versions. While augmenting with the pre-trained sets the SWordSS embeddings, I also explored the string similarity functions given in the previous section. To evaluate the effect of these string similarity functions, I also implemented a constant similarity function ($S(w, w') = 1$, where w and w' are words), denoting the corresponding embeddings by SWordSS$_1$. Finally, I also compared the SWordSS embeddings to SO2015 (Soricut and Och, 2015), which also applied morphological analysis to generate missing word embeddings in a way quite similar to that of SWordSS.

Word Vectors	Gur65
Polyglot	28.5
Polyglot+SWordSS$_{ji}$	37.5
Polyglot+SWordSS$_{jaro}$	37.1
Polyglot+SWordSS$_{jw}$	37.0
Polyglot+SWordSS$_{mfk}$	37.2
Polyglot+SWordSS$_{ssk}$	36.9
Polyglot+SWordSS$_{tc}$	**37.6**
Polyglot+SWordSS$_1$	35.8

Table 5.3: Spearman $\rho \times 100$ with human ratings on a German word similarity task (Gur65) for SWordSS embeddings using various string similarity functions.

Results Using SWordSS embeddings definitely increased the correlation with humans in comparison to the original on the Gur65 task (shown in Table 5.3), though all the different string similarity functions except the constant function (SWordSS$_1$) led to correlations in a very close range, showing that particularly for German, different similarity functions behave very similarly. So, I only report the best correlation after applying these functions.

Task	Luong2034	
Word Vectors	*Polyglot*	*GoogleNews*
SO2015 w/o Morph	-	44.7
SO2015 w/ Morph	-	**52.0**
w/o SWordSS	9.7	45.3
w/ SWordSS$_1$	28.9	51.3
w/ SWordSS$_{sim}$	30.4	51.4

Table 5.4: Spearman $\rho \times 100$ evaluation of techniques with and without morphological features used to generate representations for the word similarity task.

Next, I compared SWordSS versions of *Polyglot* embeddings and GoogleNews embeddings on the Luong2034 task. When the SWordSS versions were compared to the original (labelled w/o SWordSS) it led to a higher correlation, as shown in Table 5.4. However, for each set of embeddings, the difference between SWordSS$_1$ and SWordSS$_{sim}$ remained small. The correlations for the SWordSS version of *Polyglot* were still lower than the correlation rates reported by SO2015. This was due to the difference in initial quality of embeddings used by each method. As *Polyglot* embeddings trained on a lesser amount of data than SO2015, they were easily outperformed.

Also in Table 5.4, I show the result of substituting *Polyglot* embeddings with *GoogleNews* embeddings. Using these embeddings, which were trained on a larger dataset than used by *Polyglot*, led to SWordSS versions having comparable results with SO2015 for the Luong2034 task.

Overall, the SWordSS technique was able to improve pre-trained embeddings' performance on the above word similarity tasks quite drastically. Even though SWordSS-augmented GoogleNews embeddings did not significantly outperform SO2015, this method provides a simpler sub-word search based alternative to the graph search over morphological relationships performed by SO2015. Furthermore, by applying sub-word search in the third step from Section 5.2, SWordSS overcomes the need for creating and tuning the graph of morphological relationships as required by SO2015.

5.4 Perplexity Experiments

Training language models (LMs) using an expanded vocabulary (having more word types than contained in the training corpus) requires assigning probabilities to words which are not present in the training set. Traditionally, these rare words are assigned a default value of probability in conventional n-gram and long short term memory (LSTM)-based reccurrent neural network LMs (Sundermeyer, Schlüter, and Ney, 2012). This is usually not beneficial for spoken term detection and automatic speech recognition systems made for low resourced languages, since presence of rare words in speech queries is high (Logan et al., 1996; Logan, Van Thong, and Moreno, 2005).

To avoid this misrepresentation of rare words, my coauthor applied SWordSS embeddings in a language modeling framework. Specifically, he used a log-bilinear language model (LBL) (Mnih and Hinton, 2007). In the experiments, when the SWordSS embeddings were used to initialize an LSTM's input layer, the system obtained the same perplexity values as the LSTM initialized with random embeddings. This observation suggests that the LBL framework is better suited than LSTMs for this naive way of initializing neural language models with SWordSS embeddings and improving perplexity on rare words.

LBL predicts the next word vector $p \in \mathbb{R}^d$, given a context of $n-1$ words, as a transformed sum of context word vectors $q_j \in \mathbb{R}^d$, as:

$$p = \sum_{j=1}^{n-1} q_j C_j$$

where $C_j \in \mathbb{R}^{d \times d}$ are position-specific transformation matrices. p is compared with the next word w's representation r_w. This comparison is performed using the vector dot product and then is used in a softmax function to obtain the probability of the next word as follows:

$$p(w_i | w_{i-n+1}^{i-1}) = \frac{\exp(p \cdot r_w + b_w)}{\sum_{v \in V} \exp(p \cdot r_v + b_v)}$$

where b is the bias term encoding the prior probability of word type w.

First, Q, the collection of context word vectors q_j, and R, the collection of next word representations (r_w), are initialized with the pre-trained word embeddings. Thereafter, my coauthor trained the LBL using stochastic gradient descent.

Previously, extensions to class-based and factor-based formulations have provided impressive improvements over regular n-gram LMs for morphological languages (Botha and Blunsom, 2014). But, these LMs do not provide straightforward ways of incorporating pre-trained word embeddings, so my coauthor used the original LBL because of the ease with which it incorporates pre-trained embeddings in its formulation.

Data To evaluate SWordSS embeddings on language modeling, my coauthor used the German section of EUROPARL as processed by Botha and Blunsom (2014). He also performed experiments on Tagalog (tl), Turkish (tr) and Vietnamese (vi) corpora, which include transcriptions of phone conversations collected under the IARPA Babel Program language collection releases `babel106b-v0.2f`, `babel105-v0.5`, and `babel107b-v0.7` respectively.

The German corpus was processed to have no out-of-vocabulary words (OOVs), however, it still had a lot of low frequency words (see Table 5.1). Contrastingly, the Babel corpora have OOVs as well as other low frequency words.

The Babel corpora were provided with training and development sets. My coauthor divided the existing development set into two halves to use one as the test set and the other half as the new development set. The statistics on these corpora are summarized in Table 5.5.

Statistics	de	tl	tr	vi
Train	1000K	585K	239K	985K
Dev	74K	30K	5K	65K
Test	73K	31K	6K	60K
Voc Size	37K	22K	25K	6K

Table 5.5: Statistical summary of corpora used for the language modeling experiments.

Table 5.1 and Table 5.2 showed that even though a lot of rare-word embeddings were missing from the pre-trained set, SWordSS was able to generate and obtain high coverage rates for such words, giving this method added benefit in the context of rare words.

Experimental Setup Before evaluating the SWordSS embeddings for predicting rare words, my coauthor used all the OOVs to expand the corresponding vocabulary. SWordSS embeddings for all the words in the expanded vocabulary were used to initialize LBL framework as described in Section 5.4. A bigram version of this LBL (LBL2$_{SWordSS}$) was further trained on language corpora before being evaluated.

My coauthor compared the LBL2$_{SWordSS}$ model with the conventional modified Kneser-Ney 5-gram LM (KN5) (Kneser and Ney, 1995; Chen and Goodman, 1996) and also with the bigram (LBL2) based log-bilinear LM. As a more powerful baseline, he also trained an LSTM based RNN LM to compare with LBL2$_{SWordSS}$. Moreover, my coauthor and I compared the LBL2$_{SWordSS}$, with a character-aware LM (Kim et al., 2016), denoted as CCNN-LSTM. The CCNN-LSTMs were chosen for comparison because of their ability to use character-based features to implicitly handle OOVs and rare words. For training each of these LMs, my coauthor used the expanded vocabulary as used by LBL2$_{SWordSS}$. In training neural network-based language models, he restricted the number of parameters to have a similar number of parameters as LBL2$_{SWordSS}$.

Language Model	German		Tagalog		Turkish		Vietnamese	
	PPL	RW1PPL	PPL	RW1PPL	PPL	RW1PPL	PPL	RW1PPL
KN5	364.2	559K	162.6	420K	478.9	139K	120.8	174K
LBL2	391.1	404K	171.4	204K	649	**94K**	137.6	**100K**
LSTM	323.1	596K	134.7	343K	489.8	110K	**102.1**	457K
CCNN-LSTM	**315.7**	636K	**117.4**	354K	**408.7**	168K	182.7	516K
LBL2$_{SWordSS}$	369.4	**260K**	167.2	**167K**	513.2	110K	136.4	143K
NoP	4.7 M		2.9 M		3.2 M		0.8 M	

Table 5.6: Perplexities on test set (PPL), RW1 perplexities (RW1PPL) in thousands and number of parameters (NoP) for LBL and LSTM LMs in millions, presented on four corpora.

General Perplexity Results This section compares the language models described above using perplexity values calculated on test sets of different languages, shown in Table 5.6.

As shown in Table 5.6, LBL2$_{SWordSS}$ was able to outperform the conventional LBL2 comfortably on all the corpora except Vietnamese. For Vietnamese, LBL2$_{SWordSS}$ and LBL2 performed comparably. Due to SWordSS' low coverage of Vietnamese vocabulary, initializing LBL2 with SWordSS embeddings led to only a marginal performance gain.

Overall in terms of test set perplexity, CCNN-LSTM outperformed LBL2$_{SWordSS}$ comfortably on most language corpora. However, on Vietnamese (in which characters represent meaning units rather than sounds) CCNN-LSTM suffered and the LSTM outperformed the other language models. In comparison to LSTM and CCNN-LSTM, LBL2$_{SWordSS}$'s lower performance on test data was expected as the former are more non-linearly complex language models.

However, for tasks like spoken term detection, having low perplexities on the most frequent words is not good enough, and hence, my coauthor and I compared LMs on the perplexity of a rare-word based test set. To perform this comparison, my coauthor computed perplexity only on rare words (RW1PPL), i.e. with training set frequency of one, present in the test set. Table 5.6 shows that LBL2$_{SWordSS}$ performed better than the LSTM-based LMs across various languages in terms of RW1PPL.

The CCNN-LSTM model could not include SWordSS embeddings easily. Hence, they were not directly comparable to LBL2$_{SWordSS}$, as the latter had more information at its disposal.

OOV and Rare Word Perplexity Results To further compare the performance of the aforementioned language models on rare words, my coauthor and I analyzed perplexities of such words (RWPPL) in the test set as a variation of the frequency classes of these words in the training set. This variation is displayed in Figure 5.1.

For OOVs (rare words with zero training set frequency), LBL2$_{SWordSS}$ outperformed the other language models built with similar number of parameters, on the Tagalog and Turkish corpora. In these cases, LBL2$_{SWordSS}$ reduced rare-word perplexities by a factor of two over the character-feature rich CCNN-LSTM, which implicitly handles rare words by design.

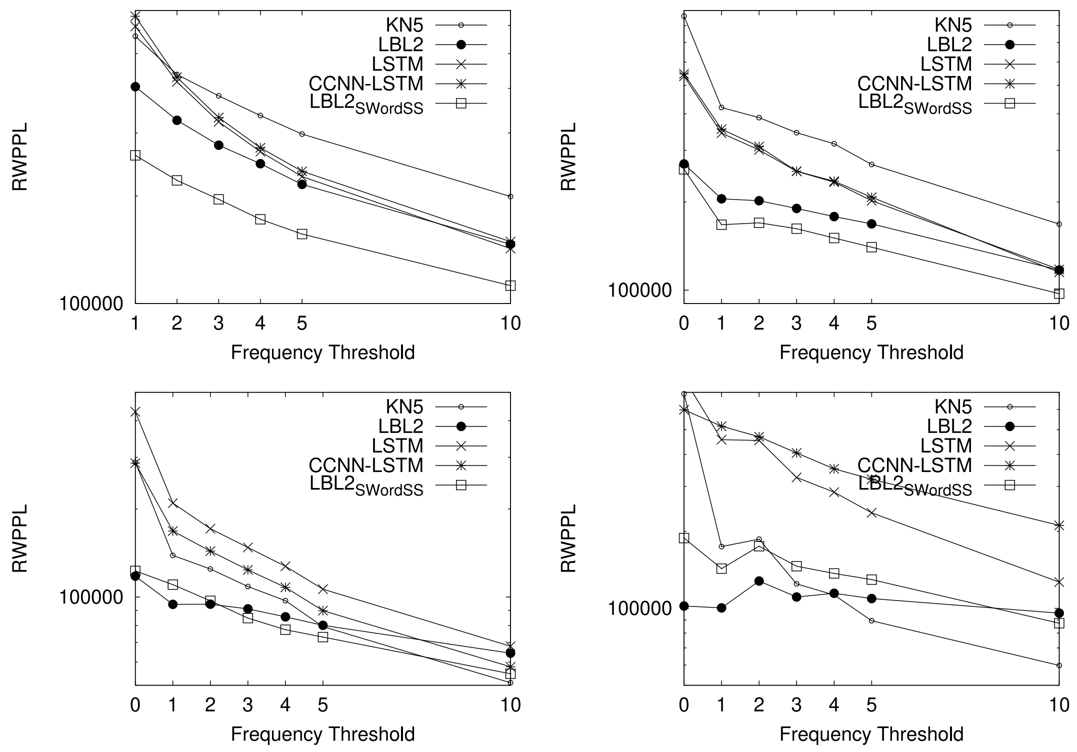


Figure 5.1: Variation of rare-word perplexity versus threshold on frequency of training-set words on German, Tagalog, Turkish and Vietnamese corpora

Even for rare words with training set frequency up to one, LBL2$_{SWordSS}$ reduced perplexity up to a factor of 2.5 times with respect to CCNN-LSTM, on the German, Tagalog and Turkish corpora. Interestingly, on these particular language corpora, Figure 5.1 shows that LBL also performed better than both the LSTM-based LMs in modeling OOV and rare words of frequency up to ten.

For Vietnamese, LBL alone was able to improve OOV and RW1 words over the other LMs. Adding SWordSS embeddings harmed the prediction of OOV and RW1 words. There could have been interference from improper representation of tones.

These perplexity improvements started to wane when higher frequency words were included into the rare word set, across the different languages. Nevertheless, for languages with rich morphology, initializing LBL with SWordSS embeddings reduced perplexities on rare words.

5.5 Long tail analysis

Word embeddings use large amounts of training data to capture regularities that make their inclusion in NLP systems worthwhile. Finding useful representations for rare words is a big problem and my coauthor and I showed that SWordSS can help, but other words also may not be well-served by standard embedding approaches. In this follow-up study, I ask and answer two questions: 1) What is the frequency distribution for words in common word similarity tasks, i.e. how long is the tail? and 2) When does the orthographically aware tool SWordSS (Singh et al., 2016) produce better embeddings than `word2vec` (Mikolov et al., 2013)?

5.5.1 What's in the tail?

As described, the SWordSS method *estimates* an embedding using the non-impoverished embeddings of the word's non-rare orthographic neighbors. Since words in the head of the distribution have enough data, I hypothesize that for these words, information on the neighbors is less useful than information on the exact target. But for words in the long tail, the neighbors are the key to a high quality meaning representation.

Therefore, in the first set of experiments, I investigate a reasonable, task-specific boundary between the head and the tail, i.e. when does a word count as rare? To begin, consider the vocabulary distributions in Figure 5.2.

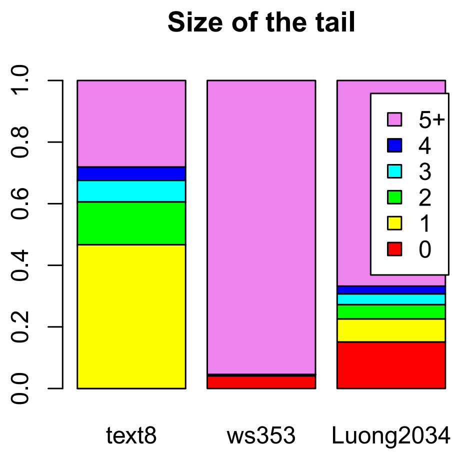


Figure 5.2: Corpus and word similarity test set vocabularies broken down by number of occurrences in the text8 corpus.

The distribution on the left is derived from the text8 corpus, which is the first 100MB of an English Wikipedia dump (Mahoney, 2011). I reproduce the common trend that roughly half of the words in this corpus occur only once. And for this corpus, only a quarter of the words in the vocabulary occur five times or more. The middle distribution are the words from the ws353 word similarity task (Finkelstein et al., 2002). Note that almost all of the words in the dataset are all relatively frequent, which is rather unnatural. The Luong, Socher, and Manning (2013) word similarity dataset in the righthand distribution was designed to have rare words and thus much more closely captures the "natural" distribution on the left.

5.5.2 When does SWordSS help?

By default, `word2vec` does not generate embeddings for words occurring fewer than five times. But, I varied this minimum and trained standard `word2vec` embeddings on the text8 corpus. The resulting correlations with human judgements on the Luong, Socher, and Manning (2013) dataset are shown in the blue curve in Figure 5.3.

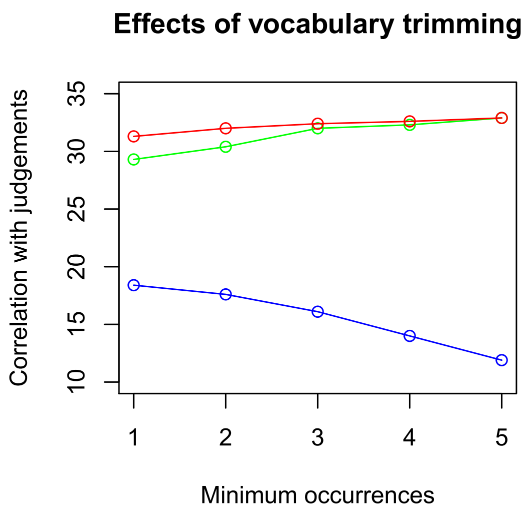


Figure 5.3: Spearman $\rho \times 100$ on the Luong, Socher, and Manning (2013) similarity task versus the minimum occurrences included during `word2vec` training.

The blue curve in Figure 5.3 was the resultant correlations with human judgements while varying only the minimum number of occurrences in the corpus for inclusion in the `word2vec` model. No SWordSS embeddings were used, so words without `word2vec` embeddings could not be scored. So, the correlation decreased as more and more pairs could not be compared against the human judgements. The green curve in Figure 5.3 used the same five `word2vec` models as the blue curve, but all words below the varied minimum occurrence threshold received SWordSS embeddings. This marked improvement showed a clear benefit of using SWordSS for rare words. The red curve in Figure 5.3 also used the same five `word2vec` models from the blue curve. However, this time, all words occurring fewer than five times received SWordSS embeddings, even if they were above the threshold of the `word2vec` model for that configuration. From the nominally better performance in the red curve over the green curve, I show that `word2vec` embeddings can be harmful for rare words. That is, just because the toolkit provided an embedding does not mean that the embedding is useful.

5.6 Conclusion

In this chapter, I introduced SWordSS, a novel sub-word similarity based search for generating rare word embeddings. It leverages the sub-word similarity to search for close matches of a rare word, and then combines their embeddings to make the rare word's embedding.

Even though SWordSS is an unsupervised approach like Soricut and Och (2015), it differs from the latter in the way it utilizes morphological information. Soricut and Och (2015) automatically induces morphological rules and transformations to build a morphological word graph. This graph is then tuned and used to induce embedding of a rare word. By contrast, SWordSS simply indexes by sub-word units of a fixed length. So, it only uses flat orthography rather than any specific morphological structure.

To test the SWordSS technique, I augmented pre-trained embeddings and then evaluated them on word similarity tasks. The augmented embeddings outperformed the initial set of embeddings drastically. However, it lagged behind the state-of-the-art performance of Soricut and Och (2015). But, by employing embeddings trained on larger corpora, SWordSS was able to perform comparably on a rare-word task.

My coauthor also investigated the effects of using SWordSS augmented embeddings for modeling rare words. To perform this experiment, he trained LBL$_{SWordSS}$ LM and compared it with language models like the character aware LM, an LSTM-based RNN LM restricted to similar size. On almost all datasets, the character aware LM outperformed the other LMs with respect to perplexity on complete test sets. But on rare words, SWordSS showed up to 50% reduced perplexity values in comparison to other LMs. Hence, SWordSS embeddings contributed substantially in modeling rare-word tasks.

My follow-up study found that for words occurring fewer than five times, i.e. words in the long tail, a SWordSS embedding, which combines embeddings of the orthographically-similar words, leads to a better meaning representation than a plain `word2vec` embedding.

The publication and follow-up study received substantial interest when I presented them, especially when I advertized that the technique requires no additional data, knowledge, or training. I discovered one important severe limitation in that Vietnamese is a tonal language and the tones may not have been encoded properly for these experiments. So, the tones may have contributed to why the Vietnamese data did not look like the data from other languages.

Fasttext (Bojanowski et al., 2017) came about around the same time as SWordSS. I attribute its greater success than SWordSS at least partially to the better availability of code. I have not explicitly compared SWordSS, fasttext, and more modern approaches that train directly on sub-word information. However, I concede that training directly was the correct technique to persist in the field because SWordSS could bottleneck on its own objective. But the point is that both humans and language models are good at leveraging sub-word information, albeit in different ways.

Chapter 6

Some linguistic patterns that language models failed to capture

So far in this thesis, I have demonstrated many kinds of knowledge that language models are good at encoding, especially long distance dependencies and sub-word information. To finish this first part, I present a striking case study in which seemingly simple features were *not* captured, so rules based on the features outperformed the language model on a text classification task. While my coauthors in Wiegand et al. (2018) developed and tested these features without me, a thorough discussion of them is quite relevant for the thesis because this may be the kind of information that language models could continue to miss.

6.1 Introduction

`thelawdictionary.org` defines abusive or offensive language as "hurtful, derogatory or obscene utterances made by one person to another person". Closely related terms include *hate speech* (Waseem and Hovy, 2016) or *cyber bullying* (Zhong et al., 2016). Several research efforts just focus on utterances addressed towards minorities, but various specifications are all compatible with that general definition for abusive language. Consider (1), (2), and (3)[1].

(1) stop editing this, you **dumbass**.

(2) Just want to slap the **stupid** out of these **bimbos**!!!

(3) Go lick a pig you arab muslim piece of **scum**.

Due to the rise of user-generated web content, in particular on social media networks, the amount of abusive language is also steadily growing. NLP methods are required to focus human review efforts towards the most relevant microposts.

[1]The examples in this work are included to illustrate the severity of abusive language. They are taken from actual web data and in no way reflect the opinion of the authors.

This work addresses the task of detecting abusive words (e.g. *dumbass, bimbo, scum*). The main assumption is that abusive words form a subset of negative polar expressions. The classification task is to **filter the abusive words from a given set of negative polar expressions**. Using a base lexicon that is a small subset of negative polar expressions where the abusive words among them have been marked via crowdsourcing (Section 6.3), my coauthors calibrated a supervised classifier by examining various novel features (Section 6.4). A classifier trained on that base lexicon, which contains 551 abusive words, was then applied to a very large list of unlabeled negative polar expressions (from Wiktionary[2]) to extract an expanded lexicon of 2989 abusive words (Section 6.5).

The new lexicon was evaluated on the novel task of **cross-domain classification** of abusive documents (Section 6.6) where it was used as a *high-level* feature. In this work, microposts were considered documents. While for in-domain classification, supervised classifiers trained on generic features, such as bag of words or word embeddings, usually score very well, on cross-domain classification they perform poorly since they latch on to domain-specific information. In subjectivity, polarity and emotion classification, high-level features based on predictive domain-independent word lists have been proposed to bridge the domain mismatch (Dias, Lambov, and Noncheva, 2009; Mohammad, 2012; Wiegand, Klenner, and Klakow, 2013).

New abusive words constantly enter natural language. For example, according to Wiktionary[2] the word *gimboid*, which refers to an incompetent person, was coined in the British television series *Red Dwarf*, possibly from the word *gimp* and the suffix *-oid*. According to Urban Dictionary[3], the word *twunt*, which is a portmanteau of the swearwords *twat* and *cunt*, has been invented by humourist Chris Morris for the Channel 4 series 'Jam' in 2000. One of the most recent abusive words is *remoaner* which describes someone who complains about or rejects the outcome of the 2016 EU referendum on the United Kingdom's membership of the European Union. It is a blend of *moan* and *remainer*. Wiktionary states that this word has a pejorative connotation.

These examples show that the task of creating a lexicon of abusive words cannot be reduced to a one-time manual annotation effort. Recent web corpora and crowdsourced dictionaries (e.g. Wiktionary) should be ideal resources to find evidence of such words.

My coauthors' **contributions** were presenting the first work that systematically describes the *automatic* construction of a lexicon of abusive words and examining novel features derived from various textual resources. I trained a language model on a large dataset with labeled microposts, and my coauthors and I showed that the information derived from the novel features cannot be equally derived from the language model. The effectiveness of the expanded lexicon is demonstrated on cross-domain detection of abusive microposts. This was also the first work to address this task in general. The supplementary material to the published paper (Wiegand et al., 2018)[4] includes all resources newly created from the research.

[2]`https://en.wiktionary.org`

[3]`https://www.urbandictionary.com`

[4]`https://github.com/uds-lsv/lexicon-of-abusive-words`

The task is framed as a **binary classification problem**. Each given expression is to be classified as either abusive or not. While my coauthors and I used English data, many of the novel features should also be applicable to other languages.

6.2 Related Work

Lexical knowledge for the detection of abusive language has only received little attention in previous work. Most approaches consider it as one feature among many. Very often existing word lists from the web are employed (Xiang et al., 2012; Burnap and Williams, 2015; Nobata et al., 2016). Their limited effectiveness may be due to the fact that they were not built for the task of abusive language detection. Only the manually-compiled lexicon from Razavi et al. (2010) and the lexicon of *hate verbs* from Gitari et al. (2015) have been compiled for this specific task. Since the latter lexicon is not publicly available, only the former was considered in this work. In both publications, very little is said on the creation of these resources.

Previous work focused on in-domain classification, a setting where generic features (e.g. bag of words) work well and word lists are less important. There have been investigations examining features on various datasets (Nobata et al., 2016; Safi Samghabadi et al., 2017). However, these studies always trained and tested on the *same* domain. This work shows that a lexicon-based approach is effective in cross-domain classification.

Schmidt and Wiegand (2017) gives a more detailed overview on previous work on the detection of abusive language in general.

6.3 Data

Base Lexicon The base lexicon exclusively comprises negative polar expressions. It is a small set annotated via crowdsourcing. Abusive words were considered to be a proper subset of negative polar expressions. By just focusing on these types of words, my coauthors were more likely to obtain a significant amount of abusive words than just considering a sample of arbitrary words. This lexicon will be used as a gold standard for calibrating features of a classifier. That classifier will be run on a large set of unlabeled negative polar expressions to produce the expanded lexicon (Section 6.5).

My coauthors sampled 500 negative nouns, verbs and adjectives each from the Subjectivity Lexicon (Wilson, Wiebe, and Hoffmann, 2005). This lexicon was chosen since my coauthors had extra information available for its entries that were of interest, namely polar intensity (Section 6.4.1) and sentiment views (Section 6.4.2). However, since the Subjectivity Lexicon misses some prototypical abusive words (e.g. *nigger, slut, cunt*), my coauthors added another 10% (i.e. 150 words) which are abusive words frequently occurring in the word lists mentioned in Schmidt and Wiegand (2017).

Each of the negative polar expressions was judged by 5 annotators from the crowdsourcing platform ProlificAcademic.[5] Each annotator had to be a native speaker of English and possess a task approval rate of at least 90%. For the base lexicon (Table 6.1), the task was a binary word categorization: *abusive* or *non-abusive*. A word was only classified *abusive* if at least 4 out of the 5 raters judged the word to be abusive. This threshold should prevent many ambiguous words from being classified as abusive, a general problem of existing resources (Davidson et al., 2017).

	adjective		**noun**		**verb**		**all**	
class	freq	%	freq	%	freq	%	freq	%
abusive	170	33.8	291	45.3	90	17.8	551	33.4
not abusive	332	66.2	352	54.7	415	82.2	1099	66.6

Table 6.1: The base lexicon: 1650 entries in total of which 551 are abusive.

Corpora The experiments employed three unlabeled corpora (Table 6.2). The two larger corpora, the *Amazon Review Corpus* – AMZ (Jindal and Liu, 2008) and the *Web As Corpus* – WAC (Baroni et al., 2009), are used for inducing word embeddings (Section 6.4.7). AMZ and the smallest corpus, *rateitall.com* – RIA[6], are used for computing polar word intensity (Section 6.4.1) from star ratings.

corpus	**tokens**	**properties**	**good for computation of**	
RIA	4.7M	reviews focused on persons	polar intensity	(Section 6.4.1)
AMZ	1.2B	product review corpus	polar intensity word embeddings	(Section 6.4.1) / (Section 6.4.7)
WAC	2.3B	large general web corpus	word embeddings	(Section 6.4.7)

Table 6.2: Information about unlabeled corpora used.

6.4 Feature Calibration

In this section, I describe the novel linguistic features and also word embeddings derived from a language model that failed to capture them. My coauthors also developed some competitive baselines for the base lexicon. The classifier was support vector machines as implemented in SVM^{light} (Joachims, 1999). This classifier was chosen since it is most commonly used for the detection of abusive language (Schmidt and Wiegand, 2017). For all classifiers in this paper, the supplementary material[4] contains information regarding (hyper)parameter settings.

[5]The supplementary material[4] contains more information regarding the annotation set-up (including guidelines).

[6]This is a crawl from the review website `https://www.rateitall.com`.

6.4.1 Polar Intensity (INT)

Intuitively, abusive language should coincide with high polar intensity. My coauthors inspected 3 different types.

Binary Intensity (INT$_{\textbf{bin}}$) The first feature is a simple binary intensity feature obtained from the Subjectivity Lexicon. In that resource, each entry is categorized as either a *weak* polar expression (e.g. *dirty*) or a *strong* polar expression (e.g. *filthy*). Table 6.3 (left half), which shows the distribution of intensity on the intersection of the base lexicon and the Subjectivity Lexicon, confirms that abusive words are rarely weak polar expressions and more frequently strong polar expressions.

		intensity (Section 6.4.1)		**views** (Section 6.4.2)	
class	**all**	weak	strong	actor	speaker
abusive	26.7	14.1	32.0	9.7	32.8
not abusive	73.3	85.9	68.0	90.3	67.2

Table 6.3: Percentage of abusive / not abusive instances among (binary) intensity and views. All numbers only refer to the subset of the base lexicon (Table 6.1) taken from the Subjectivity Lexicon (i.e. 1500 entries).

Fine-grained Intensity (INT$_{\textbf{fine}}$) My coauthors also investigated a more fine-grained feature which assigns a real-valued intensity score to polar expressions. It is computed by leveraging the star-rating assigned to the reviews comprising the AMZ corpus (Table 6.2), a large publicly available review corpus. A review is awarded between 1 and 5 stars where 1 is the most negative score. The system infers the polar intensity of a word by the distribution of star-ratings associated with the reviews in which it occurs. Negative polar expressions with a very high polar intensity were assumed to occur more often in reviews assigned few stars (i.e. 1 or 2). Ruppenhofer, Wiegand, and Brandes (2014) established that the most effective method to derive such polar intensity is by ranking words by their *weighted mean of star ratings* (Rill et al., 2012). All words of the base lexicon were ranked according to that score, and then the ranks were used as a feature.

Intensity Directed towards Persons (INT$_{\textbf{person}}$) Not all negative polar expressions with a high intensity are equally likely to be abusive. The high intensity expressions should also be words typically directed towards persons. Most polar statements in AMZ, however, are directed towards a movie, book or some electronic product. In order to extract negative polar intensity directed towards persons, my coauthors used the RIA corpus (Table 6.2). RIA contains reviews on arbitrary entities rather than just commercial products as in the case of AMZ. Each review has a category label (e.g. *computer*, *person*, *travel*) that very easily allows us to extract from RIA just those reviews that concern persons.

Table 6.4 compares a typical 1-star review from AMZ with one from RIA. The RIA review was considered an abusive comment. It contains many words predictive of abusive language (e.g. *self-absorbed*, *loser*, *arrogant*, and *loud-mouthed*).

AMZ	*on Halloween 5*: this movie is horrible with a bad plot a disappointment to the halloween series.
RIA	*on Bill Maher*: Self-absorbed loser who tries to pretend to be fair. He is rude, arrogant, loud-mouthed…

Table 6.4: 1-star reviews in different corpora.

6.4.2 Sentiment Views (VIEW)

Wiegand, Schulder, and Ruppenhofer (2016) define sentiment views as the perspective of the opinion holder of polar expressions. They distinguish between expressions conveying the view of the implicit speaker of the utterance typically referred to as *speaker views* (e.g. *cheating* in (4); *ugly* and *stinks* in (5)), and expressions conveying the view of event participants typically referred to as *actor views* (e.g. *disappointed* and *horrified* in (6); protested in (7)). Wiegand, Schulder, and Ruppenhofer (2016) provided sentiment-view annotations for the entries of the Subjectivity Lexicon.

(4) Peter is always **cheating**$_{\textit{speaker view}}$. (*holder: speaker*)

(5) Mary is an **ugly**$_{\textit{speaker view}}$ girl that **stinks**$_{\textit{speaker view}}$. (*holder: speaker*)

(6) [Peter]$_{\textit{holder}}$ was **disappointed**$_{\textit{actor view}}$ and **horrified**$_{\textit{actor view}}$ at the same time.

(7) [The public]$_{\textit{holder}}$ **protested**$_{\textit{actor view}}$ against that law.

Sentiment views have been used for improving the extraction of opinion holders and targets (Deng and Wiebe, 2016; Wiegand, Bocionek, and Ruppenhofer, 2016). This work shows that they also have relevance for the detection of abusive words. Among actor-view words, there is a much lower proportion of abusive words than among speaker-view words (right half of Table 6.3). This can be explained by the fact that verbal abuse usually originates from the speaker of an utterance rather than some other discourse entity. Sentiment-view information was used as a binary feature.

6.4.3 Emotion Categories (NRC)

My coauthors also examined whether knowledge of emotion categories associated with words is helpful. Potentially negative emotions, such as disgust or anger, should correlate with abusive words. My coauthors used the NRC lexicon (Mohammad and Turney, 2013) and employed the categories associated with the words contained in that resource as a feature.

6.4.4 Patterns (PAT)

Noun Pattern (PAT_{noun}) My coauthors identified that the noun pattern (8) can be used to extract abusive nouns. Since this pattern is very sparse even on the largest corpus (i.e. WAC), my coauthors also ran the pattern as a query on Twitter and extracted all matching tweets coming in a period of 14 days. (This appeared to be a saturation point.)

(8) a. *pattern*: called {me|him|her} a(n) <noun>

b. *pattern match example*: He called me a **bitch**.

Table 6.5 compares the most frequent matches for that pattern. The pattern matched much more frequently on Twitter than on WAC. The quality of the matches on Twitter was also much better than on WAC, where there were still many false positives (e.g. *name* or *saint*). My coauthors asserted that Tweets, in general, are much more negative in tone than arbitrary web documents (as represented by WAC) which could explain the fewer false positives on Twitter. Note that the ranking from Twitter is not restricted to just prototypical abusive words (as Table 6.5 might suggest). The entire ranking also contains many less common words, such as *weaboo*, *dudebro* or *butterface*. The frequency ranks of the nouns extracted from Twitter are used as a feature.

WAC	liar (19), coward (7), name (6), idiot (6), hero (5), horse (5), saint (5), fool (5), snob (4), genius (4)
Twitter	bitch (1534), hoe (432), liar (317), cunt (274), whore (254), pussy (228), nigger (226), loser (217), faggot (217), slut (197)

Table 6.5: Comparison of the 10 most frequent pattern matches (*numbers in brackets indicate frequency*).

Adjective Pattern (PAT_{adj}) Abusive adjectives often modify an abusive noun as in *brainless idiot*, *smarmy liar* or *gormless twat*. Therefore, my coauthors mined Twitter for adjectives modifying mentions of the extracted nouns (PAT_{noun}). (My coauthors did not find a construction identifying abusive verbs, so the output from PAT includes no verbs.)

6.4.5 WordNet (WN) and Wiktionary (WK)

My coauthors also used WordNet (Miller et al., 1990) and Wiktionary[2], two general-purpose lexical resources. Unlike WordNet, Wiktionary is produced collaboratively by volunteers rather than linguistic experts. It contains more abusive words from the base lexicon, i.e. 97% (WK) vs. 87% (WN).

A common way to harness a general-purpose lexicon for induction tasks in sentiment analysis is by using its **glosses** (Choi and Wiebe, 2014; Kang et al., 2014). Assuming that the explanatory texts of glosses are similar among abusive words, glosses were treated as a bag-of-words feature.

My coauthors also exploited information on **word usage**. Many abusive words are marked with tags such as *pejorative*, *derogatory*, or *vulgar*. Both WordNet and Wiktionary contain such information. However, in Wiktionary more than 6 times as many of the entries include a tag compared to WordNet.

In order to incorporate a semantic representation more general than individual words, my coauthors employed **supersenses**. Supersenses are only contained in WordNet. They represent a set of 45 classes into which entries are categorized. They have been found effective for sentiment analysis (Flekova and Gurevych, 2016). Some categories correlate with abusive words. For example, 76% of the words of the base lexicon that belong to the supersense *person* (e.g. *loser*, *idiot*) are abusive words.

6.4.6 FrameNet (FN)

FrameNet (Baker, Fillmore, and Lowe, 1998) is a semantic resource which provides over 1200 semantic frames that comprise words with similar semantic behavior. My coauthors used the frame-memberships of a word as features, expecting that abusive and non-abusive words occur in separate frames.

6.4.7 Generic Features: Word Embeddings

I induced word embeddings from the two largest corpora, i.e. AMZ and WAC (Table 6.2) using `word2vec` (Mikolov et al., 2013) in default configuration (i.e. 200 dimensions; cbow). The best performance was obtained by concatenating for each word the vectors induced from the two corpora.[7]

6.4.8 Baselines to Feature-based Approach

In addition to a majority-class classifier, my coauthors considered the following baselines:

Weak Supervision (WSUP) This baseline was a lightweight classifier that does not require proper labeled training data. It was inspired by previous induction approaches for sentiment lexica, such as Hatzivassiloglou and McKeown (1997) or Velikovich et al. (2010), which heuristically label some seed instances and then apply graph-based propagation to label the remaining words of a dataset. On the basis of word embeddings (Section 6.4.7), my

[7] I also ran experiments with pre-trained embeddings from *GoogleNews* but they did not improve classification.

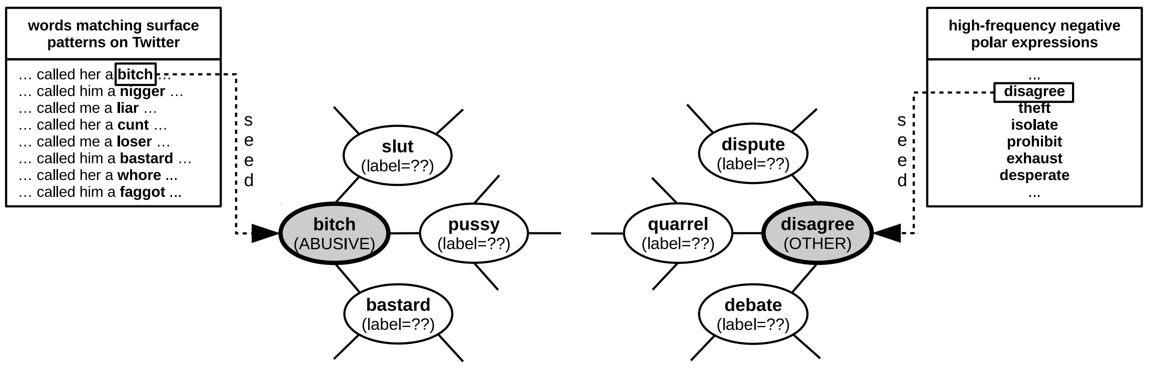

words matching surface patterns on Twitter
... called her a bitch ...
... called him a nigger ...
... called me a liar ...
... called her a cunt ...
... called me a loser ...
... called him a bastard ...
... called her a whore ...
... called him a faggot ...
seed
slut (label=??)
bitch (ABUSIVE)
pussy (label=??)
bastard (label=??)
dispute (label=??)
quarrel (label=??)
disagree (OTHER)
debate (label=??)
seed
high-frequency negative polar expressions
...
disagree
theft
isolate
prohibit
exhaust
desperate
...

Figure 6.1: Illustration of word-similarity graph as used for weakly-supervised baseline (WSUP); seeds for abusive words (e.g. *bitch*) are obtained by the output of feature PAT (Section 6.4.4); seeds for non-abusive words (e.g. *disagree*) are high-frequency negative polar expressions.

coauthors built a word-similarity graph, where the nodes represent negative polar expressions and each edge denotes the cosine similarity between the embeddings of two arbitrary words. The output of PAT from Twitter (Section 6.4.4) is considered as positive class seed instances. PAT was chosen since it is an effective feature that does not depend on a lexical resource. The most frequent words in the WAC corpus (Table 6.2) were chosen as negative class seeds. The rationale was that high-frequency words are unlikely to be abusive. My coauthors chose WAC instead of Twitter since the evidence of PAT (Table 6.5) suggested less abusive language in that corpus. This word-similarity graph is illustrated in Figure 6.1. In order to propagate the labels to the unlabeled words from the seeds, my coauthors used the Adsorption algorithm (Talukdar et al., 2008).

Using Labeled Microposts (MICR) The last baseline examined the power of labeled microposts to detect individual abusive words. These experiments were driven by the fact that labeled microposts already exist. My coauthors considered two methods using the largest dataset comprising manually labeled microposts, *Wulczyn* (Table 6.6). The class labels of the microposts and the base lexicon (Section 6.3) are the same. The aim is to produce a ranking of words where the high ranks represent words more likely to be abusive. Following standard machine learning procedure, my coauthors set a cut-off rank as a decision boundary (*see supplementary material*[4]). Every word higher than this rank is considered abusive and all other words not abusive.

The first method **MICR:pmi** ranks the words of the base lexicon by their Pointwise Mutual Information with the class label *abusive* that is assigned to microposts. To be even more competitive, I assisted in introducing a second method **MICR:proj** that learns a projection of embeddings. MICR:proj has the advantage over MICR:pmi that it does not only rank words observed in the labeled microposts but all words represented by embeddings. Since the embeddings (Section 6.4.7) are induced on the combination of AMZ and WAC corpora,

dataset	microposts	abusive	source
Warner and Hirschberg (2012)	3438	14.3%	*diverse*
Waseem et al. (2017)	16165	35.3%	Twitter
Razavi et al. (2010)	1525	31.9%	UseNet
Wulczyn, Thain, and Dixon (2017)	115643	11.6%	Wikipedia

Table 6.6: Datasets comprising labeled microposts.

which together are about 360 times the size of *Wulczyn*, MICR:proj is likely to cover more abusive words. Let $M = [\vec{w_1}, \ldots, \vec{w_n}]$ denote a labeled micropost of n one-hot representations of words. The aim is learning a one-dimensional projection SE where $E \in \mathbb{R}^{e \times v}$ represents the unsupervised embeddings of dimensionality e over the vocabulary size v (Section 6.4.7) and $S \in \mathbb{R}^{1 \times e}$ represents the learned projection matrix. A projected micropost is computed as $\vec{h} = (SEM)^t$ which is an n-dimensional vector. Each component represents a word from the micropost. Each value represents the predictability of the word towards being abusive. A bag-of-words assumption is then applied to use that projected micropost to predict the binary class label y: $p(y|M) \propto \exp(\sum_{i=1}^{n} h_i)$. This model is a feed-forward network trained using Stochastic Gradient Descent (Rumelhart, Hinton, and Williams, 1986). The negative polar expressions were ranked on the basis of the projected embeddings.

6.4.9 Evaluation of Features on Base Lexicon

My coauthors conducted experiments on the base lexicon (Table 6.1). I report macro-average precision, recall and f-score. SVMs were evaluated on a 10-fold cross-validation. Table 6.7 displays the performance of the different classifiers. The least effective information source was labeled microposts (MICR), though, as expected, the projected embeddings (MICR:proj) outperformed PMI. Weak supervision (WSUP) outperformed MICR.

Among the SVM configurations, embeddings were already effective. The linguistic features outperformed all other methods. The best classifier was an SVM trained on embeddings, linguistic features and the output of WSUP as a further feature.[8] Table 6.8 shows the performance of SVMs using different linguistic features (Section 6.4.1-Section 6.4.6). Among the three intensity types, the most effective one was the person-based intensity ($\text{INT}_{\text{person}}$). Among the lexical sentiment resources used (i.e. NRC, INT_{bin}, and VIEW), VIEW was most effective. Their combination also resulted in an improvement. The surface patterns (PAT) were surprisingly predictive. Of the general-purpose lexical resources (i.e. WN, WK, and FN), WN and WK were both very effective resources. Glosses from WN were the strongest individual feature. Combining WK, WN and FN resulted in significant improvement. The best feature set combined all features.

[8]MICR was not included among the further features because they were trained on data that overlapped with the test data for the extrinsic evaluation (Section 6.6).

classifier	Precision	Recall	F1
MAJORITY	33.3	50.0	40.0
MICR:pmi	65.3	59.5	62.2†
MICR:proj	67.1	64.6	65.8∗†
WSUP	77.3	71.0	74.0∗†
SVM:embeddings	77.6	73.9	75.7∗
SVM:linguistic	81.6	73.8	77.5∗
SVM:linguistic+WSUP	82.5	76.5	79.4∗†
SVM:linguistic+embeddings	81.6	79.7	80.7∗
SVM:linguistic+embed.+WSUP	**82.9**	**80.4**	**81.6**†

Table 6.7: Different classifiers on base lexicon (Table 6.1). Statistical significance testing (paired t-test at $p < 0.05$):
$*$: better than two lines above; †: better than previous line.

The results also suggest that for languages other than English, there are some very strong features, such as PAT, WK, or embeddings, that could be easily adopted since they do not depend on a resource which is only available in English.

6.5 Expanding the Lexicon

My coauthors produced a large **feature-based lexicon** of abusive words by classifying all (unlabeled) negative polar expressions from Wiktionary[2]. Wiktionary was chosen since previous experiments indicated a high coverage of abusive words on that resource (Section 6.4.5). The negative polar expressions were identified by applying to the vocabulary of Wiktionary an SVM trained on the words from the Subjectivity Lexicon with their respective polarities. My coauthors used my word embeddings as features (Section 6.4.7). In order to produce the feature-based lexicon of abusive words, my coauthors trained another SVM on the base lexicon (Table 6.1) using the best feature set from Table 6.7. With 2989 abusive words, the expanded lexicon is 5 times as large as the base lexicon.

To measure the impact of the engineered features on the quality of the resulting lexicon, my coauthors devised an alternative expansion which just employed word embeddings. For this, they used *SentProp*, the most effective induction method from the `SocialSent` package (Hamilton et al., 2016).[9]

[9]Since SentProp produces a ranking rather than a classification, they considered 2989 as a cut-off value to separate the instances into 2 classes. This corresponds to the size of abusive words predicted by the feature-based lexicon (Table 6.9).

features used in SVM	Precision	Recall	F1
MAJORITY	33.3	50.0	40.0
INT_{fine}	62.0	57.0	59.4†
INT_{bin}	61.7	60.4	61.0∗
$\text{INT}_{\text{person}}$	70.8	55.4	62.1∗
INT_{fine}+INT_{bin} +$\text{INT}_{\text{person}}$	70.8	60.7	65.3∗†
NRC	60.2	60.1	60.2
VIEW	65.6	62.8	64.2†
INT_{bin}+NRC+VIEW	66.9	68.8	67.9∗†
PAT_{noun}	79.9	58.4	67.4
PAT_{noun}+PAT_{adj}	76.4	63.2	69.1
WN_{usage}	82.6	52.6	64.3
FN	66.3	66.4	66.4
WK_{usage}	76.7	61.0	67.9∗†
WK_{gloss}	74.8	64.9	69.5∗†
WN_{super}	78.7	64.9	71.1∗†
WN_{gloss}	75.9	67.4	71.4∗
WN_{usage}+WN_{super}+WN_{gloss}	76.7	68.0	72.0∗
WK_{usage}+WK_{gloss}	79.5	67.0	72.7∗
all WN + *all* WK	80.0	68.7	73.9∗
all WN + *all* WK + FN	80.3	69.5	74.5∗
all from above	81.6	73.8	77.5∗†

Table 6.8: Performance of the different linguistic features on base lexicon (Table 6.1). Statistical significance testing (paired t-test at $p < 0.05$):
∗: better than two lines above; †: better than previous line.

baseline lexicon	entries	newly created lexicon		entries
Hatebase	430	base	(Table 6.1)	551
Derogatory	1609	expanded:SentProp	(Section 6.5)	2989
Ottawa	1746	expanded:feature-based	(Section 6.5)	2989

Table 6.9: Lexica used in cross-domain classification of microposts
(*figures denote the number of unigrams*).

6.6 Cross-domain Classification

6.6.1 Motivation and Set Up

My coauthors then used the expanded lexicon (Section 6.5) to classify entire microposts rather than words out of context. Table 6.6 shows the datasets of labeled microposts that were used. The difference between these datasets is the source from which they originate. Consequently, different topics are represented in the different datasets. Still, the datasets contained similar types of abusive language (e.g. racism and sexism). For example, both (9) and (10) from *Waseem* and (11) from *Wulczyn* are sexist comments[10] but (9) and (10) discuss the role of women in sports while (11) addresses women's hygiene in Slavic countries.

(9) *from Waseem*:
maybe that's where they should focus? Less **cunts** on football.

(10) *from Waseem*:
I would rather brush my teeth with sandpaper then watch *football* with a girl!!

(11) *from Wulczyn*:
slavic women don't like to wash…Their **pussy** stinks.

Since the aim is to produce the best possible cross-domain classifier, **all classifiers are trained on one dataset and tested on another**. This is a real-life scenario. Often when a classifier for abusive microposts is needed, sufficient labeled data is only available for other text domains.

Having different topics in training and test data makes cross-domain classification difficult. For example, since a large proportion of sexist comments in *Waseem* relate to sports, traditional supervised classifiers (using bag of words or word embeddings) will learn correlations between words of that domain with the class labels. For instance, the domain-specific word *football* occurs frequently in *Waseem* (i.e. 90 occurrences) with a strong correlation towards abusive language (precision: 95%). Other words, such as *sports* and *commentator*, display a similar behavior. A supervised classifier will assign a high weight to such words. While such domain-specific words may aid in-domain classification and enable a correct classification of microposts, such as (10), it had a detrimental effect on cross-domain classification. My coauthors claimed that the predictive words that abusive comments share across different domains are abusive words, just of the sort that the expanded lexicon contains, e.g. *cunts* in (9) and *pussy* in (11).

The classifier for labeling microposts was an SVM trained on features derived from the expanded lexicon (Section 6.5). There was no binary feature encoding the presence of abusive words. Instead, all abusive words of the lexicon were ranked according the classifier that the lexicon induced, and then the ranks were used as features.

As **baseline classifiers** my coauthors considered publicly available word lists (Table 6.9). The resource from Razavi et al. (2010), henceforth referred to as ***Ottawa***, the entries of ***Hatebase***[11], which has been used in Nobata et al. (2016) and Davidson et al. (2017), and the derogatory words from Wiktionary (***Derogatory***)[12] were included.[13] Finally, the base

[10](11) is also a racist comment.
[11]`https://www.hatebase.org`
[12]`https://en.wiktionary.org/wiki/Category:English_derogatory_terms`
[13]There are also similar but smaller lists in Wiktionary, e.g. *offensive terms*. They produced no better results.

lexicon from this work (Table 6.1) was included to evaluate the expansion process of the two expanded lexica (Section 6.5). For all lists, my coauthors trained on a single feature indicating the frequency of abusive words in a micropost to be classified. *Ottawa* also contains weights assigned to abusive words.

Further, 3 classifiers representing the state of the art of in-domain evaluations were evaluated: ***FastText*** (Joulin et al., 2017), Gated Recurrent Units Recurrent Neural Networks ***RNN***, which have been reported to work best on English microposts (Pavlopoulos, Malakasiotis, and Androutsopoulos, 2017), and ***Yahoo***, an SVM trained on the sophisticated feature set proposed by Nobata et al. (2016). Next to character and token n-grams, *Yahoo* includes word and comment embeddings, syntactic features, and some linguistic diagnostics.

6.6.2 Results

Table 6.10 lists the performance of the 3 state-of-the-art classifiers along with my coauthors' classifier using the expanded lexicon on in-domain 10-fold cross-validation. The point of this list was to demonstrate the strength of the state-of-the-art classifiers on in-domain evaluation. On this setting, a lexicon-based approach is not competitive since domain-specific information is not included. However, shown in Table 6.11, for cross-domain classification, it is exactly that property that ensures that the feature-based lexicon provides best performance. Compared to the in-domain setting, *FastText*, *RNN* and *Yahoo* display a huge drop in performance. They all suffer from overfitting to domain-specific knowledge.

classifier	Razavi	Warner	Waseem	Wulczyn
expanded:feature-based (SVM)	75.7	64.8	63.8	78.4
FastText	**83.4**	71.8	76.3	85.6
RNN	74.8	70.5	78.0	86.9
Yahoo (SVM)	82.4	**78.2**	**84.1**	**90.0**

Table 6.10: In-domain classification of microposts (*metric: F1-score*).

Of all lexica, my coauthors' feature-based lexicon performs best. The poor performance of *Hatebase* was surprising, but attributed to its small size and the high amount of ambiguous (and debatable) entries, such as *Charlie*, *pancake*, and *Pepsi*. Although the feature-based lexicon is the largest of all tested (i.e. 2989 words), the experiments do not support the general rule that larger lexica always outperform smaller ones. For instance, already the base lexicon with 551 abusive words was much better than the lexica *Derogatory* and *Ottawa* which are about 3 times larger (Table 6.9). Each word in the base lexicon was only included if 4 out of 5 raters judged it to be abusive. This ensured a fairly reliable annotation. In contrast, *Derogatory* and *Ottawa* suffer from many ambiguous entries (e.g. *bag*, *Tim*, and *yellow*). The high precision of the base lexicon is what ensures that the expanded lexicon does not include much noise.

					SVM						
datasets						baseline lexica			newly created lexica		
test	**train**	**majority**	**FastText**	**RNN**	**Yahoo**	**Hatebase**	**Derogat.**	**Ottawa**	**base**	**SentProp**	**feature-b.**
Razavi	Warner	40.50	50.59	53.76	53.40	40.50	40.50	60.95	61.08	64.20	**66.13**
	Waseem	40.50	51.64	53.39	51.66	44.29	51.35	63.13	69.69	63.12	**74.15**
	Wulczyn	40.50	71.74	71.59	**75.10**	40.50	40.50	40.50	40.50	68.50	74.83
	Average	40.50	57.99	59.58	60.05	41.76	44.12	54.86	57.09	66.27	**71.70**
Warner	Razavi	46.14	57.73	48.99	55.42	46.14	57.49	59.81	63.57	**67.57**	64.98
	Waseem	46.14	61.45	57.63	56.54	63.52	57.49	**64.67**	63.57	62.75	64.64
	Wulczyn	46.14	58.35	57.36	60.19	46.14	46.14	46.14	46.14	**65.34**	63.35
	Average	46.14	59.18	54.66	57.38	51.93	53.71	56.87	57.76	**65.22**	64.32
Waseem	Razavi	40.62	60.91	54.67	57.83	40.62	52.66	52.95	57.33	**64.56**	63.32
	Warner	40.62	58.28	58.85	**60.65**	40.62	40.62	40.62	54.93	51.98	58.66
	Wulczyn	40.62	56.33	54.13	51.76	40.62	40.62	40.62	40.62	50.27	**62.90**
	Average	40.62	58.51	55.88	56.75	40.62	44.63	44.73	50.96	55.60	**61.63**
Wulczyn	Razavi	46.88	64.65	64.43	70.70	46.88	50.97	57.70	69.56	67.69	**73.71**
	Warner	46.88	56.21	56.13	52.73	46.88	46.88	55.93	59.55	66.38	**70.06**
	Waseem	46.88	52.66	57.33	51.23	43.51	50.97	60.08	69.56	66.38	**72.39**
	Average	46.88	57.84	59.30	58.22	45.76	49.61	57.90	66.22	63.52	**72.05**

Table 6.11: Different classifiers on **cross-domain** classification of microposts; best result in bold (*metric: F1-score*).

Another shortcoming of most of the other existing lexica is that they overwhelmingly focus on nouns. While nouns undoubtedly represent the most frequent abusive terms, there is, however, a substantial number of abusive words that belong to other parts of speech, particularly adjectives (e.g. *vile*, *sneaky*, *slimy*, and *moronic*). In the base lexicon, more than 30% of the abusive words are adjectives. The expanded lexicon, which roughly preserves that ratio, includes about 800 adjectives in total. Since abusive adjectives often co-occur with abusive nouns (Section 6.4.4), they may compensate for abusive nouns that are missing from the lexicon. Such unknown nouns often occur when authors of microposts try to obfuscate their abusive language, e.g. *sneaky assh0le* and *slimy b*st*rd*. Interestingly, the modifying adjectives are not obfuscated, probably because they are considered slightly less offensive in tone.

Given that, among the newly created lexica, the feature-based expanded lexicon performed best, my coauthors concluded that the expansion was effective (since there was an improvement over the base lexicon), and the features were more effective than a generic induction approach (i.e. *SentProp*).

6.6.3 Explicitly vs. Implicitly Abusive Microposts

The results in Table 6.11 also show that the cross-domain performance of my coauthors' feature-based lexicon is lower on the two datasets *Warner* and *Waseem*. My coauthors observed that while for the other two datasets almost all abusive microposts can be considered *explicitly abusive* posts, i.e. they contain abusive words, a large proportion of microposts labeled abusive in *Warner* and *Waseem* are *implicitly abusive* (Waseem et al., 2017), i.e. the abuse is conveyed by other means, such as sarcasm or metaphorical language as in (10). My coauthors asked raters from *ProlificAcademic* to identify explicitly abusive microposts by marking abusive words in those posts. The annotators were not given access to any lexicon of abusive words. They then conducted cross-domain classification on those subsets where the abusive instances were only those rated as explicit. The results are displayed in Table 6.12. The table shows that the feature-based lexicon is much better on this subset, while the most sophisticated supervised classifier (*Yahoo*) still performs worse. From that, my coauthors concluded that only explicitly abusive microposts can be reliably detected in cross-domain classification.

		Yahoo		**feature-based lexicon**	
test	**train**	**all**	**explicit**	**all**	**explicit**
Warner	Razavi	55.4	65.2	65.0	80.6
	Waseem	58.1	55.9	64.6	79.0
	Wulczyn	60.2	72.8	63.4	80.7
	Average	57.9	64.6	64.3	80.1
Waseem	Warner	58.5	61.2	63.3	62.0
	Razavi	61.1	63.1	58.7	78.8
	Wulczyn	51.2	68.2	62.9	78.5
	Average	56.9	64.2	61.6	73.1

Table 6.12: Cross-domain classification of microposts: *all* test data vs. *explicit* subset (*metric: F1-score*).

6.7 Conclusion

For the task of inducing a lexicon of abusive words, my coauthors painstakingly developed novel features including surface patterns, sentiment views, polar intensity, and adapted general purpose lexical resources, particularly Wiktionary. Surprisingly, machine learning over labeled microposts could not capture this information to the same degree, even with a projection-based classifier. So the big question is why. I cannot dismiss the possibility that with a larger, more advanced model trained on more data, some of these patterns could be learned. But the features were, for the most part, local. As discussed in previous chapters, long distance dependencies, sparse data, and polysemy make language modeling harder. These are not the issue here. Looking towards the second part of this thesis, the explanation consistent with my data is that there is common sense knowledge, perceptuomotor knowledge, and other kinds of knowledge at play that are divorced from the kinds of patterns that language models capture. In my work with thematic fit (Greenberg, Sayeed, and Demberg, 2015; Greenberg, Demberg, and Sayeed, 2015a,b; Sayeed, Greenberg, and Demberg, 2016), I argued that humans use implicit knowledge that they do not talk about because it is assumed that everyone already knows it. The key, as explained in Part II, is catching the implicit knowledge within some explicit human behavior.

Part II

Relating language models and human behavior measures

Chapter 7

An experiment on distinguishing human-written and LM-written text

The successes from the novel language model architectures that I worked on led me to consider a grander perspective on our results. Were language models already at or approaching some ceiling of performance? From this starting point, my coauthors and I ran an experiment in which the second halves of sentences were removed, language models were sampled to re-complete the sentences, the originals were mixed in, and humans rated all of the sentences for humanness (Shen et al., 2017). At the time, the original sentences scored higher, but I postulate that now, with wide-scale availability of Transformers, the scores would be much closer, if not indistinguishable. Of course, the task can, and has, been made harder (Dugan et al., 2020), but in later chapters, I use this experiment as a backdrop for the argument that language models should be evaluated against human behavior for their underlying probability distributions rather than text that is sampled from them.

7.1 Introduction

Advances in language modeling, including those described in the first part of this thesis, have led to remarkable performance gains across many language model-dependent tasks. Speech recognition systems have been reported to perform almost as well as humans (Xiong et al., 2016; Saon et al., 2017). In machine translation, purely neural network-based models can already achieve performance comparable to the traditional phrase-based machine translation system *Moses* with a small vocabulary (Meng et al., 2016). Language models were generating coherent, human-like novel sentences at least as far back as Bowman et al. (2016).

Such gains led my coauthors and me in Shen et al. (2017) to consider the possibility that language models were at or approaching some sort of ceiling in terms of a comparison with human performance on various applications of language models. I was instrumental in designing and analyzing an experiment that quantified the position of language models at the state of the art in 2017 relative to this ceiling. In the paper, my coauthors and I used the terms “ideal” and “perfect” to describe a hypothetical language model that could perform tasks at least as well as a human could. As such, my coauthors and I named the project “the language model Turing Test” referring to the “Imitation Game” put forward by Turing (1950). The essence of this game is that a human converses with some entity over a medium that obscures whether that entity is another human or a machine. In some variants, the human converses with one of each and is asked to guess which is which. In other variants, the human converses with just one entity at a time and is asked afterwards if they thought the entity was human. Note that in the original formulation, the outcomes are binary. Either the entity is deemed human or not. Or in the two entity version, the human distinguishes correctly or not. One of my major individual contributions to this work was designing the experiment in such a way that the experiment generates more than binary data. Through this design, my coauthors and I were able to “reverse engineer” a perplexity for the language model at the performance ceiling. Thus, this can be considered the “perplexity” of a human.

Recall that perplexity is usually used as a metric to measure the quality of language models. Nonetheless, it would be challenging to calculate a direct estimate of the perplexity of a “perfect” language model. As a language model approaches optimality, its perplexity approaches the exponential of entropy of the language itself. Cover and King (1978)’s variation of the Shannon Game (Shannon, 1951) tried estimating the entropy of English, but this was calculated at the character-level and was based on pure human subjects. Brown et al. (1992) proposed estimating the upper bound for the entropy of English by training a 3-gram language model, but this was also at the character-level and language models have made dramatic breakthroughs since then. Word error rate in speech recognition is strongly correlated with perplexity regardless of the language model architecture (Klakow and Peters, 2002; Sundermeyer, Ney, and Schlüter, 2015). However, a language model is one of two components in a classical speech recognition system, where its role is to score how well options for recognized language units fit together. The other component, the acoustic model, generates these options from the acoustic signal. Disentangling the effect of language modeling and thereby estimating the performance of an ideal model is infeasible.

In this chapter, I describe the experiments that I helped to design and analyze. In the analysis, my coauthors and I assumed:

- A perfect language model can fully understand the mechanisms beneath a language and assign the word probability in a way similar to the human intuition.
- When asked to complete sentences with provided contexts, by greedily generating the most probable word, the generated sentences should be absolutely plausible.
- The generated sentences, when mixed with normal sentences, should be at least indistinguishable or have even higher scores with respect to plausibility.

On account of these assumptions, my coauthors trained a variety of language models, which ranged from the basic 3-gram count-based model to a long short-term memory (LSTM) model. These models were sampled so they greedily generated words given a sentence start with fixed 8 words. The generated sentences, together with the original ones, are then randomly shuffled and judged by humans. Every model is assigned a human judgement score for its generated sentences. The experiment revealed a strong correlation between the language model performance and the human judgement score. Based on the correlation, my coauthors and I estimated performance of a human-comparable language model by polynomial regression. It is worth mentioning that this estimated performance is just a lower bound, not the exact value, of a perfect language model as every assumption itemized previously is a necessary but not sufficient condition of a previous one. The experiment results show that in 2017, there was still a discrepancy between the models and the lower bound.

7.2 Language Models

Four classes of language models were featured in the experiments, n-gram & feedforward neural network (FNN), maximum entropy (ME), RNN with maximum entropy (RNNME), and RNN & LSTM. I describe them here so that I can more precisely argue that this formulation of the task may not obtain similar results if the experiment were to be repeated with contemporary state-of-the-art language models.

n-gram and Feedforward Neural Network Traditionally, count-based models are used to approximate such a probability based on the frequency information extracted from a training corpus (Katz, 1987). With the Markov assumption (Ney, Martin, and Wessel, 1997) and smoothing techniques, probability can be easily estimated from the counting information of n-grams. My coauthors trained 3-gram and 5-gram models with Chen and Goodman (1996)'s modified Kneser-Ney smoothing. Compared with the original Kneser-Ney smoothing, instead of using a single discounting parameter D, it has three different parameters D_1, D_2 and D_{3+} that are applied to n-grams with one, two and three or more counts. Experiments showed that this method outperformed other smoothing techniques (Chen and Goodman, 1996).

FNNs were the first neural network architecture introduced to language modeling (Bengio et al., 2003). Similar to count-based models, FNNs only consider the most recent $n-1$ preceding words in predicting the next word. Every word is mapped from a one-hot vector to a distributed representation where more semantic information can be encoded.

Maximum Entropy Maximum entropy (ME) is another popular model. With features and constraints, it tries to maximize the entropy of the word probability distribution (Rosenfeld, 1996). This model is computationally expensive but more features could be added to improve the performance. ME estimates the word probability as follows

$$p(w|h) = \frac{\exp(\sum_i \gamma_i f_i(w,h))}{Z(h)}$$

where f is the feature function, γ is the set of parameters to be learned and Z is a normalization term for a given history. My coauthors applied the ME extension of the SRILM toolkit (Stolcke, 2002) for training. In our case, only up to 5-gram features are used and $l_1 + l_2^2$ regularization is added to prevent overfitting. For parameter optimization, the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) method through the `libLBFGS` library is applied (Alumäe and Kurimo, 2010).

RNN with Maximum Entropy In comparison to FNNs, the RNN architecture has a recurrent layer to maintain the memory of all past information so that it can learn longer-range dependencies than FNNs. Maximum entropy can also be incorporated as part of the RNN model (RNNME), which would further reduce the perplexity (Mikolov et al., 2010, 2011b). The RNN part includes an input layer $x(t)$, a recurrent hidden layer $s(t)$ and an output layer $y(t)$ for each time step t. To speed up normalization, my coauthors divided words into, say 50, classes and the output layer is factorized as a class probability and a word probability (Mikolov et al., 2011c). Words are assigned to classes based on their unigram frequencies. RNNME trains ME as part of the RNN model. Since ME models and the softmax outlayer layer are similar, they can be viewed as a direct connection between input and output layer and the direct parameters can be learned during training (Mikolov et al., 2011b). In our experiment, only bigram and 3-gram features are used, a 1-billion-size hash (Mikolov et al., 2011b) is used to map such features in order to reduce the complexity and speed up the training process.

Vanilla RNN and LSTM The vanilla RNN is an RNNME without ME part. Though theoretically powerful, the training of RNNs is much more complex than FNNs. Because of the vanishing gradient problem (Bengio, Simard, and Frasconi, 1994), vanilla RNNs fall short of learning weight parameters in a way that long-range dependencies can be captured. To solve this problem, Hochreiter and Schmidhuber (1997) proposed the LSTM neural network architecture, further extended by Gers, Schmidhuber, and Cummins (2000) and Gers, Schraudolph, and Schmidhuber (2002). The resulting structure utilized a gating mechanism to ensure backpropagation of useful information through many time steps. Unlike vanilla RNN, where the effective backpropagation ranges usually up to 6 steps, LSTM can propagate errors for more than 20 steps without losing validity.

My coauthors trained the FNN, RNN and LSTM on multiple GPUs by separating training data into sequences of fixed length. An embedding layer and a projection layer is added to reduce complexity. They implemented batch noise-contrastive estimation (Gutmann and Hyvärinen, 2010) (FNN and RNN) and importance sampling (Bengio and Senécal, 2003) (LSTM) to handle vocabulary size at scale.

7.3 Human Judgement

Independently from language model training, my coauthors and I extracted 400 sentences from the test corpus, each containing at least 16 words. Sentences containing colons or quotation marks were filtered out beforehand because, as initial experiments confirmed, such punctuation severely interferes with human judgements and people themselves disagree on proper usage. For each sentence, the first 8 words were kept and the language models completed the sentence by greedily generating the most probable word until an end token was reached. If a language model did not output an end token within 50 words, the generated sentence was considered incomplete. In the experiment, only 3-gram models generated a fair amount of incomplete sentences; the other language models finished almost all of the sentences within the limit of 50 words. The incomplete sentences were *retained* for the experiment with ellipses appended to them.

A snippet of sentence examples is shown in Table 7.1.

Context	Wednesday was the first day at school for
Trigram	the first time in the first time in the …
5-gram	the first time in the history of the world .
ME	the first time in the history of the world .
RNN	the first time .
RNNME	the city 's history .
LSTM	the school 's president , who has been …
Original	quadruplets Sarah , Peter , Lucy …

Table 7.1: Example of generated sentences.

My coauthors and I randomly shuffled together all of the generated sentences, together with the original versions, and submitted them to the crowd-sourcing website CrowdFlower.[1] Participants were instructed that the first 8 words of the sentences displayed were the fixed context and the following words could have been generated either by human or machine. The participants rated on a 4-level Likert (1932) scale:

- 3 (clearly human)
- 2 (slightly human)
- 1 (slightly not human)
- 0 (clearly not human)

I specifically had the idea to have the participants rate the sentences on a 4-level scale instead of the more common 5-level or 7-level scale. That way, participants were forced to

[1]`https://www.crowdflower.com`

indicate a preference on every sentence rather than choose a neutral score. Only native English speakers with the highest trust level on CrowdFlower (based on previous participation) participated. At least three different participants judged each sentence and majority vote determined the final score for each sentence. If all three participants disagreed, my coauthors and I collected more judgements dynamically until at least half of all judgements were in consensus. In total, there were 288 participants.

7.4 Experiments and Results

My coauthors derived the training and testing corpora from the One Billion Word Benchmark dataset (Chelba et al., 2013) collected from English newspapers with about 850 million words. As mentioned before, my coauthors pruned sentences containing special tokens. This led to a corpus containing approximately 700 million words. A small subset was reserved as the test corpus. The rest was split into 100 segments. The vocabulary was fixed to be all types that appeared in the first segment, which was 158451 words. Then, other tokens were mapped to UNK.

My coauthors trained the 3-gram, 5-gram and ME on both the first segment and the whole data set, 6 language models in total. They trained a 5-gram 500-1500-600 FNN on the entire training set. 500, 1500 and 600 refer to the embedding size, hidden layer size and projection size respectively. My coauthors trained the RNNME incrementally on the first 1, 3, 6 until 21 segments with the hidden layer size fixed at 600, so 8 language models in total. Since RNNME models can only be trained sequentially on CPUs, the training process was quite slow. So, training was stopped at 21 segments, which constitutes around 20% of the whole corpus. My coauthors trained a 500-1500-600 vanilla RNN on both the first segment and the whole corpus. They also trained seven LSTM models on the whole corpus, all of which sharing the same embedding size of 512 but differing in the state size, projection size and drop-out rate.

7.4.1 Uncertainty of Data

This experiment involved ranking language models by perplexity values that were somewhat close. So, I had an imperative to ensure that the differences in perplexity were robust (significant) enough to justify the rankings. So, my coauthors and I adopted the uncertainty measures and correlation adjustments from Klakow and Peters (2002).

All data points in the experiment were assumed to be independent, as in one sentence does not influence the score of another. According to central limit theorem, the arithmetic mean of a sufficiently large number of independent random variables is approximately Gaussian-distributed (Bárány and Vu, 2007) with standard deviation $SD_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$. Here my coauthors and I used sample standard deviation s to approximate the population standard deviation σ. The uncertainty of the measurements could then be defined as $x \pm 2SD_{\bar{x}}$. The factor of 2 yields a 95% confidence interval.

When dealing with functions, uncertainty can be transformed by

$$\Delta f(z_1, \ldots, z_n) = \sqrt{\sum_{i=1}^{N} (\frac{\partial f}{\partial z_i})^2 (\Delta z_i)^2}$$

where z_i is the uncertainty of each parameter (Klakow and Peters, 2002).

After fitting the data points with polynomial regression, my coauthors and I measured the goodness of fit with adjusted R^2 (Cameron and Windmeijer, 1996)

$$R^2_{\text{adjusted}} = 1 - \frac{(1 - R^2)(N - 1)}{N - k - 1}$$

where R^2 is the sampled R^2, and N and k are the number of samples and regressors. Compared with normal R^2, R^2_{adjusted} imposes an increasing penalty as the number of regressors increases to prevent overfitting.

Table 7.2 gives the full list of the language models featured in the experiment. As an example, L-2-4096-1024-0.1 denotes a 2-layer, 4096 state size, 1024 projection size, 0.1 drop-out rate LSTM language model. This was the largest model that fit into Titan X GPU memory.

Model	PPL	Rank	Top1(%)	Score
3-gram-1	303.2	3.48	19.7	0.11
3-gram-all	112.2	2.75	24.0	0.16
5-gram-1	281.0	3.43	21.1	0.24
5-gram-all	73.7	2.43	31.2	0.60
ME-1	286.5	3.46	21.2	0.27
ME-all	68.8	2.40	31.8	0.64
FNN-all	83.0	2.56	26.3	0.56
RNN-1	211.1	3.28	21.5	0.33
RNN-all	45.7	2.12	31.9	2.08
RNNME-1	196.3	3.21	22.2	0.44
RNNME-3	136.0	2.93	23.7	0.41
RNNME-6	109.7	2.78	24.8	0.43
RNNME-9	107.5	2.76	25.4	0.42
RNNME-12	103.1	2.72	25.0	0.40
RNNME-15	91.3	2.63	26.1	0.48
RNNME-18	106.9	2.76	24.0	0.44
RNNME-21	78.9	2.52	26.9	0.71
L-1-512-512-0.1	63.2	2.41	30.0	1.36
L-1-1024-512-0.1	54.5	2.29	31.8	1.86
L-1-2048-512-0.1	45.3	2.19	33.1	2.39
L-1-8192-2048-0.5	35.9	1.95	33.8	1.54
L-1-8192-2048-0	37.5	1.97	34.8	2.60
L-2-2048-512-0.1	39.8	2.09	35.0	2.91
L-2-4096-1024-0.1	33.6	1.94	36.2	3.51
Human (estimated)	12.0	1.14	40.5	7.95

Table 7.2: Performance of language models in LM Turing Test

7.4.2 Metric-based Performance

My coauthors and I tested the performance of language models with three automatic evaluation metrics: perplexity, mean log rank, and percentage of the target word's probability being ranked first (Clarkson and Robinson, 1999). Only the generated part of each sentence (everything after the first 8 words) factored into the calculation. Table 7.2 contains all results for the models that were tested. As shown, the three metrics are highly consistent with each other. Also, the models that had larger training corpora had a substantial advantage.

For RNNME, increasing the size from 1 to 21 segments continuously brought the perplexity from 196 down to 78, nearly monotonically. As expected, the 3-gram performed the worst among all the language models. FNN (83.0) performed worse than the 5-gram (73.7) despite consuming much more training time. ME trained on the whole corpus performed surprisingly well (68.8) with only up to 5-gram features included. The best LSTM model took 2 weeks to converge and achieved a perplexity of 33.6, only slightly worse than the reported record (30.0) from Jozefowicz et al. (2016).

7.4.3 Human Judgement Score

Let n_i denote the number of sentences being judged with score i. My coauthors and I noticed n_1 (slightly not human) and n_2 (slightly human) were quite similar over all language models. After manually examining these sentences, my coauthors and I found they were quite difficult to be judged clearly. n_0 and n_3 were more reliable. In consequence, the final form of the humanness metric in this experiment was set to be the ratio of n_3 (clearly human) to n_0 (clearly not human).

The last column of Table 7.2 shows the human judgement scores. Models with better metric scores normally, though not always, had better chances of successfully fooling humans. One major exception was the 3-gram model trained on the full corpus, which achieved a very low human judgement score (0.16) with an intermediate perplexity (112.22). Original sentences received a human judgement score of 7.95, which is a big lead over all language models, even to the best LSTM model (3.51). It is worth noting that even for original sentences, quite a few of them were judged as clearly not human since they were novel, contained professional terms, or had problematic artifacts from tokenization. This suggests that the real human performance is even higher in general, highlighting that our estimate is a lower bound.

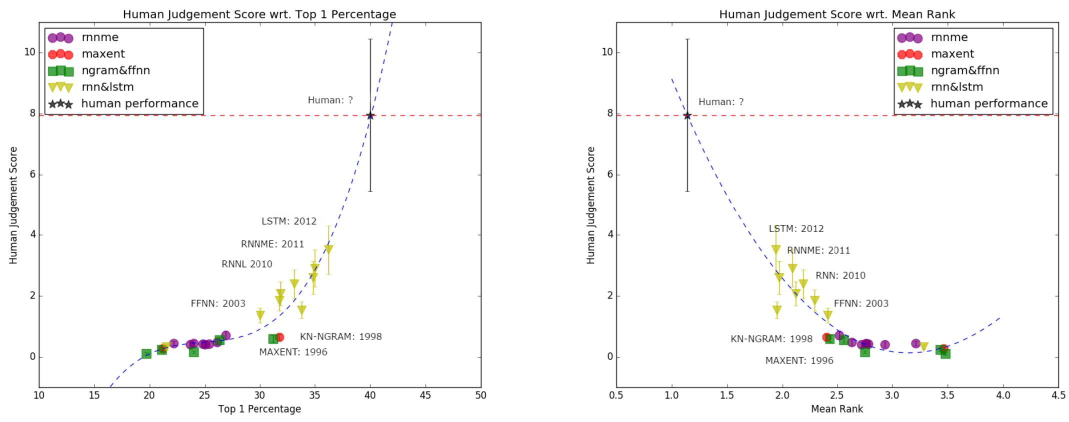


Figure 7.1: (Left) Human Judgement Score versus Top 1 Percentage, Adjusted R^2: 0.955. (Right) Human Judgement Score versus Mean Rank, Adjusted R^2: 0.934.

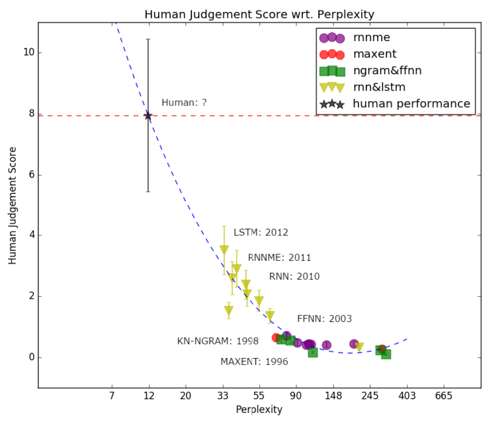


Figure 7.2: Human Judgement Score versus Perplexity, Adjusted R^2: 0.953.

Figure 7.1 and Figure 7.2 illustrate the correlation between human judgement score and each metric: perplexity, top 1 percentage, and mean rank. My coauthors and I fit a third degree polynomial to each plot (blue curve). The horizontal dashed line stands for the human judgement score from the original sentences. These lines can be viewed as the lower bound of a "perfect" language model. The points of intersection between the dashed lines and the fitted curves can then be used to approximate our three metrics for a "perfect" language model. My coauthors and I further annotated the points in the scatter plots with years of publication. Note the logarithmic x-scale in Figure 7.2 and the linear x-scale in Figure 7.1. The adjusted R^2 was greater than 0.9 for all three automatic measurement metrics. The fitted curve suggests a perplexity of 12 for a human-comparable language model. The suggested values for mean log rank and top 1 percentage are 1.14 and 40.5%. Interestingly, given the average word length of 5.5, Shannon (1951) estimated the lower bound on human-level word perplexity as $2^{0.648 \cdot 5.5} = 11.8$, which is consistent with our result.

7.5 Conclusion

This work had participants explicitly rate whether they thought sentences were human written or AI written. While the experiment rather conclusively showed that humans could tell the difference in 2017 with this specific data and these language models, there are some important limitations to note. The half sentence contexts, while appropriate at the time, probably are too short for language modeling capabilities and prompt engineering needs of today. Multiple human ratings per text would be more reliable. Lastly, there is much room to develop how the language models are sampled, and published methods have shown promising ways to increase the diversity of the generated text (Ippolito et al., 2019).

Following my work, Dugan et al. (2020) produced a similar, more advanced framework. They had participants guess where text shifted from human-written to AI written. This was a great way to make the task harder, which keeps it relevant as language models improve.

Chapter 8

LMs capture varying amounts of word length and frequency information

In Chapter 7, I shared the details of a study in which people rated human-written and artificially created text. I observed trends that the perplexity and the ratings were closely related, and the perplexities of state-of-the-art language models were steadily moving towards some ideal. The paradigm and trends seemed to persist even with more advanced language models than existed at the time (Dugan et al., 2020). But a language model divergence would predict that these trends would weaken or reverse. I first hypothesized the divergence when I saw the results from Chapter 7, but I did not (yet) have data to support the hypothesis.

For reasons of policing academic integrity, attributing authorship, and many others, it is of great importance to be able to tell reliably between human-written and artificially created text. This seems to have become difficult only recently. Although, as I mentioned in Chapter 1, this could be more about *accessibility* of technologies than the technologies themselves. As further anecdotal evidence for the accessibility argument, GPTZero (Tian, 2023a), a classifier for human-written versus artificially created text, debuted one month after the release of ChatGPT. GPTZero rose to intense popularity, securing high-profile relationships with educational institutions and others (Tian, 2023b).

Perhaps surprisingly, the core idea behind GPTZero is incredibly simple. Using your own language model, compute two metrics: the perplexity of the entire test document and the standard deviation of the perplexities of the individual sentences (Tian, 2023a). GPTZero set a decision boundary based on these, and later some proprietary, features to classify text as human-written or artificially created. I claim that this very simple technique works, in part, because of the language model divergence. I agree with Tian (2023a) that human-like text will have higher perplexity than artificially created text. The success of GPTZero directly suggests that this is true. Further, if language models continue to exploit information that humans do not use, I predict that the language model divergence will widen. So, it may become easier to distinguish between artificially created text and human-written text!

That said, there are a number of potential confounds that complicate the endeavor to bring scientific weight to these claims. By confound, I mean that there could be some other variable that affects both the perplexity and whatever metric for humanness at the same time.

Consider word frequency, defined in terms of the count of occurrences of a given word in a sufficiently large text. Kirchenbauer et al. (2023) proposed a way to "watermark" text generated by language models by making the language model generate words from a secret list more frequently than corpus statistics would suggest. The problem with using perplexity in this case is that the words in the test corpus all have different frequencies which *together* inform a single perplexity value. As such, perplexity is most likely not enough on its own to detect these frequency manipulations. I claim that a language model builder could also do an analogous manipulation for the lengths of words, with the added benefit that the relationship between average surprisal and word length has been more tested in the literature (Piantadosi, Tily, and Gibson, 2011; Meylan and Griffiths, 2021; Levshina, 2022).

A promising solution would be to consider not the perplexity of the test document, but the individual surprisals (negative logarithm of probability) of the words in the test document. That way, if there are any differences between what the surprisals are and what they "should be" according to some other model, this would be detectable. Specifically, the procedure is to compute the average surprisal for each word and correlate these against word length. Now note how cyclical this procedure is: a language model possibly created some text and then another language model is needed to provide individual surprisals for the words to compare against those from yet another language model (based on word length only).

It would be much simpler to have all language models under consideration score the same human-written text and compare the properties of the surprisals. I also argue that working with the surprisals rather than text generated from the language models constitutes using something that is more essential to the language model than whatever text was generated from it on isolated occasions.

In this chapter, I set up the computational framework for obtaining word-by-word surprisals from state-of-the-art language models. Then, I use the framework to address evidence for three hypotheses. First, I verify that average surprisals per word according to many types of language models are correlated with word length. Second, the overall relationship between perplexity and the sizes of these correlations is unclear, since fluctuations in the size of the test corpus and out-of-vocabulary (OOV) rate correspond to fluctuations in the size of the correlations. Third, comparing within families of GPT models, the surprisals from the larger models are less correlated with word length and frequency.

8.1 Background

In applications from Huffman (1952) coding to language evolution, there is a clear effort incentive for the most frequently used symbols to be the shortest. This observation by Zipf (1936, 1949) goes hand-in-hand with his most famous result, Zipf's Law: there is a negative linear relationship between the log frequency of a word in text and the log of its rank in a frequency list. Although languages are, for the most part, products of cultural evolution rather than consciously designed, the fact that languages universally uphold this property lends a sense of optimality to their design. Saffran, Aslin, and Newport (1996) observed that even eight month old infants are sensitive to the frequency distributions of syllables. So, this type of information is clearly available, although subconscious, during the diachronic processes that influence lexical innovation and evolution.

Piantadosi, Tily, and Gibson (2011) provided a groundbreaking refinement to the apparent relationship between word length and word frequency. Their work argued that it is the word's predictability in context, rather than its frequency, that has the stronger relationship with word length (in letters). To support this claim, they examined huge corpora across eleven languages with a straightforward operationalization of predictability in context. Namely, they trained unsmoothed n-gram language models (LMs) on their data, ran the same data through them, and computed Spearman rank correlations between the language model surprisals and the lengths of the words. In short, predictability is negative surprisal. Importantly, since an n-gram LM computes the probability of an upcoming word solely on the basis of the preceding $n - 1$ words, the notion of context in their work was a purely *local* one. To illustrate, consider the example sentence:

(1) GPT-4 is an **LM**.

Disregarding the dubious status of "LM" as a word, this two letter word is, in terms of overall English language use, quite infrequent. Yet, knowing the context that this sentence is about GPT-4, "LM" is quite *predictable*. Accordingly, a language model that considers at least three words of history might assign "LM" a low surprisal. A 2-gram or 3-gram language model might assign "LM" an especially high surprisal given that words that start with "L" almost never follow the word "an". Thus, the surprisals from the "better" language models are better equipped than frequency to predict that "LM" is a short word.

Despite the intuitiveness of the example and psycholinguistic reasons that make this finding attractive, the Piantadosi, Tily, and Gibson (2011) result has been the subject of several studies that doubt its robustness, especially in languages other than English. Meylan and Griffiths (2021) found that the Piantadosi, Tily, and Gibson (2011) effect is "attenuated" when following best practices of corpus linguistics. They pointed out that the choices for what counts as a word during LM scoring, which words should be included in the correlation analysis, and how to match vocabularies across languages can change the correlations. Additionally, they noted that web data may not be suitably created by humans for this analysis, using ASCII to encode words may have collapsed important distinctions between words, and Piantadosi, Tily, and Gibson (2011) should have used dictionaries of words[1], to select lists of words for the correlation analyses.

[1]They identified the Intercontinental Dictionary Series as a way to capture somewhat of a common list of semantic concepts among languages.

Just as with Meylan and Griffiths (2021), Levshina (2022) found that after considering some important factors, especially differences among properties of noun phrases in different languages, the Piantadosi, Tily, and Gibson (2011) effect was attenuated. Levshina (2022) even found that running a backwards language model (one that only uses the future words as context to predict the past) was sometimes a better predictor of word lengths than frequencies or regular surprisals. Specifically, Levshina (2022) found that the following are related to whether the Piantadosi, Tily, and Gibson (2011) effect is observed for a given language: 1) whether modifiers go before or after their objects, 2) how long the modifier and object words are (influenced by where the agreement markers go), and 3) "orthographic conventions" such as where the writing system includes spaces.

8.2 Common methods

In my experiments, I used the WikiText-103 corpus (Merity et al., 2016) because of its comparable size to the British National Corpus and because many contemporary language models published perplexity benchmarks on this dataset. The corpus contains 28,475 English Wikipedia articles in the training set and 60 in the test set. Merity et al. (2016) preprocessed the text, replaced rare words, and preserved the hierarchical structure of each article by surrounding headings with various numbers of '=' symbols. For instance, titles start and end with '=', sections with '= =', subsections with '= = =', etc. Splitting on white space, it has 101,425,658 tokens in the training set and 241,211 tokens in the test set. Merity et al. (2016) reported slightly larger numbers, perhaps using different tokenization.

I experimentally found some support for the Piantadosi, Tily, and Gibson (2011) claim in this dataset, provided that I performed the analysis on at least 10 million tokens in the training set. So, I extracted the first 3,085 articles from the training set into a new version of the corpus that I call "10M". Splitting on whitespace, this corpus has 10,004,966 tokens.

Recall that in Chapter 1, I *defined* a language model as a program that associates individual surprisals to words (more precisely tokens). So, it may be surprising that I need multiple pages to describe my process for obtaining these numbers from language models that have already been trained. While it is true that the application programming interface from huggingface contains commands that release the surprisals, there are many, many complications that could render the numbers unusable for my purposes. Note that for several conventional use cases of language models, the specific surprisal numbers are not needed, so any complication that makes the surprisals "uniformly" wrong or only occurs rarely might not affect the application too much.

I can only speculate as to why researchers such as Wilcox et al. (2020) explicitly mentioned that they could release their code for this but decided not to. My speculation is that they, too, discovered how complicated it is. I developed my own code base from scratch over several years, after not finding something existing that was suitably correct and customizable. When Oh and Schuler (2023) released their code for this, it further justified my claim that writing code to produce accurate word-by-word surprisals is a substantial accomplishment. Even they and I disagreed in interesting ways, and I recommend either my code adapted from theirs or my code written strictly before theirs in different situations.

In the next part of this section, I outline the most important complications for computing word-by-word surprisals using contemporary language models. Then, I finish with a discussion of the data that I used and the most important accompanying choices.

I developed **customizable preprocessing** for my surprisal calculation pipeline as language models require that the evaluation text is formatted similarly to the training text. For example, if the training text is all lower case, the evaluation text must be converted to all lower case. I wrote code to handle lowercasing, automatic addition of beginning of text and / or end of text tokens, reformatting of special tokens such as *unk*[2] that already exist in the corpus, transliteration of German umlauts, cleanup of extraneous white space, and removal of punctuation characters, as appropriate on a case-by-case basis.

After preprocessing, the text needs to be split into "sentences" for the purpose of providing good contexts for the word position. For Wikipedia articles, the most appropriate choice is to **split the corpus into its component articles**. The competing choice (treating the entire corpus as one document) does experimentally result in nominally better perplexities. Further, as far as I can tell, researchers such as Radford et al. (2019) used the competing choice when reporting their perplexity benchmarks. It comes down to whether the end of a Wikipedia article should be allowed to contextualize the beginning of the next one. While there are no guarantees as to a topic overlap between two Wikipedia articles, there could be other structural properties that positively inform the language model predictions just because they help the language model customize its predictions to the domain of Wikipedia articles. Indeed, large Transformer language models are usually trained on many types of text, not just Wikipedia, so their predictions are general. But, I assert that the first part of a Wikipedia article is the best contextualizing information, so I chose to split on articles as that favors these article introductions during batching (see below).

For the psycholinguistic datasets that I explore in later chapters, the most appropriate choice is to **split into sentences based on the experimental items**. It is best practice to shuffle the experimental items, so no one item is systematically influenced by another. Splitting this way leads to worse perplexities from the language models, but it more closely mirrors the conditions of the human psycholinguistic experiment.

After sentence splitting, the text needs to be tokenized, using either a **byte-pair encoding (BPE) tokenizer** specific to a Transformer language model, or whitespace tokenization for other models. huggingface tokenizers do not use UTF-8 encoding, meaning that characters with diacritics and characters in other scripts would sometimes be rendered with some "emergency" character string. This mattered because then the tokens would not always be a subset of the words they came from, which complicates the inverse process (detokenization).

[2]This denotes a token that occurred fewer times in the corpus than some threshold.

Also, during tokenization, a **suitable start token** needs to be inserted at the beginning of the text so that the first real token of the sentence can be scored correctly using the same process as that for all following tokens. I worked with three families of English GPT models, all available on huggingface: GPT-2 (Radford et al., 2019), GPT-Neo (Black et al., 2021), and OPT (Zhang et al., 2022). GPT-2 and GPT-Neo have the same token that starts and ends a text, but it is not automatically added to the sentence during tokenization, so my code handles that. OPT adds a start token automatically during tokenization. The German GPT-2 models that I used for experiments in later chapters did not always have an apporpriate token to start a text with, so I used a newline character in this case.

After tokenization, the sentences need to be batched because Transformer models usually have a large but limited maximum context size. GPT-2 has a maximum context size of 1024 tokens and GPT-Neo and OPT have a maximum context size of 2048 tokens. I found that about 14% of the articles in WikiText-103 are 1024 tokens or fewer and 41% of the articles are 2048 tokens or fewer. Therefore, I had to make a decision about how to handle the case in which the entire article did not fit within the Transformer context. On my own, I developed a **slow batching** solution. In this version, the batches are as large and overlapping as possible. So, for example, when scoring with GPT-2, the first batch would be the first 1024 tokens, and the model would predict the 1024th token using the previous 1023 tokens as context. Then, the window would move forward by one token. I call this slow batching because the model only makes one prediction per batch.

In contrast, the **fast batching** solution, based on code from Oh and Schuler (2023), uses a 50% overlap scheme. So for a long Wikipedia article being scored by GPT-2, the first batch would produce 1023 predictions (not counting the start token), and then the second batch would start halfway through the first batch, resulting in around 512 new predictions. While this method is faster, it is substantially less accurate because of the 50% overlaps. Effectively, some tokens would be predicted using half of the available context, rather than the longest context possible. Note that the start token and the token being predicted count within the 1024 tokens. So, a Wikipedia article exactly 1024 tokens long would still need two batches. The Oh and Schuler (2023) code that I used did not handle this case correctly and had to be changed.

While the shortened contexts from fast batching were not ideal, it made **feasible** some use cases that were not feasible using slow batching. With slow batching, the smallest GPT-Neo model could score a 241,211 words corpus in 20 hours. With fast batching it took one minute. With fast batching, the smallest GPT-Neo model could score my largest corpus (10 million words) in 90 minutes. The estimated 60 days of processing time to use slow batching instead was not feasible.

For my code that generates one prediction per batch, I was able to use the log likelihood function directly from huggingface. The fast batching method computes a softmax over the output logits for each batch. The output from both versions of this code is a list of surprisals, one for each token according to the tokenizer specific to the language model that generated the surprisals.

As the last step in the GPT pipeline, the system needs to undo the tokenization and aggregate the token-by-token surprisals so they become word-by-word surprisals. This was challenging to implement because each word comprised some variable and unpredictable number of tokens. However, the uncontroversial mathematical solution to this problem is to **sum the surprisals** of the tokens that comprise a word. By the properties of exponents, this is equivalent to multiplying the probabilities of the subword tokens. Once the false, but conventionally accepted, assumption that the subword tokens are independent of each other is adopted, that sum of surprisals is, by definition, the surprisal of the word.

Practically speaking, this was not easy because detokenizers that comes with language models do not provide a method to merge the surprisals alongside the tokens. So, I implemented two solutions. For the **special character detokenizer**, I observed that whitespace before a word appears as special character at the beginning of the first token of a word for GPT-2 family tokenizers, thus distinguishing these word-initial tokens. Combined with more logic for handling newline characters, this function successfully detokenized over ten corpora in multiple languages, including text with non `latin1`-encoded characters, such as Roman letters with accents and other diacritics.

However, this code absolutely relies on the existence of the special whitespace character, so I also adapted code from Oh and Schuler (2023) as a **greedy detokenizer**. This one works by matching tokens to parts of a current word. Once all parts of the word are accounted for, the system knows that the word is complete, so it computes the sum of the token-level surprisals and moves on the next word. While this solution does not expect or require any whitespace characters, it does require that the tokens are all strictly part of the original words. Unfortunately, as discussed within the tokenizer paragraph, this was not always the case. Thankfully, one of the two detokenization methods worked for all of my experimental configurations.

As the pipeline above was architecture-specific for tokenization, batching, scoring, and detokenization, I had to write separate pipelines for BERT and for LSTM language models. The BERT pipeline used the **fill-mask pipeline** from huggingface as its base. The fill-mask pipeline either supplies the top words to fill a masked position in a text, or gives surprisals for specific words that could fill the masked position. Because of BPE tokenization, I tokenized on whitespace, masked the relevant position, detokenized, and then BPE-tokenized the masked version of the sentence. According to huggingface documentation and experience with providing filler words for the mask that would have been tokenized by the BPE tokenizer into multiple parts, this process could successfully accommodate filler words comprising multiple tokens.

I queried BERT in two settings: "future" which gives the masked and encoded sentence to BERT as described, and "no_future" in which I **removed all tokens following** the mask token. For the LSTM pipeline, I adapted code from Gulordava et al. (2018). I substantially modified the way that the model was saved and loaded, and the batching code. The batching part was somewhat straightforward since, as a recurrent model, the context is theoretically infinite. So, I used each full article as a batch.

I trained a 2-gram, 3-gram, 4-gram, and 5-gram language model on the full WikiText-103 training set (100 million tokens) using the SRILM toolkit (Stolcke, 2002). After calculating the individual token surprisals, I averaged over the unique words (types) to get one average surprisal per type. I computed these averages while truncating the corpus at many points along a logarithmic scale, to see how they change as the size of the test corpus varies. This introduced a second way to generate frequencies. I could either use the frequencies from the entire corpus (100M freq) or just the portion of the corpus that is currently being used to compute the average surprisals (running freq). As expected, the correlations with word length for these two frequency calculation methods approached the same value because once the whole corpus was included in the surprisal averaging, the theoretical distinction disappeared. I converted all frequencies to surprisals so that the correlations would be in the same direction as the language model surprisals. And, I verified that the log transformation did not alter the Spearman correlations, as remarked in Levshina (2022).

I obtained the LSTM fully pre-trained from Gulordava et al. (2018). From their supplementary materials, it is a two-layer model with a hidden and embedding size of 650, a batch size of 128, a dropout rate of 0.2, and a learning rate of 20.0, trained for 40 epochs.

Work by Oh and Schuler (2023) on eye-tracking reading times inspired me to investigate GPT models within the same family but at different sizes. The GPT models that I used were as follows: GPT-2 in the "small" (GPT-2-SM) and "extra-large" (GPT-2-XL) sizes (Radford et al., 2019), GPT-Neo in the 125 million parameter (Neo-125M) and the 6 billion parameter (Neo-6B), also called GPT-J, sizes (Black et al., 2021), and OPT in the 125 million parameter (OPT-125M) and the 6.7 billion parameter (OPT-6.7B) sizes (Zhang et al., 2022). I used bert-base-cased (BERT) with 110 million parameters (Devlin et al., 2018), as well. All models were freely available on the huggingface hub.

To investigate the impact of quality of the frequencies, I also imported a list of the 333,333 most common types (1T freq) in Google's Trillion Word Corpus, made available by Norvig (2009). Piantadosi, Tily, and Gibson (2011) used the n-grams from this corpus (Brants and Franz, 2006) in their original analysis. I expected that these frequency statistics would be higher quality because the dataset is larger. So, I hypothesized that this would correspond to a higher correlation with word length.

Since Levshina (2022) used the Hunspell dictionary, I obtained the list for English and restricted my analysis to those words as well. This dictionary is quite clean and cased, meaning that words appear in their canonical casing, such as lower case for common nouns, first letter upper case for proper nouns, and all upper case for acronyms. But then I merged this dictionary with the 1T frequency list, which is not cased. Therefore, the only words left were those that are naturally all lower case. Finally, I removed all words with non-alphabetic characters. After this procedure, the number of tokens in each corpus was roughly halved, and the number of types varied from 1,000 to about 24,000 based on the original corpus size.

8.3 Results and Discussion

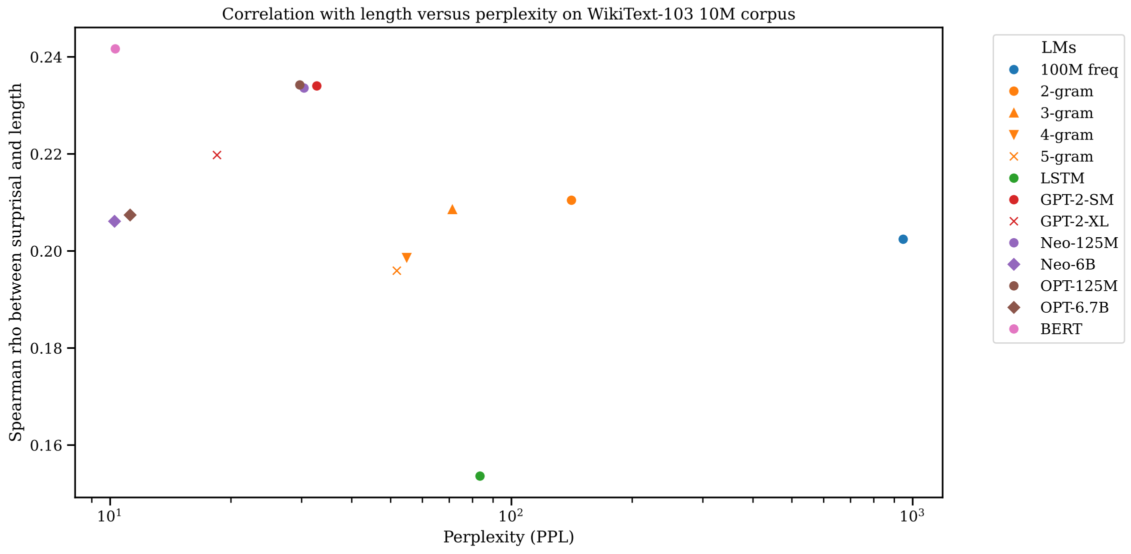


Figure 8.1: Scatter plot of the average-surprisal to word-length Spearman correlations versus perplexity for several language models.

My first hypothesis was that average surprisals per word according to many types of language models are correlated with word length. This has been tested before for n-gram language models, but to my knowledge, my experiment is the first to explore this for Transformer language models following the framework and best practices from Piantadosi, Tily, and Gibson (2011), Meylan and Griffiths (2021), and Levshina (2022).

The smallest correlation in Figure 8.1 is positive and very significant: Spearman $\rho = 0.15$, $p = 6.9 \cdot 10^{-81}$. I assert that the LSTM correlation was smaller than the others because the LSTM had a smaller vocabulary than the other models did. I did not test the LSTM further because the main focus is state-of-the-art language models, and the Transformer models performed much better. This is visible in Figure 8.1: the points for Transformer models appear much farther to the left. Points above the blue "M1" point in Figure 8.1 are consistent with Piantadosi, Tily, and Gibson (2011). The correlation coefficient for the points in Figure 8.1 is negative, but not significant: Spearman $\rho = -0.46$, $p = 0.12$.

In the figure, the perplexity for BERT was based on the regular "future" technique. This was necessary because with the "no_future" technique, the model has to score incomplete sentences unlike anything seen in training, resulting in abnormally high surprisals. However, for the correlation with word length, I used the "no_future" version as this one is comparable to the other language models in that it makes predictions in the forward direction only. The fact that the perplexity and correlation values for BERT are consistent with the other models justifies these choices.

LM	Steiger (1980) t-value	p-value	significance level
2-gram	1.52	0.13	
3-gram	1.11	0.26	
4-gram	-0.67	0.50	X (wrong direction)
5-gram	-1.14	0.25	X (wrong direction)
LSTM	-13.93	0	X (wrong direction)
GPT-2-SM	5.05	$4.4 \cdot 10^{-7}$	***
GPT-2-XL	2.65	0.0080	**
Neo-125M	4.92	$8.8 \cdot 10^{-7}$	***
Neo-6B	0.55	0.58	
OPT-125M	5.00	$5.7 \cdot 10^{-7}$	***
OPT-6.7B	0.74	0.46	
BERT	6.18	$6.4 \cdot 10^{-10}$	***

Table 8.1: Comparison of several average-surprisal to word-length Spearman correlations against the 100M freq to word-length Spearman correlation.

My second hypothesis was that the overall relationship between perplexity and the sizes of the average surprisal-word length correlations is unclear, since fluctuations in the size of the test corpus and out-of-vocabulary rate correspond to fluctuations in the size of the correlations. My results for this hypothesis are in Table 8.1, Figure 8.2, and Figure 8.3. As shown, the correlations using language model surprisals were often significantly higher than the correlation using frequency. But there are some important exceptions. The high quality 1T frequencies achieved a higher correlation than any of the language models. Granted, all of these language models were trained and tested on fewer than one trillion words. It is possible that the small version of GPT-Neo could overtake the 1T freq curve if I tested on a larger dataset. But, since the corpus size is on a log scale, it appears to need at least one billion filtered words (so perhaps two billion words total). I did not test this due to memory limitations of my computational setup.

Note that, as shown in Figure 8.2, at 241,211 words, the size of the WikiText-103 test corpus, the correlations nicely decrease as perplexity decreases. Although Piantadosi, Tily, and Gibson (2011) only claimed that surprisal *in general* correlates better with length than frequency does, it would be very reasonable to extend the theory such that better surprisals result in even better correlations. But at this corpus size, this is the opposite of what happened. That result, and those from other experiments that I ran with only medium size datasets over the years, led me to believe that the Piantadosi, Tily, and Gibson (2011) claim was false. However, with at least 10 million words and analyzing as many types as possible, my results are somewhat consistent with their theory.

But Figure 8.3, showing results from limiting the analysis to just the most frequent 1,000 words, is again inconsistent with the Piantadosi, Tily, and Gibson (2011) theory. This manipulation is somewhat similar to using a cross-linguistic dictionary of semantic concepts or

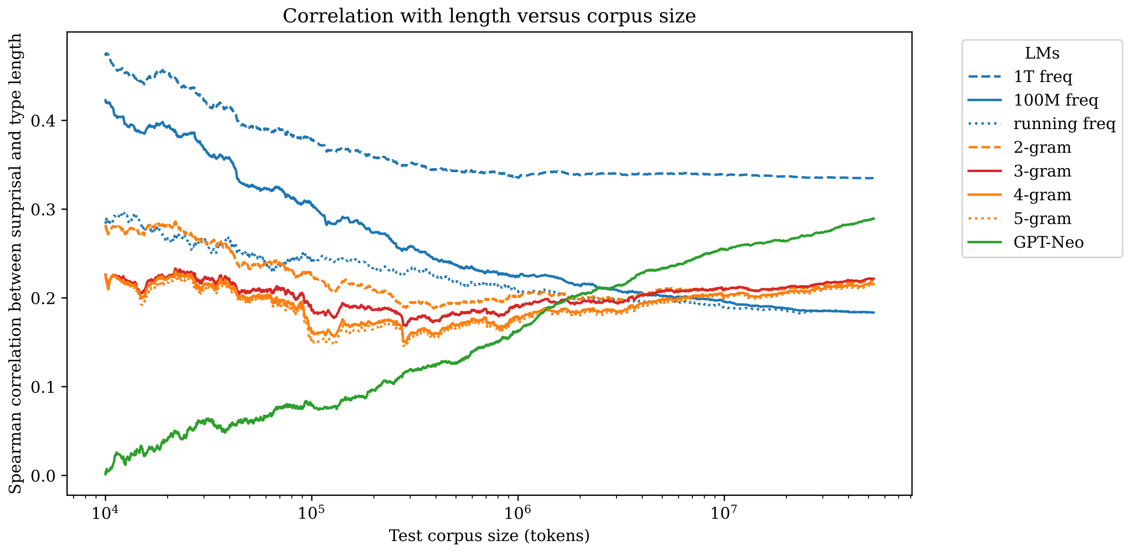


Figure 8.2: Average-surprisal to word-length Spearman correlations for three classes of language models as corpus size grows.

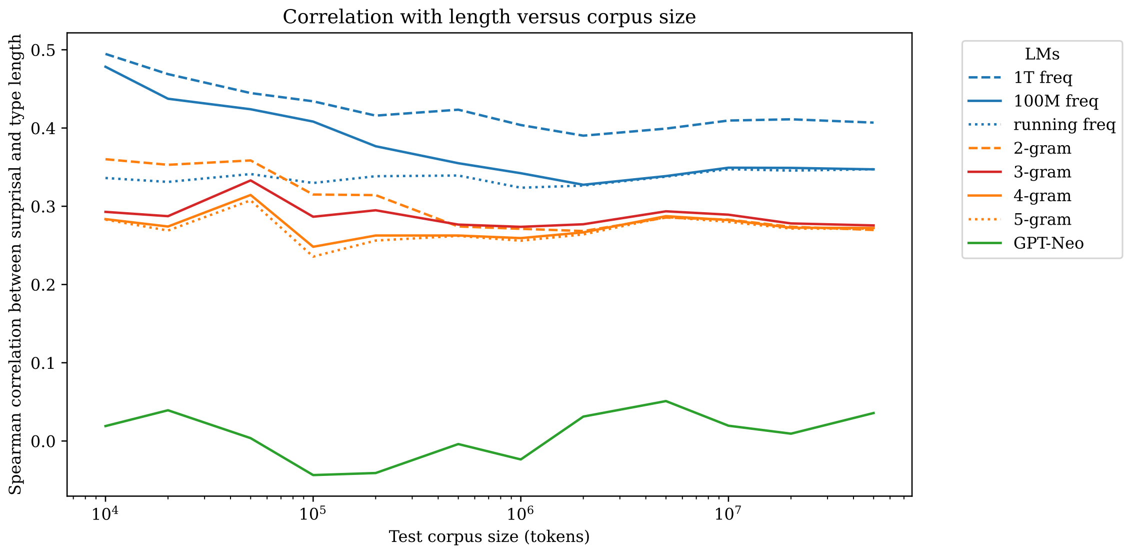


Figure 8.3: Average-surprisal to word-length Spearman correlations for three classes of language models as corpus size grows, limited to the top 1,000 most frequent words.

considering more words to be out-of-vocabulary, as recommended by Meylan and Griffiths (2021). Figure 8.3 also explains that GPT-Neo only starts correlating strongly after enough words are included in the analysis, suggesting that infrequent but correctly unsurprising words seem to interfere with the theory. Some spot-checking revealed that such words like "LM" in the introductory example are the strongest counters to the trend.

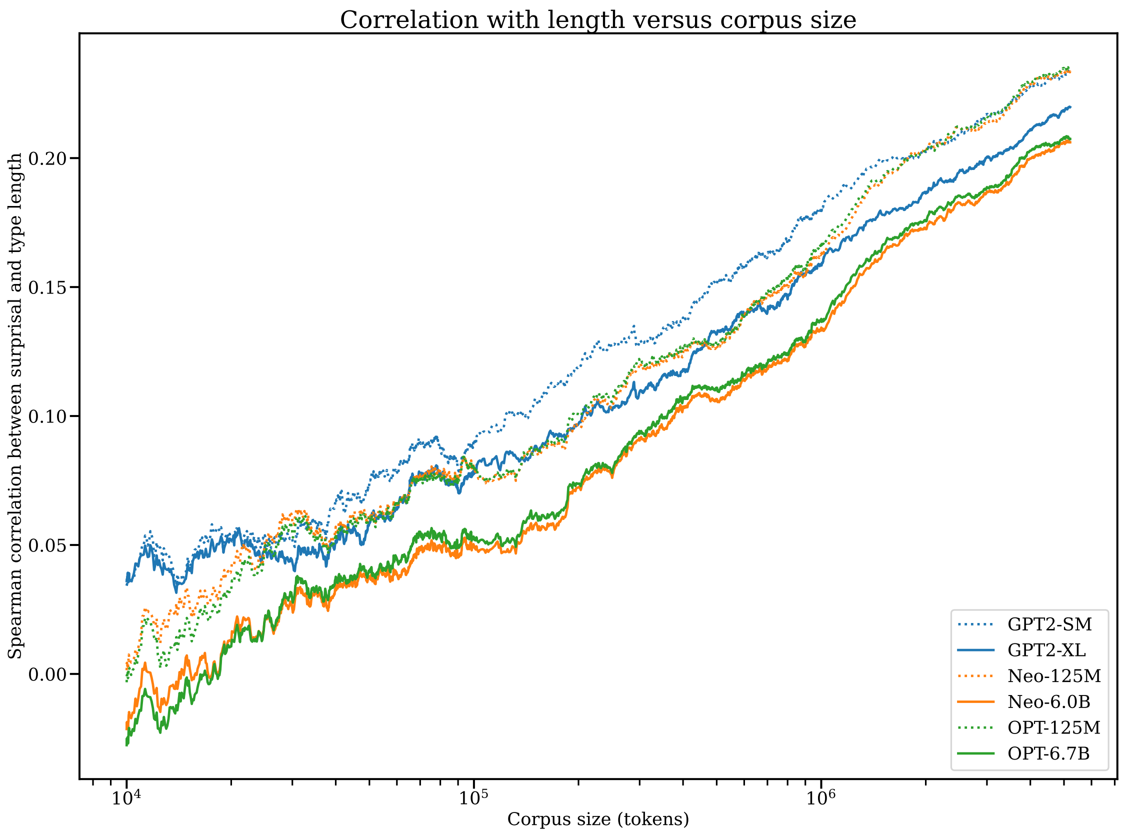


Figure 8.4: Spearman correlations between surprisal and type length for three families of GPT language models as corpus size grows. The dotted curves correspond to the "small" size models and the solid curves correspond to the "large" size models.

My third hypothesis was that comparing within families of GPT models, the surprisals from the larger models are less correlated with word length and frequency. Figure 8.4, Figure 8.5, and Figure 8.6 all support this from various angles. Figure 8.4 is in the same style as Figure 8.2 and Figure 8.3. For nearly all corpus sizes, the smaller models have higher correlations with word length. Note that GPT2-XL is much smaller than the other two "large" configurations (1.5 billion parameters versus 6.0 and 6.7 billion), so this hypothesis would also predict that the larger GPT-2 model would have higher correlations with word length than the other two larger language models. As shown in Figure 8.4, this happened as well.

For Figure 8.5 and Figure 8.6, I computed "simple" linear models (slms). These are linear models that do not contain so called mixed effects, which are a best practice when working with measures taken experimentally, such as human behavior data. Mixed effects separate out variance in the data that is assumed to be random, such as idiosyncrasies of a particular word or experimental participant. Since these models relate word length, word frequency, and average surprisal for all occurrences of a word, there were no experimental measurements taken.

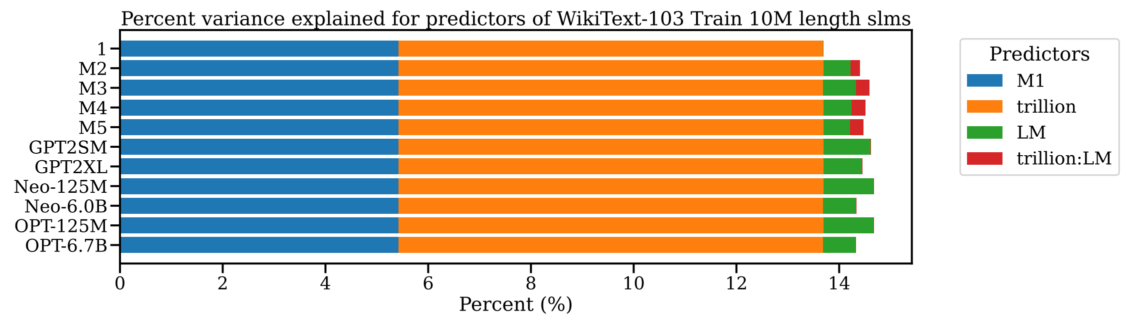


Figure 8.5: Percent of variance explained by each predictor for linear models of word length trained on the WikiText-103 train set 10 million tokens (10M).

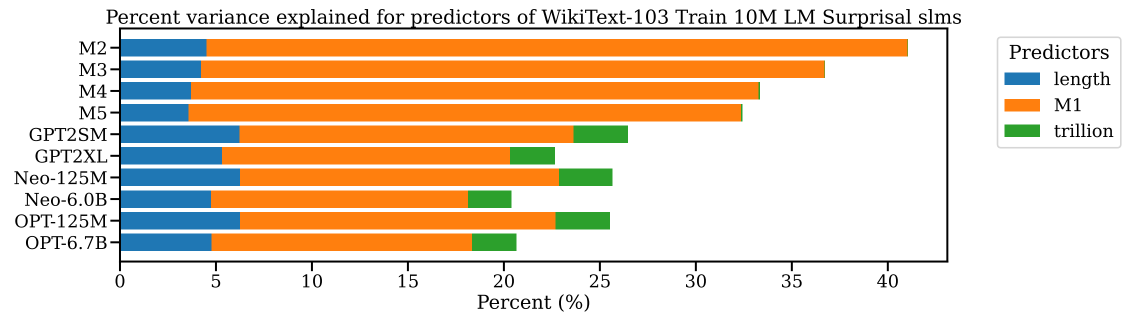


Figure 8.6: Percent of variance in LM surprisals explained by each baseline predictor for linear models trained on the WikiText-103 train set 10 million tokens (10M).

For Figure 8.5, the models learned a linear formula for word length in terms of the full WikiText-103 training set frequency (M1), the frequency from the Trillion Words Corpus (trillion), the average surprisals from the various language models explored in this chapter (LM), and the interaction between the trillion frequency and the LM surprisal. M2, M3, M4, and M5 correspond to the n-gram language models with the numbers representing n. The “1” slm did not use any nontrivial language model, just the two frequency variables M1 and trillion.

I computed an analysis of variance table for each of the models, which associates a percent of variance explained with each of the predictors in the model. The figure shows that the smaller GPT model surprisals explained more variance in word length, and the interaction between the high quality trillion frequencies and the language model surprisals made up most of the difference between the n-gram models and the Transformer models.

For Figure 8.6, I trained models that predict in the other direction, so length and two kinds of frequency were used to predict the language model surprisals. The figure shows that the lower the perplexity was, the smaller the variance explained was for length, M1 frequency, and overall.

In conclusion, the percent variance explained values decreased monotonically with perplexity, but the correlation with word length increased from n-grams to Transformers and then fell back as the Transformers grew. I assert that correlating word length or frequency against surprisal values is a first indication of the language model divergence. However, the results are dependent on dataset size, other implementation choices, and a foundational assumption that is somewhat disputed in the psycholinguistic literature. Namely, if the Piantadosi, Tily, and Gibson (2011) result were verifiable in more configurations, it would be easier to claim that correlations with word length or frequency are robust metrics of humanness. But since this is not the case, I find it worthwhile to investigate other metrics against which to compare the language model surprisals.

These experiments have the limitation that they do not exhaustively evaluate whether the result from Piantadosi, Tily, and Gibson (2011) remains supported for contemporary language models. I followed best practices from subsequently published critiques of that result, but I did not scale up to the prohibitively large dataset size needed to take on the main result. I stopped short of this not only for the memory considerations, but also because I do not need to verify or assume that result for the bigger picture of the thesis. Instead, the results presented in this chapter are intermediate in nature. The three hypotheses serve as small, self-contained scientific findings that support the idea that my framework for calculating word-by-word surprisals produces sensible and sanity checked surprisal values. They also underscore the importance of accounting for length and frequency effects when evaluating language model surprisals against proposed humanness metrics.

Chapter 9

Perplexity is sometimes dissociated from fit to psychometric data

In this chapter, I further develop the idea that language model surprisals should be compared against suitable humanness metrics as a way to evaluate the language models that is in some ways "superior" to perplexity. Recall that I theorized a language model divergence in which perplexities would continue to improve but only because the language models were learning patterns that humans do not use. So, after a language model divergence, perplexity and metrics of humanness would decrease together.

While I used correlation with word length as a humanness metric in Chapter 8, I selected two online reading behavior measurement paradigms for this chapter: eye-tracking fixation durations and G-Maze response times. It is common knowledge that longer words take longer to read and more frequent words take less time to read, so it is clear that word length and word frequency are correlated with these online reading behavior measures. In Chapter 8, I showed that word length and word frequency are correlated with language model surprisals. Therefore, when predicting online reading behavior measures from language model surprisals, length and frequency are positioned to confound the results. As such, the psycholinguistics literature (e.g. Wilcox et al., 2020) has coined *psychometric predictive power*. This term refers to the difference in log likelihood of a psychometric (in my case, the reading behavior measurements) between a mixed effects model that incorporates language model surprisals and a mixed effect model that does not. These mixed effects models account for baseline fixed effects, including length and frequency, and random effects, including an intercept for each experimental participant. By comparing a mixed effects model against one that does not factor in language model surprisals, The change in the log likelihood represents the psychometric predictive power of the language model *above and beyond* length and frequency.

Before discussing the results of several psychometric predictive power analyses, I present in Section 9.1:

9.1.1 background for and a systematic comparison of measures derived from eye-tracking fixation durations

9.1.2 descriptions of important properties of the eye-tracking and G-Maze datasets analyzed in this work

9.1.3 a literature review of work that relates language model surprisals and eye-tracking

9.1.4 a list of baseline predictors that have been explored in the past and my accompanying decisions whether to include them in my models.

Then, I discuss evidence for three important hypotheses for this thesis.

9.2 I show the outcome of the structure of baseline, experimental, and random effects that I hypothesized were appropriate to include based on Section 9.1.4.

9.3 I show that language model surprisals

(a) explain more variance in G-Maze response times than they do in total fixation durations.

(b) explain more variance in total fixation durations than they do in first pass reading times.

(c) explain more variance in first pass reading times than they do in first fixation durations.

9.4 I show that a previously known dissociation between perplexity and psychometric predictive power becomes more extreme with dedicated handling of part of speech, which is somewhat inconsistent with existing explanations of the dissociation.

9.1 Background and common methods

9.1.1 Measures based on eye-tracking fixation durations

The movements that human eyes make during reading can be seen as both simple and complicated. With the use of high-precision recording equipment, it is now somewhat common knowledge that eye movements can be categorized into at least two groups: fixations and saccades. During a fixation, the eye stops moving while the reader attends to a small visual field. During a saccade, the eye moves quickly from one fixation to the next. According to Engbert, Longtin, and Kliegl (2002), the fovea (the central region of the retina used for sharp vision) processes a visual field of about one degree wide. That translates to about 6-8

Roman characters. The eye can pick up some information beyond this area (a process called parafoveal vision), but the clarity or resolution declines substantially with distance from the foveal visual field (Engbert, Longtin, and Kliegl, 2002).

This is especially important for the present experiments because it is possible for a word to be only partially within the foveal visual field. The eye-tracking device identifies the center of the foveal visual field, and this point is usually unambiguously within a word. So, it is mostly reasonable to say that the eye is fixating *on* that word during a fixation, with the caveats that the whole word may not be within the foveal visual field during the fixation, and other words may be wholly or partially within it as well. Suppose, as an example, a fixation centers on the first letter of a ten-letter word that follows the word "a". Conventional eye-tracking procedure would count this as a fixation on the ten-letter word, even though "a" is wholly within the foveal visual field, and less than half of the ten-letter word is within the foveal visual field.

Even if the entire word is within the foveal visual field, it is perhaps common sense that if a word is longer, i.e. has more characters, there is more information to perceive and transmit to the brain, so the reading time is longer. This can manifest either in a longer lasting fixation, more distinct fixations, or some combination of the two. This leads to the first of three "derived" eye-tracking measures that I discuss and use as data in this chapter. The first fixation duration (FFD) of a word is the time, usually in milliseconds, of only the first fixation whose measured center is within that word. If the eye saccades to another point within the word, that time is *not* added to the FFD.

The sum of the fixation durations from the first time the eye fixates on a word to the first time the eye fixates on a different word is called the first pass reading time (FPRT). Perhaps surprisingly, not all saccades are in the forward direction of reading. While, of course, the reader can make a conscious decision to go back and read something that has been passed already, it is also possible for the eye to saccade backwards without such a conscious decision. But regardless of the direction or consciousness of the saccade, leaving the word signals the end of the FPRT.

The sum of all fixation durations for fixations that are centered within a word is called the total fixation time (TFT) for that word. Thus, the TFT includes the fixations from however many passes during which the eye stops on that word.

In summary, then, the FFD must be contained within the FPRT, and the FPRT must be contained within the TFT. If no fixation centers occur within a word, then that word is "skipped" and that word has no value for the FFD, FPRT, and TFT. As Demberg and Keller (2008) pointed out, this is not the same situation as having even one very short fixation. They argued, and I adopted, the notion that it is inappropriate to consider the FFD, FPRT, or the TFT to be zero in this case. Some datasets, e.g. Provo (Luke and Christianson, 2018), have zeroes for TFTs only, implicitly arguing that if there were no fixations, the sum of the durations is zero. However, as this was the only case in which zeroes occurred in the data, I straightforwardly removed these to bring the TFTs in line with the FPRTs and FFDs.

During a fixation, the human reader must at least identify the word being read, called lexical access. One of the largest debates in psycholinguistics involves the questions “What else does the reader spend effort on?” and perhaps separately, “What else does the reader spend time on?”. The Prediction Hypothesis states that most of time is used to identify the word. The Integration Hypothesis states that most of the time is used to integrate the identified word with previously obtained information. I organized Demberg and Keller’s (2008) position on this with respect to FFD, FPRT, and TFT in Table 9.1.

Measure	Conventional association
FFD	“lexical access, but also oculomotor processes and visual properties of the region”
FPRT	“early syntactic and semantic processing (as well as lexical access)”
TFT	“textual integration processes (as well as lexical and syntactic/semantic processing)”

Table 9.1: These explanations are quoted from Demberg and Keller (2008).

The predictability of a word in context (which is negatively related to surprisal, as discussed in Chapter 8) could theoretically inform any or all of the lexical access, syntactic, semantic, and integration processes. Consider a physical dictionary (book) as a metaphor for lexical access. In the extreme case, which psycholinguistic evidence does *not* support, the words in this dictionary are in a fixed alphabetical order, so one would use nothing more than the surface properties of a word (spelling) to locate it within the dictionary. In a slightly better, but still unsupported case, the words would be ordered according to their frequencies. Thus, “the” would be the first word in the dictionary of English, because this is the most frequent word in the dictionary. But it would be even more mathematically optimal if the words in the dictionary “rearranged” during reading, such that the most probable continuations of the speech or text would appear first. Ehrlich and Rayner (1981) are often cited as the first to show that beyond the length of a word, a highly predictive context can shorten FFDs significantly. So, there is at least some truth to the rearranging-words-in-the-dictionary metaphor. But, most scientific work that verifies this result uses discrete conditions of predictability, for instance, high predictability versus low predictability. So, the precise nature and size of this effect is not clear, as natural reading does not fall into such course categories of predictability.

Fortunately, the exact mechanism does not need to be fully understood before I can justify relating language model surprisals and the three eye-tracking measures. Predictability explains clearly some, and clearly not all, of these numbers. Since TFTs describe *all* of the time that a human eye was fixated on a word, and language models use *all* of the information that scientists have figured out how to represent computationally to make their predictions, I believe that TFTs should theoretically have the closest link to predictability in context. In Section 9.3, I show that this is true quantitatively.

9.1.2 Datasets

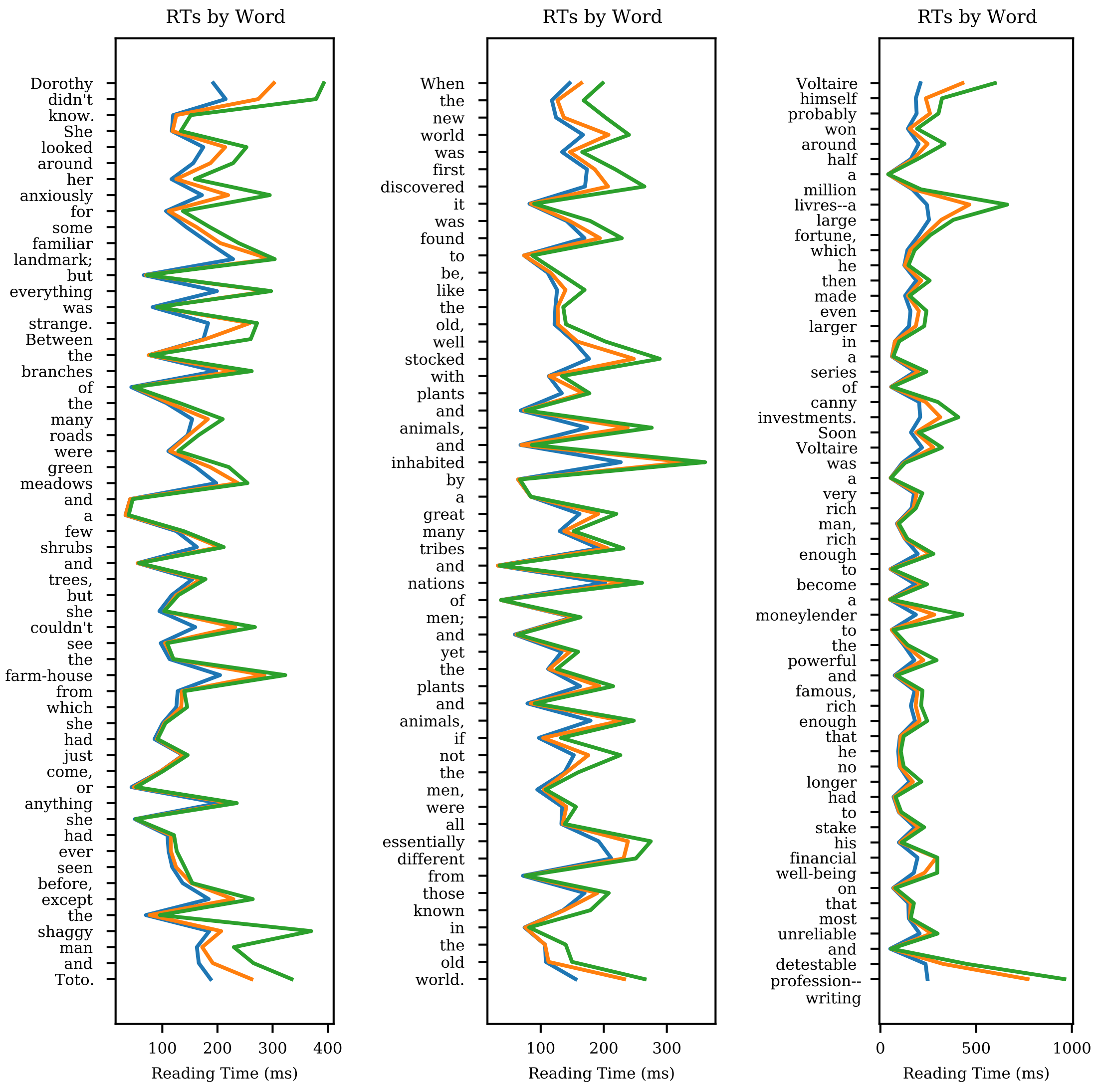


Figure 9.1: Provo example texts. Blue – FFD, Orange – FPRT, Green – TFT.

The Provo corpus (Luke and Christianson, 2018) contains 55 English "texts" of 39 to 60 words each. Each text contains 1 to 4 sentences, selected so that each text was approximately the same length. The left panel of Figure 9.1 contains the shortest sentence (*Dorothy didn't know.*) and the middle panel is all one sentence. Texts excerpted literature, history / encyclopedic, and news.

Luke and Christianson (2018) performed whitespace-based tokenization on Provo. Thus, "livres–a" and "profession–writing" in Figure 9.1 (right panel) were considered one word (region) each for eye-tracking purposes. The Transformer language model tokenizers of course separated these into multiple tokens, so I made sure that the non-Transformer language models also treated these as multiple words. Also, there were four encoding errors in the text in which an unknown character was encoded as a question mark. I do not know exactly how the text was presented to participants, but leaving these question marks in the text wildly affected the language model surprisal values. Therefore, I replaced the question marks with what I believe were the correct symbols (three apostrophes and one em dash), computed the surprisals on the corrected text, and threw out all six altered tokens from my analysis. This way, my analyses did not get an unfair advantage from my altered tokens, but the other words in the same text had more reliable surprisals because they were no longer affected by these encoding and tokenization irregularities.

Luke and Christianson (2018) measured cloze completions at each word (not used in this work) and FFD, FPRT, and TFT for 84 participants. Provo has 2,743 words. All words were fixated by at least one participant. I removed the 55 words that began a text, the 55 words that ended a text, and the 6 words that I described previously ("profession–writing" also ended a text). Therefore, my final Provo dataset contained 2,628 words.

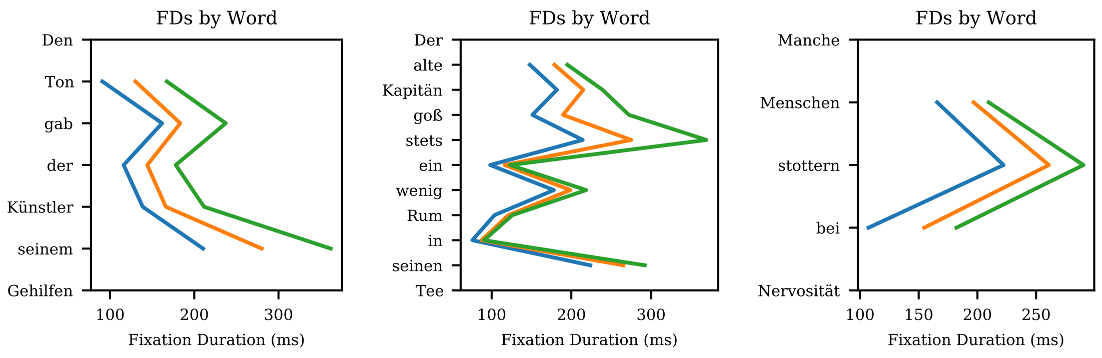


Figure 9.2: PSC example texts. Blue – FFD, Orange – FPRT, Green – TFT.

The Potsdam Sentence Corpus (PSC) (Kliegl et al., 2004; Kliegl, Nuthmann, and Engbert, 2006) contains 144 individual German sentences of 5 to 11 words each. Figure 9.2 presents the first (left panel), longest (middle panel), and shortest / last (right panel) sentence in PSC. These sentences were crafted to be uniform rather than drawn from diverse genres.

Most versions of PSC, such as Hohenstein, Matuschek, and Kliegl (2017), published single fixation durations (SFDs) for up to 273 participants on all words except the first and last in each sentence. So, if a participant skipped a word or fixated on it multiple times in the first pass, Hohenstein, Matuschek, and Kliegl (2017) removed the data for those fixations.

For uniformity and better appropriateness of the eye-tracking measures, I used the version of PSC from Boston et al. (2008, 2011) which measured FFD, FPRT, and TFT for 222 participants. PSC is 1,138 words long. Boston et al. (2011) did not report eye-tracking measures for the 144 words that began each sentence and the 144 words that ended each sentence. But, I would have removed these anyway. Ten of the remaining words were not fixated by any participant. These ten words all occurred in different sentences. Therefore, my final PSC dataset had 840 words.

The ZuCo corpus was the result of two large-scale projects that each measured eye-tracking and electroencephalography (EEG) simultaneously. ZuCo 1.0 (Hollenstein et al., 2018) had 12 participants and ZuCo 2.0 (Hollenstein et al., 2020) had 18 participants. When ZuCo 1.0 and 2.0 were published originally, the eye-tracking data (usually kilobytes or megabytes of data) were mixed with the EEG data (several gigabytes of data) and the participants were kept separate. Hollenstein et al. (2021a) released a combined edition of ZuCo 1.0 and ZuCo 2.0 with just the eye-tracking data (convenient file size) but the participants were averaged together. Hollenstein et al. (2021b) acknowledged that averaging over the participants makes analyses of the data much less meaningful because it becomes harder to claim that observed patterns are due to a robust behavior distinction and not random variation among the participants. But despite this, they left by-participant analysis for future work.

Over a process of several weeks, I painstakingly filtered the original ZuCo 1.0 and ZuCo 2.0, reverse engineered which sentences made it into the combined version, fixed bugs in the published data processing scripts, correctly handled missing data, and corrected miscellaneous encoding and file corruption errors. In a triumphant feat of data wrangling, the token count for my dataset with the by-participant data intact exactly matched the token count for the averaged dataset as reported in Hollenstein et al. (2021b): 20,545 tokens.

On account of the familiarity of the dataset that I obtained over this process, I can report that this dataset has 1,049 sentences. 400 sentences are movie reviews written in a quite informal style (e.g. "…a bland murder-on-campus yawner"), and the rest are from Wikipedia articles (300 from ZuCo 1.0 and 349 from ZuCo 2.0). Several of the Wikipedia sentences appear two and three times.

Out of the 20,545 tokens, I removed the 1,049 sentence-initial tokens and the 1,049 sentence-final tokens. An additional 106 tokens were not fixated. Finally, I removed 232 tokens because they contained no alphanumeric characters. Therefore, my final dataset contained 18,109 tokens. This size makes ZuCo the largest, commonly used, sentence-based, English dataset of eye-tracking measures. I use the term sentence-based to distinguish ZuCo from the Dundee corpus (Kennedy and Pynte, 2005) in which participants read entire newspaper articles and GECO (Cop et al., 2017) in which participants read an entire novel. While those two datasets are each approximately twice the size of ZuCo, very long passages make it much harder to control practical issues during data collection. For example, there is no hope of having the whole passage on one line, which means that for various words, the participant must shift their eyes to the following line. It is simple enough to throw out first and last words on a line, but it is possible that unwanted effects from moving to a new line spread to the other words, too. And, coming from the language modeling side, if the maximum context size for a language model is larger than the size of a context needed for humans to exhibit some behavior, it means more if the language model does not reproduce or predict that behavior. In that case, the language model would have had the capacity for it, yet it did not capture the pattern in the data.

One could argue that if the training data for the language model and the corpus on which the human behavior measures were collected were different enough, the language model would also not be able to capture the patterns that it needs as easily. Again, ZuCo is the best dataset for this, since most large language models are trained on web text[1]. Since ZuCo is composed of movie reviews and Wikipedia articles, ZuCo is an especially close match to the training data, rather uncontroversially closer than Provo, PSC, Dundee, and GECO.

In the grammaticality Maze task (G-Maze) (Forster, Guerrera, and Elliot, 2009), sentences are presented word-by-word as a sequence of forced choices between two alternatives, only one of which continues the sentence grammatically. If the participant successfully navigates the "Maze" by choosing the correct word from each pair, the selected words form a coherent sentence. Figure 9.3 shows an example G-Maze item.

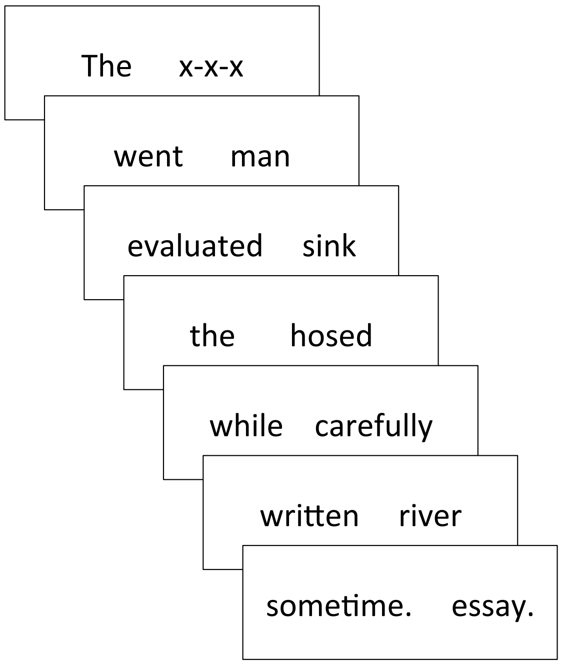


Figure 9.3: Example trial structure of G-Maze task. Sentences are presented word by word as a sequence of forced choices between two alternatives, only one of which continues the sentence grammatically.

I obtained a dataset of English G-Maze response times from Witzel, Witzel, and Forster (2012). This dataset contains 144 sentences, with 8 additional practice sentences that I discarded. I also discarded the first and last words of each sentence resulting in a final total of 1714 words. Witzel, Witzel, and Forster (2012) established that Maze response times are reflective of processing while getting rid of spillover effects. However, they did not comment on whether the response times also depend on the foil word at each choice, so I was sure to test this in my models.

[1]While Wikipedia was removed from the training data for GPT-2 (Radford et al., 2019), Wiki-style writing is ubiquitous on the internet. So, it is likely to still be represented in the model.

9.1.3 Previous frameworks for evaluating language models

A linking hypothesis links a behavior measure to a theoretical or mathematical concept. Especially under the Prediction Hypothesis as discussed in Section 9.1.1, there is an argument that the language model is completing a task similar to reading as measured by an eye-tracker. Admittedly, the G-Maze task is less natural and more complicated than eye-tracked reading. However, it is plausible that the G-Maze task must comprise at least the same elements as natural reading because the participant must identify the true word and the foil word, and they must integrate both candidates enough with the previous context that they can decide which word is the correct continuation of the sentence. As low surprisal facilitates reading, I formally assume that some of the variance in the three eye-tracking measures and in G-Maze response times can be attributed to surprisal. Of course, I accept that there are other factors at play as well, and take on the responsibility to address those other factors in my models so that any interference with the effects of interest is diminished.

The procedure in Demberg and Keller (2008) became a common starting point for this kind of careful analysis of eye-tracking data. They noted that there may be special behaviors at the first word of a sentence (sentence-initial word) and at the last word of a sentence (sentence-final word). I propose that as an example, while reading the sentence-initial word, perhaps the reader is still getting situated and the eyes move differently because of this. Similarly, at the end of the sentence, it is possible that the participant knows that they need to do something extra to move on to the next trial, and their eyes could move differently because of that. So, Demberg and Keller (2008) removed these words from the analysis. I adopted this in my work, too.

Other best practices included centering and scaling the data, log transformation if a histogram revealed that the data were skewed, and pruning insignificant predictors using a test based on the Akaike Information Criterion. Also, Hollenstein et al. (2021b) suggested scaling the dependent variable such that its values ranged from 0 to 100. That way, the mean absolute error could be more easily interpreted as a percent error and could be subtracted from 100 to give a notion that they called prediction "accuracy". Especially as this scaling of the dependent variable facilitated the convergence of my models, I adopted all of these best practices.

Goodkind and Bicknell (2018) presented a collection of very positive results concerning the interplay between FPRTs and language model surprisals. They trained n-grams up to $n = 5$, an LSTM, and linearly interpolated the LSTM with the 5-gram all on the One Billion Word Benchmark (Chelba et al., 2013). They evaluated on FPRTs from the Dundee corpus Kennedy and Pynte (2005) and preprocessed that data in a way similar to Demberg and Keller (2008). They analyzed seven language model configurations and these seven data points exhibited a monotonic behavior in the expected negative direction. The monotonicity led to an incredibly large R^2 value. They also noted that their LSTM was substantially better than the 5-gram in terms of perplexity, but on its own, it did not predict reading times all that much better. I interpret this to be an early indication of the language model divergence:

for the most part the psychometric predictive power and the perplexity have a negative relationship, but the Goodkind and Bicknell (2018) data already showed an attenuation of the negativity.

Building on Goodkind and Bicknell (2018), Wilcox et al. (2020) argued that improving *within* some language model architecture correlates with an improvement in psychometric predictive power, but found little pattern *among* architectures. They, too, argued against the psychometric power of the LSTM architecture: "For any given perplexity, deep Transformer models and n-gram models generally show superior psychometric predictive power over LSTM or structurally supervised neural models". They varied "inductive bias", which is how much syntactic structure the network uses, and amount of training data. They evaluated on FPRTs from the Dundee corpus (Kennedy and Pynte, 2005) and a self-paced reading corpus. They expanded the syntactic knowledge from binary (hierarchical or not) to a range. This range was not predictive of psychometric predictive power after accounting for perplexity. The overall relationship between fit to FPRTs and perplexity roughly looked exponential in the low perplexities and linear in the high perplexities. They found that syntactic generalization score and perplexity were related, but the residual syntactic generalization score after perplexity had no clear relationship with residual predictive power.

Somewhat at odds with Wilcox et al. (2020), Eisape, Zaslavsky, and Levy (2020) argued that their LSTM model correlated more highly with FPRT than other models with lower perplexity. They used the Provo corpus (Luke and Christianson, 2018) because it has cloze completions and eye tracking data on the same stimuli. Supported by findings from Luke and Christianson (2018), they pointed out that cloze predictions are too harsh. Humans probably "make many diffuse bets at multiple linguistic levels". Luke and Christianson (2018) referred to this as "partial match", e.g. correct part of speech but wrong word. Eisape, Zaslavsky, and Levy (2020) considered 5 LM architectures: 5-gram, LSTM, GPT-2, Transformer-XL, and XLNet. They trained the n-gram and the LSTM on WikiText-103. Transformer-XL was fine-tuned on WikiText-103. Eisape, Zaslavsky, and Levy (2020) obtained the other two from huggingface, so they were trained on their usual datasets.

They had three main contributions: 1) comparing LM surprisal distributions against cloze distributions, 2) comparing psychometric predictive power for FPRTs against perplexity, and 3) proposing Cloze Distillation as a way to "improve" the language models for predicting the FPRTs. Cloze Distillation interpolates in the KL-divergence between the surprisal distribution and the cloze distribution to the LM training objective. Most of the results that Eisape, Zaslavsky, and Levy (2020) reported showed insignificant and / or very small effect sizes. But, since Smith and Levy (2011) already made it known that cloze distributions and LM surprisal distributions are different, it was to a degree expected that including cloze information at least nominally improved psychometric predictive power.

Oh and Schuler (2023) presented a linguistic explanation for the results in Oh, Clark, and Schuler (2022), which, in turn, expanded upon the results from Oh, Clark, and Schuler (2021). The original experiment for Oh, Clark, and Schuler (2021) argued for a morphology component for predicting FPRTs. But along the way, as explained in Oh, Clark, and Schuler

(2022), they found that "experiments using Transformer-based GPT-2 models of varying capacities that share the same primary architecture and training data show a surprising negative correlation between parameter count and fit to self-paced reading and eye-tracking data. In other words, Transformer models with fewer parameters were able to make better predictions when the training data was held constant." So, while Wilcox et al. (2020) systematically varied the architecture and the amount of training data while keeping the number of parameters constant, Oh, Clark, and Schuler (2022) held the architecture and training data constant while varying the number of parameters.

I consider this result to be one of the most promising sources of evidence for the language model divergence. My work verifies it, identifies other behavior measures for which the pattern exists, controls for factors not addressed in their publications (this chapter), and based on a more controlled experimental design (Chapter 10), argues for a different and stronger explanation than the one provided in Oh and Schuler (2023).

Their primary linear mixed effects models of FPRT had baselines of word length, position in sentence, saccade length, and whether or not the previous word was fixated. I adopted all of these except saccade length and additionally included word frequency. They mentioned that they, too, explored including word frequency, but it did not meaningfully change the results.

Overall, they found very robust results that as the model size grew, the perplexity decreased (as expected) but the psychometric predictive power decreased as well (not expected). Therefore, when plotting psychometric predictive power versus perplexity, the relationship was positive.

One of the main arguments from Oh and Schuler (2023) was that larger Transformer models have more capacity to capture specialized knowledge, so they were more likely to underestimate the surprisals for nouns and verbs. They computed statistics over the raw language model surprisals and found that the largest discrepancies between surprisals for small models and for large models were on nouns and verbs.

Based on this, I hypothesized that if the explanation from Oh and Schuler (2023) is correct, by introducing two new binary variables in the model, one for whether the word is a noun (NOUN) and one for whether the word is a verb (VERB), and allowing these variables to interact with the language model surprisals, the unexpected positive relationship between psychometric predictive power and perplexity would lessen. Instead, it grew, as reported in Section 9.4.

9.1.4 Survey of successful predictors

Demberg and Keller (2008) mentioned word frequency (wf), word length (wl), position in sentence (x), and frequency of the previous word (pf) as "low-level" variables, and they mentioned whether the previous word was fixated, launch distance (characters), and landing position (character) as "oculomotor" variables. They argued that all of these, plus participant variance must be accounted for. I included the first three low-level variables, but previous word frequency was not significant in the models. I included a variable C(skip) that was True if the previous word was skipped and False otherwise. Essentially, this variable sets a parameter for how much time to add to the predicted variable (FFD, FPRT, or TFT) if the previous word was skipped. I did not include launch distance and landing position as these were not always recorded. This is likely because in Demberg and Keller (2008) and several subsequent studies, these were not always significant.

Many works proposed novel features that arguably improved fit to reading times for language models. These included a "cumulative n-gram" (van Schijndel and Schuler, 2015), PCFG surprisal, (van Schijndel and Schuler, 2015), accumulation for skipped words (van Schijndel and Schuler, 2016), entropy and entropy reduction (van Schijndel and Schuler, 2017), an adaptation mechanism (van Schijndel and Linzen, 2018), lookahead information gain (Aurnhammer and Frank, 2019), saliency (Hollenstein and Beinborn, 2021), morphological features (Oh, Clark, and Schuler, 2021), and coreference information (Jaffe, Shain, and Schuler, 2020; Jaffe, Oh, and Schuler, 2021). However, the point of these works was to model the reading times as accurately as possible, while my work aims to characterize how different language models explain different amounts of variance in the behavior measures. As such, I argue that features that could reasonably be included in the language model surprisals (many of the features listed above) should not be separated out given my purpose unless they have a clear reason for inclusion. For example, language models look up entire tokens at once, so they do not interface with word length the same way as humans do. Similarly, word frequency provides the easiest kind of surprisal information, so it might be worthwhile to see what else the language model can provide beyond frequency information. Also, including word frequency is a best practice for eye-tracking analyses, according to Hollenstein, Barrett, and Beinborn (2020).

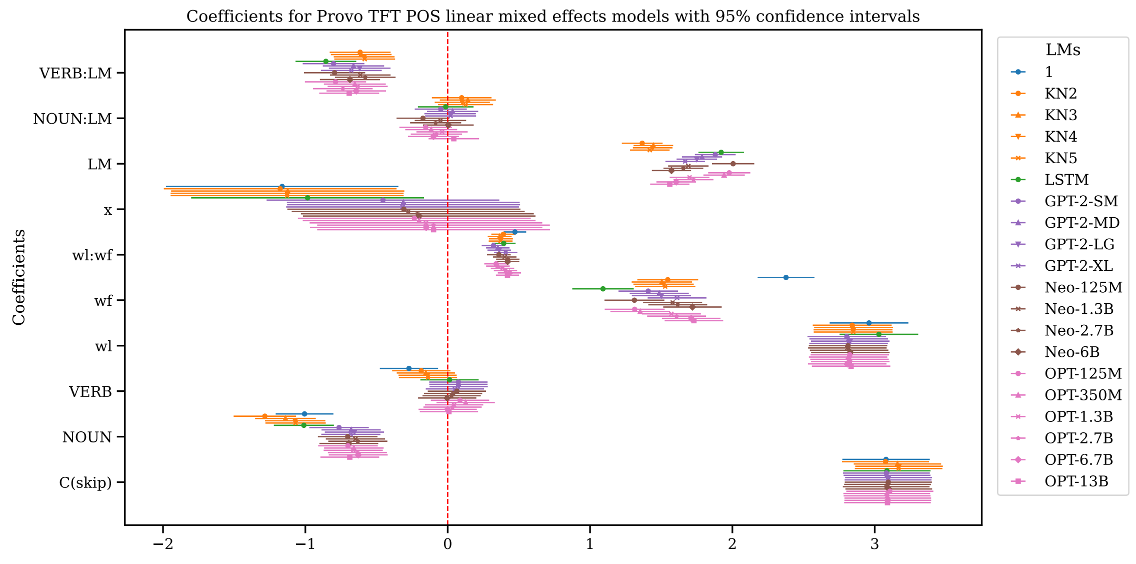
Salicchi, Xiang, and Hsu (2022) won the CMCL 2022 Shared Task on Multilingual and Crosslingual Prediction of Human Reading Behavior (Hollenstein et al., 2022a). They used sentence position, word length, word frequency, previous word length, previous word frequency, whether the first letter was capitalized, whether the word was in all capitals, the syllable count, and GPT-2 (monolingual) surprisal to predict TFTs averaged over participants. As such, the data did not allow them to use linear mixed effects models, but they tried all 36 two-way interactions, plus the 9 features and an intercept. So, there were 46 predictors in all. They also tried 9 different regressors. I claim that the extremely large number of configurations was an appropriate strategy for a shared task, especially one that had a test set in a mystery language. But for my purpose, it is best to have a small number of models with the features selected not for the absolute highest accuracy, but rather for making the most powerful and scientifically rigorous conclusions about the relationship between psychometric predictive power and specific language models.

For all of my experiments in this chapter, I obtained word-by-word surprisals according to the methods described in Chapter 8. For the English corpora (Provo, ZuCo, G-Maze), I obtained frequency information from the Corpus of Contemporary American English (Davies, 2008), and I trained 1-gram, 2-gram, 3-gram, 4-gram and 5-gram models with modified interpolated Kneser-Ney smoothing (Kneser and Ney, 1995) on the 101,425,658 token, 267,735 type WikiText-103 (Merity et al., 2016) training set using `KenLM` (Heafield, 2011; Heafield et al., 2013). Additionally, I obtained the LSTM fully pre-trained from Gulordava et al. (2018), which according to their supplementary materials, is a two-layer model with a hidden and embedding size of 650, a batch size of 128, a dropout rate of 0.2, and a learning rate of 20.0, trained for 40 epochs. Lastly, I obtained the language models from three GPT families in various sizes: GPT-2 (Radford et al., 2019), GPT-Neo (Black et al., 2021), and OPT (Zhang et al., 2022). When I refer to these models, their numbers of parameters appear after a hyphen with "M" signifying million and "B" signifying billion. Therefore, Neo-6B is the 6 billion parameter version of GPT-Neo, and this particular size is also called GPT-J.

For German (PSC), Kliegl et al. (2004); Kliegl, Nuthmann, and Engbert (2006) reported that they packaged frequency information with the dataset based on the CELEX corpus (Baayen, Piepenbrock, and Gulikers, 1995). Further, I focused on several pre-trained Transformer models as the point of analyzing PSC was to show that the English and German eye-tracking data related to Transformer model surprisals in a reasonably similar way. The Transformer models that I tested included two BERT (Devlin et al., 2018) models, one (BERT) native to huggingface, and another (BERTd) released by the Digitale Bibliothek Münchener Digitalisierungs Zentrum (DBMDZ). Then, I used GerPT2 (Minixhofer, 2020) and GPT2X (Minixhofer, Paischer, and Rekabsaz, 2022) in two sizes each. These models were initialized with different sizes of English GPT-2 and adapted to German. Finally, I used a pair of language models initialized from the small version of English GPT-2 from Stefan Schweter and the DBMDZ. One (GPT2-SM-50G) was trained using a 50G German training corpus and the other (GPT2-SM-90G) was trained using a 90G training corpus.

All models were freely available on the huggingface hub. I assert that perplexity differences within GPT families are significant and refer to significance testing in the original papers (Radford et al., 2019; Black et al., 2021; Zhang et al., 2022; Minixhofer, 2020; Minixhofer, Paischer, and Rekabsaz, 2022). Part of speech tags were packaged with Provo. I word tokenized the other corpora (ZuCo, PSC, English G-Maze) using the Natural Language Toolkit (Bird, Klein, and Loper, 2009), performed part of speech tagging using the Stanford tagger (Toutanova et al., 2003), and then manually detokenized, aligned, and corrected the output. I collapsed more fine-grained parts of speech into nouns, verbs, and other.

9.2 Coefficients for linear mixed effects models



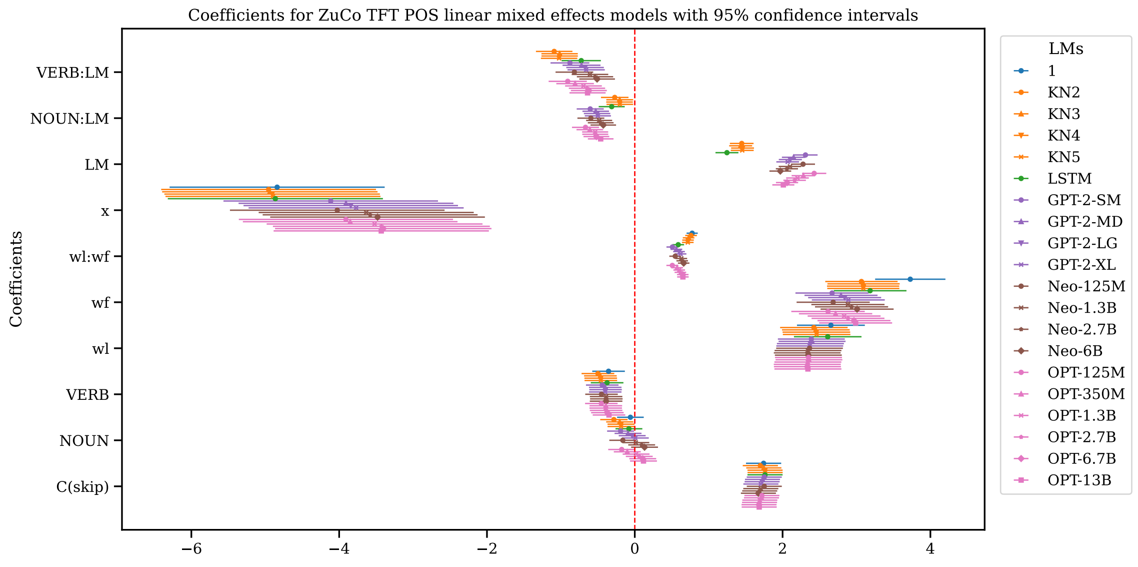
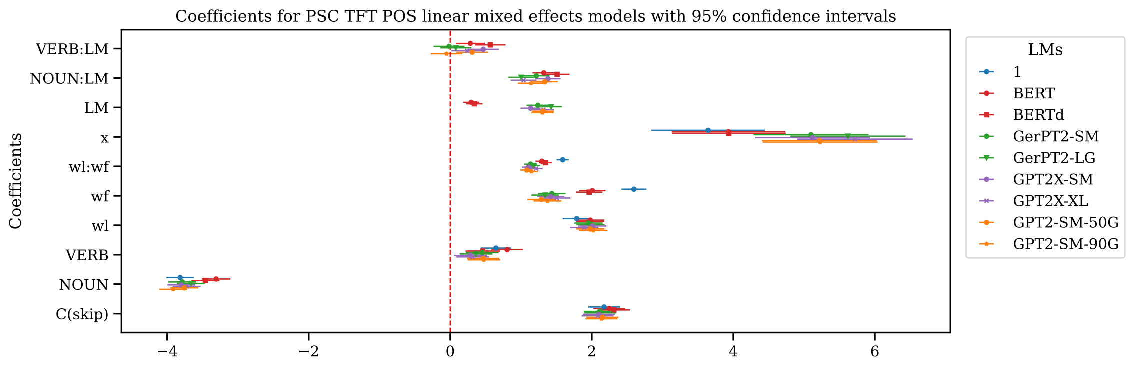


Figure 9.4: Coefficients for linear mixed effects models of TFTs in the Provo corpus (upper) and the ZuCo corpus (lower).

Figure 9.4 and Figure 9.5 show the coefficients for the "best" linear mixed effects models that I fitted on the reading behavior measures. By "best" these models for the most part had significant predictors and they all converged. For instance, I may have included an insignificant predictor for one corpus because it was significant for another, or I included it because it was involved in a significant interaction. I started with all baseline predictors from the literature and their two way interactions, but the word length : frequency interaction was the only one that was usually significant across language models and corpora. I explicitly tested previous



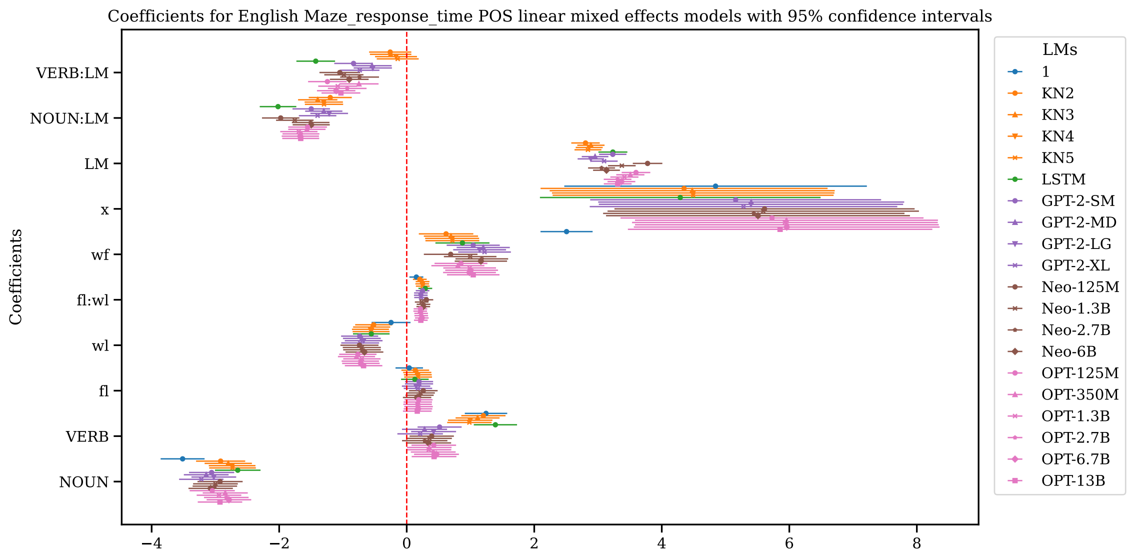


Figure 9.5: Coefficients for linear mixed effects models of TFTs in the PSC corpus (upper) and the English G-Maze corpus (lower).

word length, previous word frequency, and previous word surprisal for eye-tracking. These predictors often yielded coefficients that were not significant and often prevented the linear mixed effects models from converging, so I chose to exclude them. Also, these were not included in the models described in Oh and Schuler (2023) and I designed these experiments to be as comparable to theirs as possible.

Aside from the position in sentence variable (x) these coefficients were quite consistent across corpora and languages. Figure 9.4 and Figure 9.5 both show a trend in which as the size of the Transformer model grows, the coefficient learned for its surprisal becomes smaller. Goodkind and Bicknell (2018) did a similar analysis, and just like with mine, the distinctions were not significant.

For the G-Maze data, I included the length of the foil word (fl). I also explicitly tested the frequency of the foil word, and the surprisal of the foil word, but these led to insignificant coefficients and convergence failures.

9.3 G-Maze response time >TFT >FPRT >FFD

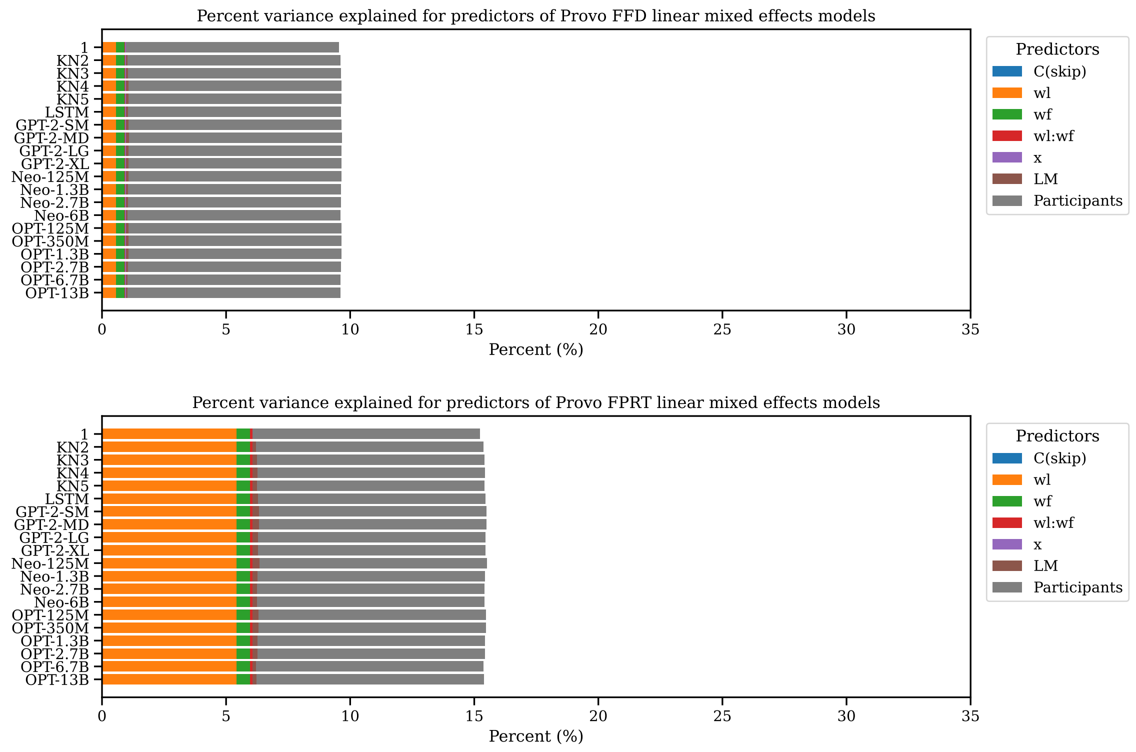


Figure 9.6: A comparison of the percent of variance that each predictor explains for models of FFD (upper) and FPRT (lower). The language model appears in brown and the random variation in the participants appears in grey.

Recall that I hypothesized that TFTs were the most appropriate of the three eye-tracking measures to compare against language model surprisals. However, as reviewed in Section 9.1.3, the literature very clearly favored working with FPRTs, perhaps because they are a middle ground between FFDs and TFTs. I built linear mixed effects models of all four behavior measures addressed in this chapter, computed analysis of variance tables, and explored how much variance the language models explained for the behavior measures.

As shown in Figure 9.6, Figure 9.7, and Table 9.2, The language models explained significantly more variance in G-Maze response times than in TFTs, they explained significantly more variance in TFTs than in FPRTs, and they explained significantly more variance in FPRTs than in FFDs. The differences are quite easy to see in the figures (informally, how big the brown-colored segments are) because the results are so strong. Namely, even for the best of the eye-tracking measures (TFT), the language models explain less than 1% of the variance. But they explain nearly 5% of the the G-Maze response times. Although not covered in the figures, I also tested the German eye-tracking dataset and its analogous percentages were similarly small and ordered the same way among the three behavior measures.

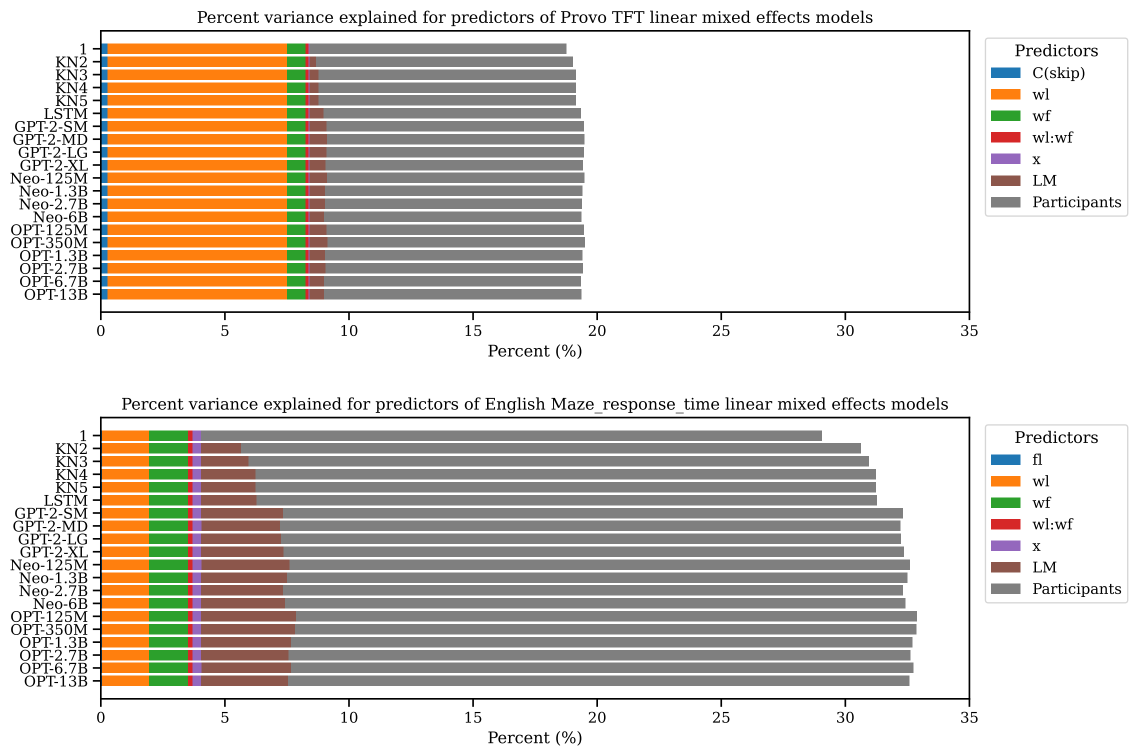


Figure 9.7: A comparison of the percent of variance that each predictor explains for models of TFT (upper) and G-Maze response time (lower). The language model appears in brown and the random variation in the participants appears in grey.

Measure 1	**Measure 2**	**Paired t-test**	**p-value**
Provo FPRTs	Provo FFDs	$t(18) = 19.3$	$p = 1.8 \cdot 10^{-13}$
Provo TFTs	Provo FPRTs	$t(18) = 14.0$	$p = 4.2 \cdot 10^{-11}$
G-Maze RTs	Provo TFTs	$t(18) = 19.2$	$p = 2.0 \cdot 10^{-13}$

Table 9.2: Hypothesis testing on whether language models explain *significantly* more variance in one measure than another.

I understand that conventional preferences for methodologies do not change quickly, but the results of my experiment are very important because they motivate a switch that could keep scientists from drawing conclusions based on significant, but problematically small effects. Yes, the best practices of hypothesis testing protect the rigor of scientific inference, but much meaning is still lost in the case that one finds that one number is significantly larger than another, by an essentially negligible amount. Despite the loss of comparability, this result motivates my choice to focus on G-Maze response times in later chapters. I also recommend using TFTs instead of FPRTs or FFDs for any future studies that relate eye-tracking and language models.

9.4 Dedicated handling of part of speech does not reduce the dissociation

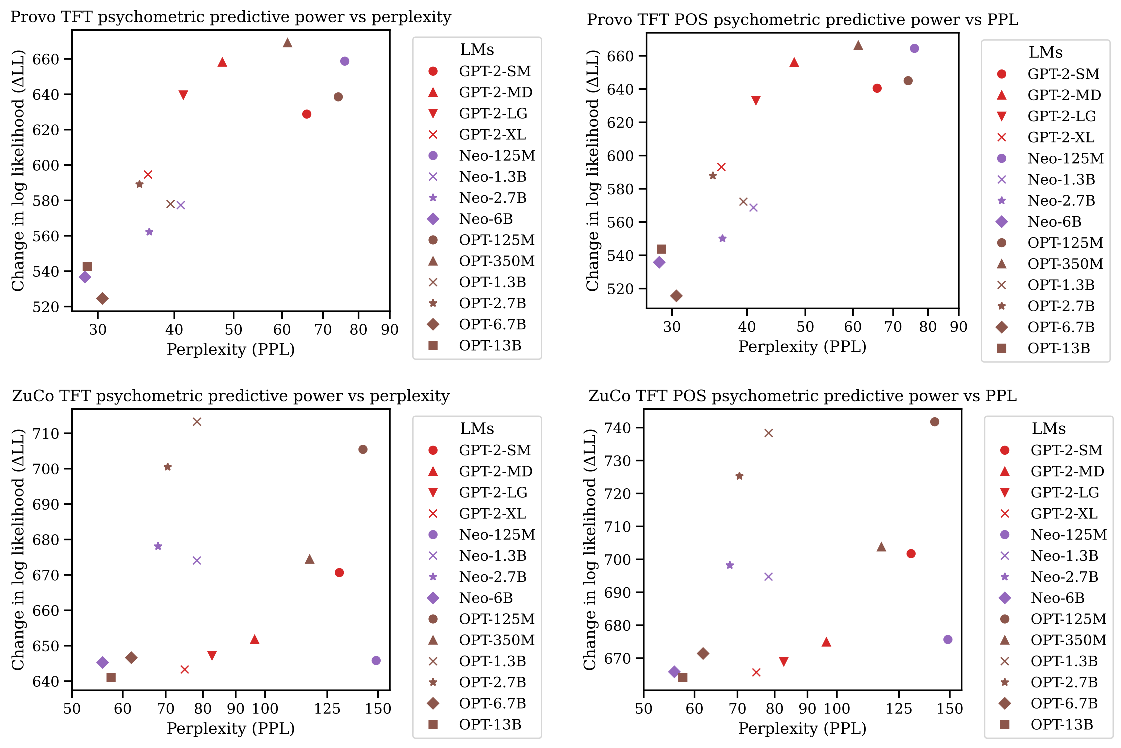


Figure 9.8: Scatterplots of psychometric predictive power versus perplexity. The right panels correspond to linear mixed effects models that incorporate categorical predictors and interactions for part of speech. The left panels correspond to linear mixed effects models that do not. The top panels are for the Provo corpus and the bottom panels are for the ZuCo corpus.

My results on Provo more than on ZuCo replicated the findings from Oh, Clark, and Schuler (2022). But it is noteworthy that with some GPT families, for example GPT-2 on ZuCo, the positive relationship occurred robustly.

As explained in Section 9.1.3, one of my novel contributions in this chapter is introducing categorical variables for part of speech into the linear mixed effects models. I hypothesized that if indeed the discrepancies between large and small models within the same family hinged on part of speech, then dedicated handling would lessen the effect.

However, the psychometric predictive power gap between the small models and the large models *increased* significantly (paired t-test $t(8) = 2.7$, $p = 0.028$) when I introduced dedicated handling of part of speech.

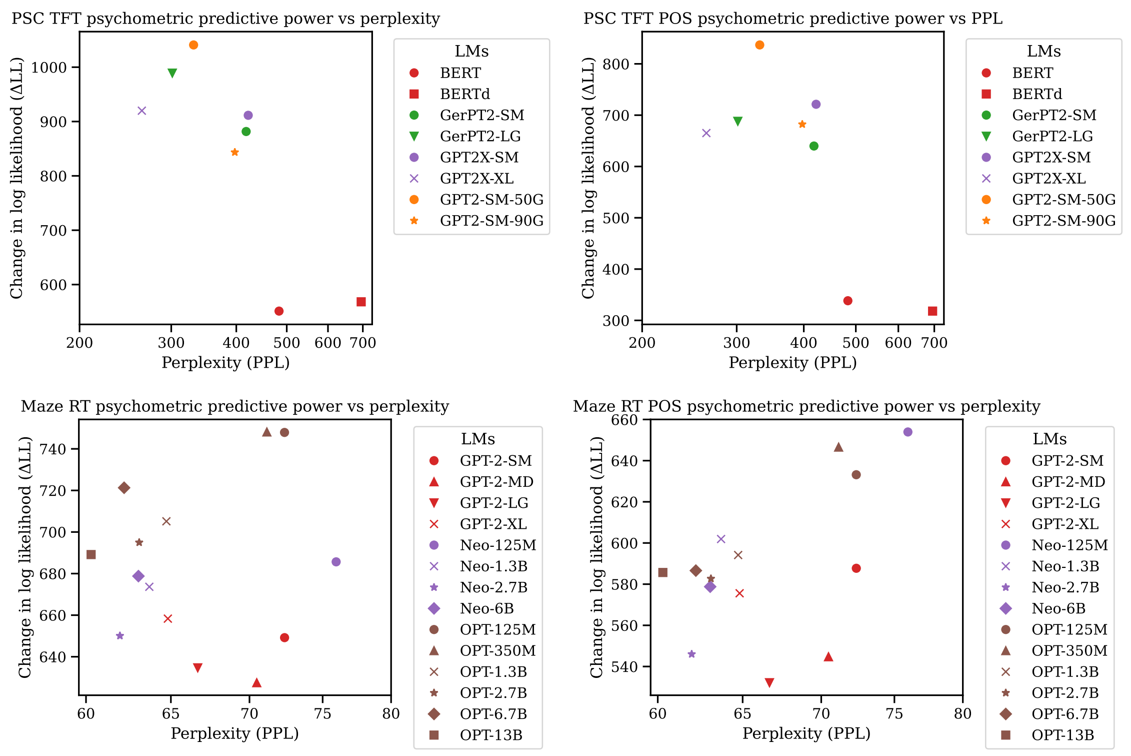


Figure 9.9: Scatterplots of psychometric predictive power versus perplexity. The right panels correspond to linear mixed effects models that incorporate categorical predictors and interactions for part of speech. The left panels correspond to linear mixed effects models that do not. The top panels are for the PSC corpus and the bottom panels are for the English G-Maze corpus.

LM Family	Provo		ZuCo		G-Maze	
	No-POS	POS	No-POS	POS	No-POS	POS
GPT-2	34	47	27	36	28	58
Neo	122	129	1	10	33	108
OPT	114	129	64	78	59	64

Table 9.3: These values give the positive difference between the "best" linear mixed effects model and the "worst" for different families of GPT models analyzing different corpora. The POS column corresponds to models that incorporate categorical predictors and interactions for part of speech.

The positive relationship between psychometric predictive power and perplexity was not really visible on PSC. However, I assert that this was due to the fact that there were no families of GPT language models for German available on the huggingface hub like there were for English. But I did find the positive relationship on English G-Maze. The gap also widened with dedicated handling of part of speech in this case, too.

9.5 Discussion and Conclusions

In this chapter, I explored the best predictors to include in linear mixed effects models for the purpose of analyzing the fit of language models to reading behavior measures. I found that language models explain more variance in G-Maze response times (a less popular measure) than in first pass reading times (the most popular measure). And, I showed a paradox for the relationship between psychometric predictive power and perplexity: the relationship is supposed to be positive due to different handling of nouns and verbs, but with dedicated predictors and interactions for part of speech, the effect grew more robust.

A substantial limitation of this work is its dependence on the proper inclusion of baseline predictors that correlate with reading times. Perhaps in part because of this, Hollenstein et al. (2021b) presented a contrasting way to evaluate Transformer models against eye-tracking data. Namely, they simply averaged over the participants and fine-tuned their language models to predict the reading times directly. I argue that the former method is preferable because it quantifies how much "external" information comes in via the baselines, it handles the participants separately as advocated by Demberg and Keller (2008), and Salicchi, Xiang, and Hsu (2022), the winner of the CMCL 2022 Shared Task on Multilingual and Crosslingual Prediction of Human Reading Behavior (Hollenstein et al., 2022a), used the former method as well. Lastly, Hollenstein et al. (2022b), a follow-up to Hollenstein et al. (2021b), claimed that fine-tuning to the eye-tracking data encodes something so divorced from surprisal that random initialization and initialization with state-of-the-art language model weights yield comparable predictions of FFD, FPRT, and TFT.

Chapter 10

Computational evidence for the divergence

Since in Chapter 9, I showed that language model (LM) surprisals explain more variance in G-Maze response times than they do in eye-tracking reading times, there was particular motivation to use G-Maze data to characterize further the relationship between perplexity, psychometric predictive power, and number of parameters of a language model. Upon inspecting the G-Maze dataset from Witzel, Witzel, and Forster (2012), most sentences were in pairs whose one-word difference illuminated an ambiguity or meaning change. For example, consider the following two versions of a sentence from this dataset:

(1) Amy will visit the man she worked with…
  a. …[last month]… (adverbial phrase)
  b. …[next month]… (adverbial phrase)
…, but she is nervous about it.

Since "visit" and "worked" are both verbs, an adverbial phrase in this position could, in theory, modify either one. However, in version (1a), the adverbial phrase "last month" must modify "worked", since "will" places the "visit" event in the future. In version (1b), the adverbial phrase "next month" must modify "visit", since "worked" is in the past tense. In linguistic terms, there is an attachment ambiguity, but the agreement features rule out one of the two interpretations in each case.

Having two sentences differ so little in form and so much in meaning is quite powerful for psycholinguistic analysis. But since the meanings of the sentences were manipulated to be different, there is not even an expectation that the surprisals for the respective words following the single-word difference between the sentences would be similar. If, instead, the sentences had essentially the same message realized with slightly different encodings, the differences in surprisal can be attributed to the encoding variation rather than a meaning distinction.

Since most foundational work in this area (e.g. Rayner and Well, 1996; Kliegl et al., 2004; Rayner et al., 2004; Drieghe, Rayner, and Pollatsek, 2005), manipulated meaning, it was not abundantly clear before the work that my coauthors and I presented in Sikos et al. (2017) that a pure encoding variation would be enough to manifest differences in human behavior. But since my coauthors and I found those differences, the data from that paper became an especially challenging test bed for evaluating humanness in language models.

Section 10.1 presents the novel dataset of G-Maze response times that my coauthors designed and collected, as described in Sikos et al. (2017). In particular, I give some minute details about how the data were collected, with an eye toward what might be important to consider when using language models to predict these measurements. While I was not involved in the data collection, I performed several language modeling experiments with this data, including the ones in the original publication. This chapter covers two such experiments. In Section 10.2.1, I describe how I curated a custom language model training corpus to accompany the Sikos et al. (2017) dataset. Then in Section 10.2.2, I show how I used this to compute highly accurate language model surprisals and illustrate the relationship between psychometric predictive power and perplexity for the German G-Maze data. I hypothesized that the relationship would be the same as in Chapter 9. But this ended up being only partially true, due to material differences among the language models evaluated.

In Section 10.2.3, I describe how I trained a small BERT model from scratch using only my custom language model training corpus. I evaluated changes in psychometric predictive power over the course of training this model. Accordingly, the training data and the size of the language model were constant throughout the experiment. Yet, psychometric predictive power fell significantly after the first 10 training epochs. This result supports my theory that optimizing for perplexity overfits the language model with respect to psychometric predictive power. Therefore, I *explain* the partial dissociation between language model quality (based on perplexity) and fit to psychometric data. I argue that optimizing for perplexity allows the language model to capture statistically valid but inhuman patterns, and a larger Transformer might be better at finding them.

10.1 A novel German G-Maze corpus

Consider the following examples:

(2)	The journalist published…	predictive context (`pred`)	
a.	…[the carefully written essay].	pre-nominal modification	(`pre`)
b.	…[the essay that was carefully written].	post-nominal modification	(`pos`)
(3)	The man evaluated…	non-predictive context (`non`)	
a.	…[the carefully written essay].	pre-nominal modification	(`pre`)
b.	…[the essay that was carefully written].	post-nominal modification	(`pos`)

The object of the main verb in all four examples is "essay" and all four examples say somehow that the essay was "carefully written". So, the so-called object noun phrase (the words

Context	Encoding	Example
Predictive	Post-nominal	Der Journalist veröffentlichte den **Essay**, der sorgfältig verfasst worden war, unter Einbeziehung des größeren Kontextes. “The journalist published the **essay** that was carefully written, taking into account the larger context.”
Predictive	Pre-nominal	Der Journalist veröffentlichte den sorgfältig verfassten **Essay** unter Einbeziehung des größeren Kontextes. “The journalist published the carefully written **essay**, taking into account the larger context.”
Non-predictive	Post-nominal	Der Mann bewertete den **Essay**, der sorgfältig verfasst worden war, unter Einbeziehung des größeren Kontextes. “The man evaluated the **essay** that was carefully written, taking into account the larger context.”
Non-predictive	Pre-nominal	Der Mann bewertete den sorgfältig verfassten **Essay** unter Einbeziehung des größeren Kontextes. “The man evaluated the carefully written **essay**, taking into account the larger context.”

Table 10.1: Example stimulus item in four conditions with approximate English translations. The object nouns are bolded and the modifier phrases are underlined.

within the brackets) in all four examples has arguably the same meaning. In (2a) / (3a) the modifier “carefully written” occurs before the nominal “essay” (pre-nominal modification) and in (2b) / (3b) the modifier “carefully written” occurs after the nominal “essay” (post-nominal modification). So, this alternation achieves the goal of having different encodings of the same meaning. While there is some meaning difference between (2) and (3), it is still arguably much less than the versions of (1). My coauthors crafted (2) such that the subject and verb of the sentence constitute, by semantic association, an especially likely context for the object (predictive context). In contrast, the subject and verb in (3) are supposed to be more neutral, such that the object does not get as likely of a context (non-predictive context).

Materials My coauthors constructed 48 sets of sentences, in German, following a paradigm that “crosses” the context and the modifier encoding. By crossing these two experimental manipulations, all four possible combinations are obtained. Table 10.1 gives the context, encoding of the modifiers, the German sentence, and the English translation for the sentences corresponding to the examples that I introduced before. Importantly, my coauthors avoided choosing highly expected object nouns (e.g., *Artikel*, “article”) to increase the possibility of detecting behavior differences between the predictive / pre-nominal and predictive / post-nominal conditions.

Four counterbalanced lists were constructed from these materials according to a Latin Square design such that each list contained 12 items in each condition, but no item appeared more than once in the same list. An additional 48 sentences with the same structures as above, but containing highly predictable object nouns, were constructed as fillers (e.g., *Der Schneider zerschnitt den stark gemusterten Stoff am Mittwoch.*, “The tailor cut the heavily patterned fabric on Wednesday.”). Half of the filler sentences contained pre-nominal modification of the object noun and the other half contained post-nominal modification. No object nouns were repeated across experimental or filler items.

Cloze probability and contextual constraint An offline cloze completion study was conducted to confirm that object nouns were more expected following predictive than non-predictive contexts, but were not highly expected in either context. A group of 58 native German speakers (age: $\mu = 22.0$, $\sigma = 2.9$) were presented with sentence fragments from the 48 experimental items described above. Fragments contained only the contexts, followed by a blank (e.g., *Der Mann bewertete* ____; *Der Journalist veröffentlichte* ____). Predictive and non-predictive contexts were counter-balanced across two lists. Participants were asked to fill in the blank with the first determiner–noun combination that came to mind. Cloze probabilities were computed as the percentage of participants who provided the experimental object noun for a particular item. As expected, object nouns had low cloze probabilities in both contexts but were reliably more expected following predictive (cloze: $\mu = 0.06$, $\sigma = 0.18$) than non-predictive contexts (cloze: $\mu = 0.00$, $\sigma = 0.01$), $t(47) = -2.32$, $p < .05$.

The percentage of the most frequently occurring response to each sentence fragment in the cloze test was also used to assess the contextual constraint of predictive and nonpredictive contexts. As expected, the mean constraint of predictive contexts was reliably greater (51.3%) than that of the non-predictive contexts (21.3%), $t(47) = -8.46, p < .001$.

Participants A separate group of 27 native German speakers (age: $\mu = 24.0$, $\sigma = 2.6$) with normal or corrected to normal vision and recruited from the Saarland University community navigated G-Mazes based on the materials described before. They were compensated 8€ for their participation. The three participants who did not successfully navigate at least 70% of G-Mazes in all experimental conditions were excluded. Each participant only saw one of the four context / encoding condition for a given item (sentence set).

Procedure Participants were randomly assigned to a stimulus list (6 per list). The G-Maze task was implemented in E-prime (Schneider, Eschman, and Zuccolotto, 2002). Each trial began with two crosses (+) that remained on screen for 1000 ms, indicating where subsequent word pairs would appear. Each word in the sentence (except the first word) was then presented together with a foil word,[1] which was not a grammatical continuation of the sentence. The first word in every sentence was paired with "x-x-x". The presentation side (left, right) was randomized such that the correct word appeared equally often on each side. Any punctuation (i.e., comma, period) that appeared with a word also appeared with its foil. Participants were instructed to choose as quickly and as accurately as possible the word that best continued the sentence. Participants indicated their selection by pressing the left or

[1]Foils were created in a two-stage process. First, a custom Python script randomly selected a foil candidate for each word in each experimental and filler item. Foil candidates were constrained such that they did not appear in bigrams with the correct word at the previous position in the sentence within a large German corpus. Second, each foil was then hand checked by at least two trained native-German linguists to ensure that it was not a grammatical continuation of the sentence. The same foil was used for identical words (or derivationally related words) across conditions.

right button on a button box and the amount of time required for selecting the grammatical continuation was recorded as the response time (RT). If the correct word was chosen, the next pair of words appeared automatically. However, if a foil word was selected, negative feedback (*Inkorrekt*, “Incorrect”) was displayed and the trial was aborted. Once the end of a sentence was reached, positive feedback (*Korrect*, “Correct”) was given. Participants initiated each new trial by button press.

To confirm that participants read the sentences for meaning, a Yes / No comprehension question appeared after one-third of the items. Half of the questions asked about the subject noun and half about the object noun. The correct answer was Yes for 50% of questions. Participants used the button box to respond. No feedback was given.

In order to familiarize participants with the task, five practice items (three with comprehension questions) were presented before the experiment began. Participants took approximately 40 minutes to complete the experiment.

Completed G-Mazes Overall performance on the G-Maze task was high, with participants successfully navigating 85.6% ($\sigma = 0.08$) of experimental and filler items to completion. However, because the critical region of interest was the object NP, the RT analyses reported below were conducted on all experimental items that were completed through at least the end of the critical region ($\mu = 0.90$, $\sigma = 0.06$).

Comprehension question accuracy Performance on the comprehension questions was near ceiling ($\mu = 0.97$, $\sigma = 0.04$), confirming that participants were reading the sentences for meaning during the G-Maze task.

10.2 Psychometric predictive power and perplexity

In this section, I give a partial replication of some results from Chapter 9 using the German G-Maze dataset and my accompanying language model training corpus. I hypothesized that there is still generally a negative relationship between psychometric predictive power and perplexity overall, but this relationship turns positive when comparing language models of different sizes within the same family. These experiments on the German G-Maze data provide a pivotal refinement to the result that increased Transformer size hurts psychometric predictive power. Namely, the families of German Transformer language models were not built the same way that the English ones were, and only one of them replicates the English result. In the remainder of this section, I will describe how I made the custom training corpus, how the German language models were made, how I constructed the linear mixed effects models to predict the G-Maze response times, and what these intermediate results show for the relationship between psychometric predictive power and perplexity.

10.2.1 A custom language modeling corpus

To ensure that the language models that I trained had the highest possible psychometric predictive power, I curated a novel, custom training corpus with knowledge of the German G-Maze data. Like WikiText-103 (Merity et al., 2016), my corpus was about 100MB and based on Wikipedia. Since my corpus is in German, I refer to it in this work as the dewiki corpus. To make the dewiki corpus, I started with a January 1, 2017 dump of German Wikipedia, filtered the original XML dump using the tool WikiExtractor[2], split the corpus into sentences using the NLTK sentence splitter for German (Bird, Klein, and Loper, 2009), and finally, preprocessed the resulting dataset. The preprocessing that I chose included lowercasing, replacing punctuation with space, replacing digits with NUM, removing empty lines, replacing tabs with spaces, removing multiple spaces, removing multiple NUMs, replacing umlauts by their conventional character bigrams as shown in Table 10.2, and adding sentence begin and end markers. Note that some of these choices, most notably the lowercasing and the character bigrams were ideal for the current "small language model" setting, but not so for training or fine-tuning a large language model. Specifically, this preprocessing maximizes the chance that alternate forms of a word (e.g. with and without umlauts) are considered together during training, but collapses distinctions that a large language model can use to make its predictions more precise.

Original character	Replacement bigram
ä	ae
ö	oe
ü	ue
ß	ss

Table 10.2: Four special characters in German and their replacements in the dewiki corpus.

At this point the corpus was still around 4GB. To reduce the size, I compiled a list of words in the German G-Maze data that occurred below a certain frequency threshold and selected only those sentences that contained at least three words overall and one word from the "words of interest" list. Finally, by converting every type that occurred fewer than 15 times to "<unk>", I restricted the vocabulary to 47,440 (down from 833,734). This, too, was essential for a relatively quick language model training process.

The technique of using a "words of interest" list is somewhat nonstandard because once individual sentences are extracted from their Wikipedia articles, there is no context beyond a single sentence. Recall that GPT language models can encode 1024 or 2048 tokens at one time. And, Chapter 8 showed that perplexity vastly improves when the model has access to full-size contexts. However, I argue that this was the right choice for the present experiments because it offsets the deliberate choice that my coauthors made to select uncommon

[2]`https://github.com/attardi/wikiextractor`

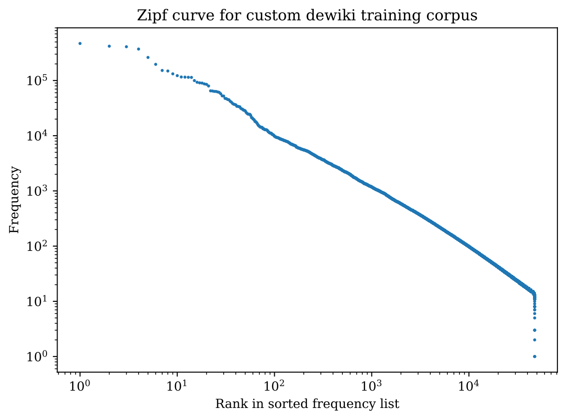


Figure 10.1: Zipf curve for the dewiki corpus.

words in German. In essence, my "words of interest" list ensured that whatever information Wikipedia contained on these uncommon words ended up in my custom corpus. Also, recall that most of the ZuCo corpus is also individually extracted sentences from Wikipedia. One main argument against using this technique is that it could, in theory, result in an unnatural distribution of tokens in the corpus. So, I will present one statistical analysis to show that this did not happen. Namely, a fundamental result from statistical natural language processing is Zipf's Law (Zipf, 1936, 1949), which states that there is a precise exponential relationship between the number of times a type occurs (frequency) and its position (rank) in a frequency ordered list. For example, since "the" is usually the most common type in an English text, its position in the frequency ordered list is 1. "a" might have position 2, and all of the types that occurred only once (predicted by Zipf's Law to be about half of the types in the text) would appear at the end of the frequency ordered list. If the frequency and rank are both log transformed, the relationship appears linear. Figure 10.1 gives the Zipf curve for the dewiki corpus after restricting to sentences with at least one word of interest. The curve is fully as expected. The bottom part of the curve changes shape as it does because I computed the Zipf curve on just 80% of the data. The other 20% (reserved for testing and evaluating the models only) contains the missing occurrences of those least frequent words such that their total frequency is 15.

10.2.2 Results for *n*-gram, RNN, and pre-trained Transformers

The most important pair of language models that I evaluate in this section came from Stefan Schweter and the Digitale Bibliothek Münchener Digitalisierungs Zentrum (DBMDZ), made available on the huggingface hub. These language models were both initialized using the small version of *English* GPT-2. Then, one (GPT2-SM-50G) was trained using a 50G German training corpus and the other (GPT2-SM-90G) was trained using a 90G training corpus. As such, these models, with the same architecture and different training data, are not exactly siblings in the way the models were in Chapter 9, since those models had different architectures and the same training data. Indeed, this is closer to the experimental setup in Wilcox et al. (2020), but those were not initialized with an even larger corpus.

I also evaluated GerPT2 (Minixhofer, 2020) and GPT2X (Minixhofer, Paischer, and Rekabsaz, 2022) in two sizes each. These models further developed the idea of using cross-lingual information. GerPT2 was based on English GPT-2 small and large, while GPT2X was based on English GPT-2 small and extra-large. These families used the same training data within their families, but the cross-lingual transfer meant that the different sizes of these models started their specialization to German from quite different initial states.

Additionally, I trained modified interpolated Kneser-Ney n-gram models ($n = 2$ to $n = 5$) using KenLM (Heafield, 2011; Heafield et al., 2013), a recurrent neural network LM (RNN) (Mikolov et al., 2011c), a long short-term memory (LSTM) LM (Sundermeyer, Schlüter, and Ney, 2012), LSTM with dropout (LSTMd), and LSRC (Oualil et al., 2016b) all using my custom training corpus dewiki. I used the version of German BERT native to huggingface (pre-trained), and I also used another pre-trained one from DBMDZ (BERTd). As in previous chapters, I computed perplexity using the unmodified BERT models and psychometric predictive power using versions for which I masked all tokens beyond the target token.

I obtained surprisals for the German G-Maze corpus, including the foil words in context, using my code base described in Chapter 8. Then, I performed part of speech tagging using the Stanford tagger (Toutanova et al., 2003) and manually corrected the tags. Importantly, I retagged pronouns, which were quite rare in this dataset, as nouns. Finally, I fit linear mixed effects models (mlms) on the surprisals to predict the G-Maze response times. As with the mlms from Chapter 9, I included baseline predictors for whether the true (non-foil) word was a noun (NOUN), whether the true word was a verb (VERB), the length of the true word (wl), the length of the foil word (fl), the length interaction (wl:fl), the frequency of the true word (wf), and the position in the sentence (x). Also like in Chapter 9, I allowed the noun status and the verb status to interact with the language model surprisal (NOUN:LM and VERB:LM). The frequency of the foil word prevented the mlms from converging, and even the converged mlms usually did not assign a significant coefficients to it. I included random intercepts for the participants and items[3], and I included random slopes for wl, fl, and x. Further random effects prevented the mlms from converging.

[3] A sentence in its four variants consituted one item.

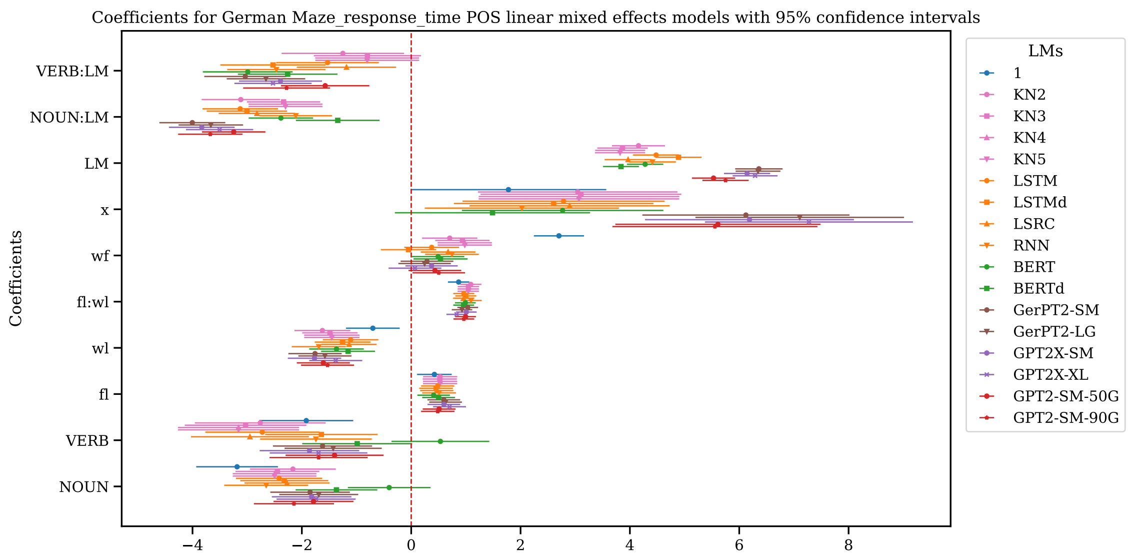


Figure 10.2: Coefficients for linear mixed effects models of G-Maze response times in the German G-Maze corpus.

Figure 10.2 shows the fitted coefficients for the mlms, which all converged. Note that the main coefficient for GPT2-SM-90G is slightly larger than that for GPT2-SM-50G, but the negative coefficients are also stronger for the part of speech and language model interaction for GPT2-SM-90G than for GPT2-SM-50G. This means that dedicated handling of nouns and verbs successfully brought the predictions closer together for the two language models in that family, something that explains some results from Oh and Schuler (2023), but that I did not find evidence for in other datasets. Recall that in Chapter 9, the language model coefficient was smaller for the larger language model in the family, and the interaction terms were mixed. Consistently with the English G-Maze dataset, however, nouns had mostly negative coefficients, meaning that their processing seems to have been facilitated by something not covered by the language model. Indeed, this seems to be the case as well for verbs in the German G-Maze data. In general, this does not support the idea from Oh and Schuler (2023) that the language model, especially a larger Transformer, underestimates the surprisals for nouns and verbs.

Also as with the English G-Maze data, the coefficients for position in the sentence were positive, meaning that the human participants became slower as they progressed through the sentence, but the language models did not get more surprised. This is to be expected since the G-Maze task requires memory and there is more of the sentence to remember as the sentence continues. The foil length and the length interaction coefficients were positive, meaning that longer foils took longer to process as expected and that the human participants needed disproportionately more time to make their response when the true word and the foil were both very long. After accounting for these two variables, the true word length coefficients were negative, which also happened on the English G-Maze dataset.

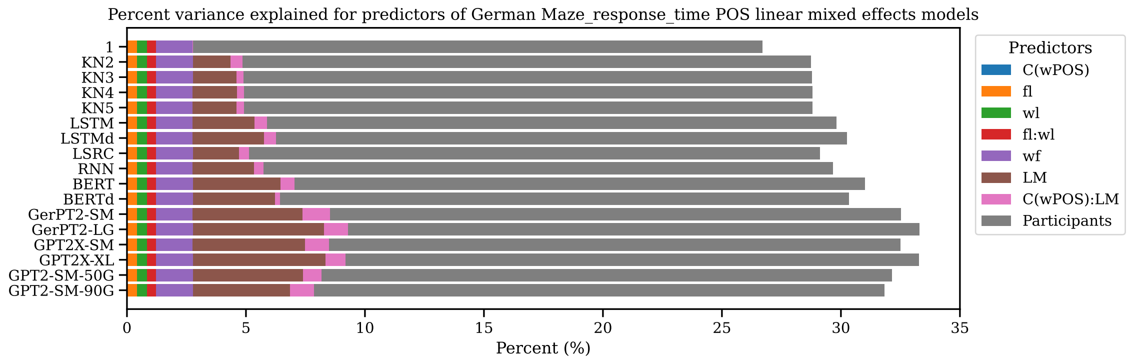


Figure 10.3: A comparison of the percent of variance that each predictor explains for models of G-Maze response time. The language model appears in brown and the random variation in the participants appears in grey.

As an especially nice result, Figure 10.3 shows that the language model surprisals explained clearly more than 5% of the variance in the G-Maze response times, independently of word length, part of speech, and variation among human participants. Further, the GPT2-SM model trained on less data explained more variance than the model trained on more data, although this difference lessened after considering the interactions between parts of speech and the language models.

Surprisingly, though, the opposite was true for the GPT2X models. I attribute this to the possibility that the larger GPT2X model received more good cross-lingual information from English, and this overpowered the negative effects relating to larger Transformer sizes.

The psychometric predictive power results shown in Figure 10.4 confirm the trends observed in the variance explained results. Recall that if psychometric predictive power and perplexity are in agreement, then the data should be negatively sloped. Indeed, the recurrent neural networks (orange), BERT models (green), and the two GPT families based on cross-lingual information (brown and purple) all suggest that improvements in perplexity were accompanied by improvements in humanness. Only with the two GPT2-SM models (red) is there a positive slope. The difference of 26 nats between the log likelihoods for these two mlms was clearly significant ($p = 5.6 \cdot 10^{-13}$). The n-gram models had a normal perplexity trend (perplexity improves as n increases) but these gains did not correspond to improvements in humanness. Overall, these results are most reminiscent of the Wilcox et al. (2020) results in that there was little pattern among different architecture families, but within architecture families, the results are quite consistent.

It is important to remember that these results do not directly conflict with those from Chapter 9 because the language models within a family were related in different ways. There may not be enough language model families to draw scientific conclusions, but it is reasonable to conjecture that the nature of the statistical properties learned by the language model modulates the humanness. The language model might learn statistical properties that are human and it might learn statistical properties that are inhuman.

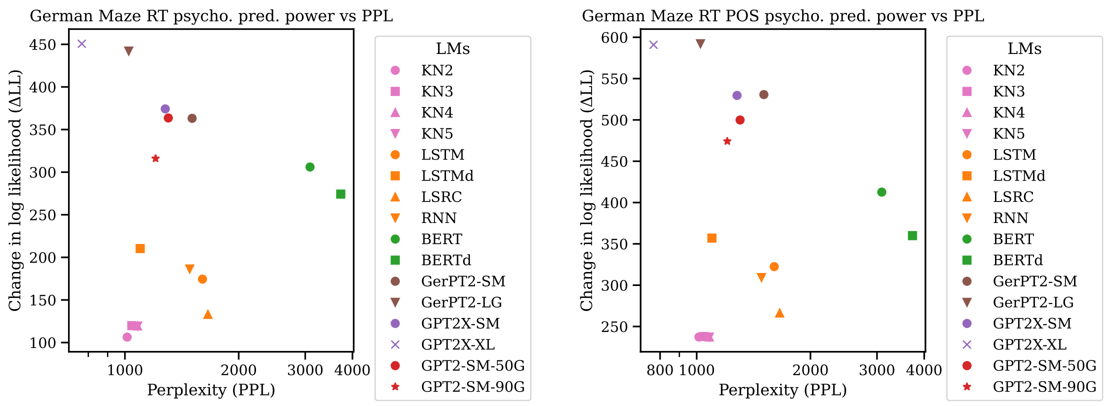


Figure 10.4: Scatterplots of psychometric predictive power versus perplexity. The right panel corresponds to linear mixed effects models that incorporate categorical predictors and interactions for part of speech. The left panel corresponds to linear mixed effects models that do not.

Note that the n-gram models had quite low perplexities. Indeed, they did not outperform the recurrent neural neural networks and the Transformer models in Chapter 8 and Chapter 9. This was probably due to my custom training corpus. The custom training corpus was designed to contain the rare words that were in the G-Maze dataset. In general, Figure 10.4 does not show a strong relationship between psychometric predictive power and perplexity. I needed a more constrained experiment to draw clearer insights.

10.2.3 A BERT model trained from scratch

Recall that in Chapter 9, I discussed that fine-tuning to predict reading times would be at odds with the goal of evaluating the humanness of language models trained in the usual way, i.e. minimizing perplexity. Given that the perplexities of the large GPT-2 models presented in the previous section were already lower than those of the n-gram models, it seemed unlikely that fine-tuning on my custom corpus would yield qualitatively different results.

However, the feasibility of training a BERT model fully from scratch on the custom training corpus (as opposed to a GPT model, which would generally require a much larger training corpus) presented a great opportunity to explore psychometric predictive power and perplexity *over the course of training*. Because I control all aspects of the training, I can guarantee that the architecture, the training data, random initializations, and hyperparameters are all held constant. Therefore, the changes in psychometric predictive power and perplexity observed during training are much more clearly attributable to robustness of patterns in the training data.

Accidents in the training data would not appear in the test set, so perplexity would go up or stay the same. Statistically robust but inhuman patterns in the training data would make the perplexity and psychometric predictive power fall simultaneously. Human patterns not

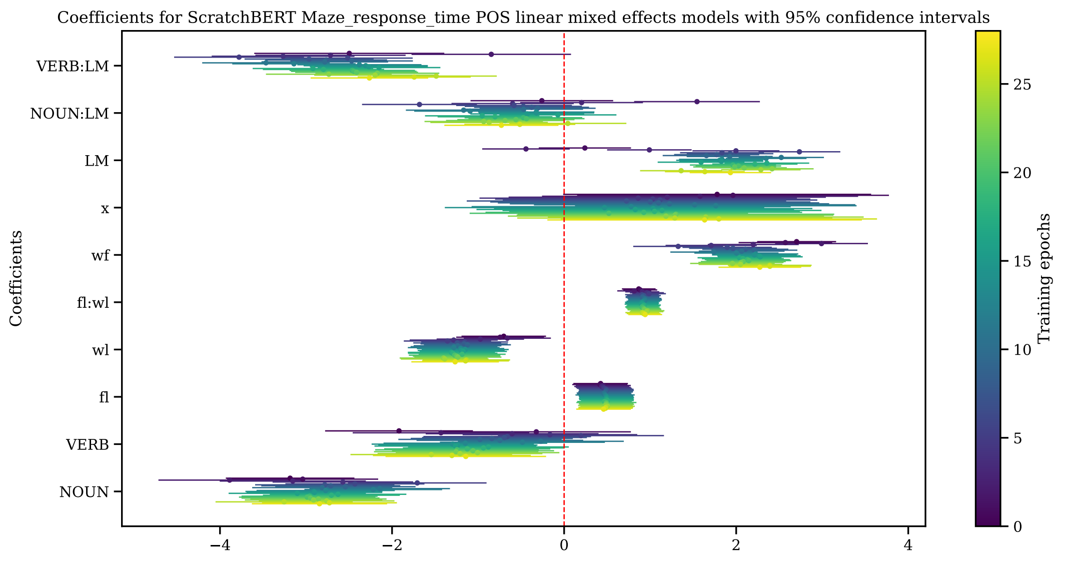


Figure 10.5: Coefficients for linear mixed effects models of G-Maze response times in the German G-Maze corpus (ScratchBERT).

present in the training data could not be captured during training since the model would not have access to them. And, of course, statistically robust *and* human patterns would make the perplexity fall and the psychometric predictive power rise. I hypothesized that at some point rather early in the training process, the network would cease to discover statistically robust human patterns and only discover inhuman ones. Further, this point should occur well before the network stops discovering any statistically robust patterns, yielding one LM that is best for psychometric predictive power and a materially different one that is best for perplexity.

I followed a recipe[4] by Julien Chaumond from huggingface (Wolf et al., 2020) to train this small BERT model, which I call ScratchBERT. Its architecture was most similar to RoBERTa (Liu et al., 2019). The model had 83,504,416 parameters: 6 layers, a hidden layer size of 768, and 12 attention heads. The model considered sequences up to 512 words long, plus start / end tags. Like GPT-2, it used a byte pair encoding tokenizer with size 52,000 (just larger than the vocabulary of my custom corpus).

When each training epoch completed, I saved the model, computed surprisals on the German G-Maze dataset, and fit an mlm over those surprisals to predict the G-Maze response times. The structures of the mlms were the same as those described in Section 10.2.2. The coefficients for these mlms are presented in Figure 10.5. Again, all mlms converged. Note that the coefficients for the LM surprisals are significantly positive, which shows that the language models learned significantly useful information for predicting the response times beyond the baseline predictors. This lends validity to the experiment.

[4]`https://huggingface.co/blog/how-to-train`

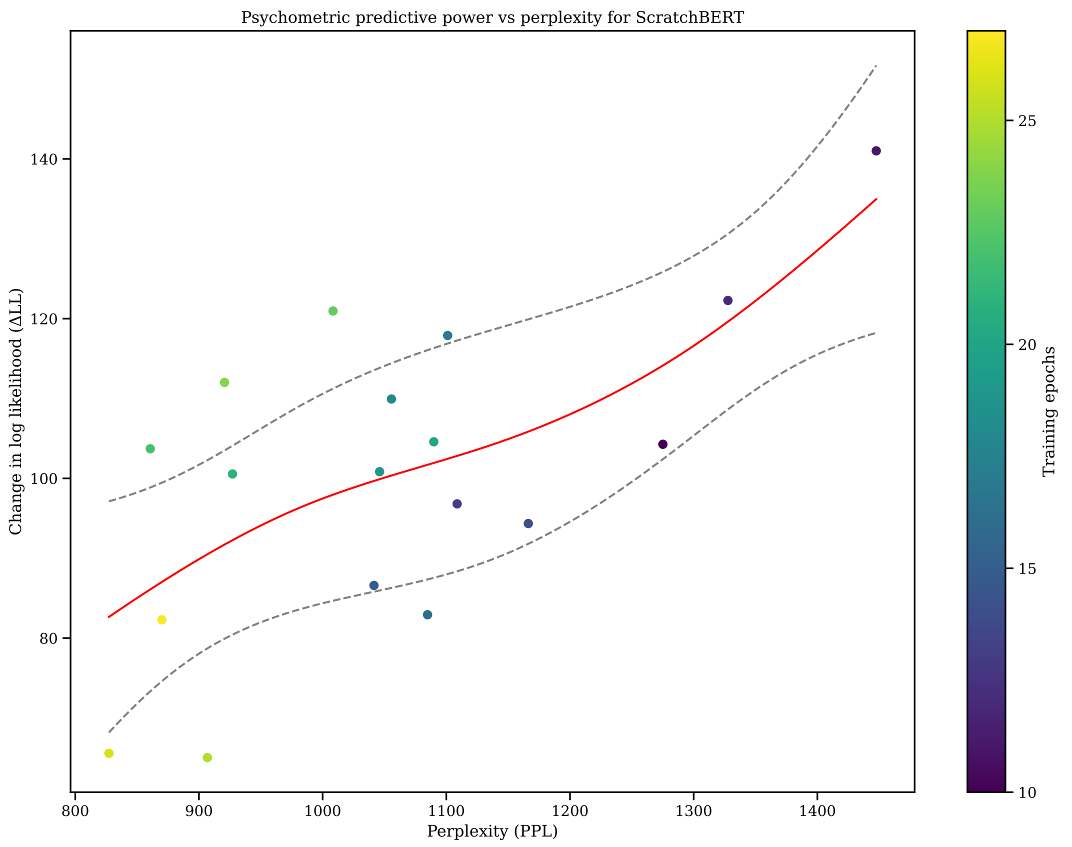


Figure 10.6: A scatterplot of psychometric predictive power versus perplexity. Each point represents a checkpoint during training of ScratchBERT. The fitted curve appears in red and the 95% confidence interval appears in dashed grey.

Figure 10.6 presents the main result from this experiment. Starting with the 10th epoch, psychometric predictive power and perplexity mostly fell together. The difference of 22 nats between the log likelihoods for the epoch 10 mlm and the epoch 29 mlm was clearly significant ($p = 3.3 \cdot 10^{-11}$). Psychometric predictive power and perplexity were positively correlated (Spearman $\rho = 0.52$, $p = 0.027$). I also fitted a Generalized Additive Model (GAM) to explore the best fit to the data allowing for nonlinear relationships. I fitted the GAM with 7 splines (each spline adds more power to the model) as this was the highest number of splines for which the output values remained significant. The curve of best fit from the GAM appears in red in Figure 10.6, as well as a 95% confidence interval (the grey dashed curves). Critically, the top left of the confidence interval is much lower than bottom right, meaning that there is at least 95% confidence that the true curve is increasing.

As such, this is quite clean and compelling evidence that the language model achieved better perplexities over the course of its training *at the expense of* psychometric predictive power.

10.3 Discussion and Conclusions

This chapter introduces a novel dataset of German G-Maze response times that my colleagues collected and I analyzed as part of a collaboration. I made a custom training corpus for the dataset, produced a wide range of word-by-word language model surprisals to compare against the G-Maze response times, analyzed patterns for the relationship between psychometric predictive power and perplexity on the new dataset, and then finally, by training a BERT model on my training corpus from scratch, observed that psychometric predictive power and perplexity fell together during training.

My power of inference with the ScratchBERT results is rather limited. Minimally, it will take further research to apply this theory of the language model divergence to GPT models rather than the BERT model that I trained. Also, the results with the pre-trained models call into question what it means for language models to be in the same family. Changing the size of the model changes what patterns it can and will capture from the training data. In this light, it is understandable that the pair of German language models, although trained on different data, behaved similarly to a family of English GPT models.

However, I can still claim that these experiments are good computational evidence for a language model divergence at least for the specific models explored. There is one model that is best in terms of perplexity and a different one is best in terms of psychometric predictive power.

Chapter 11

Linguistic evidence for the divergence

By inspecting a small BERT language model at different epochs in its training, I found that the model was achieving gains in perplexity at the expense of psychometric predictive power. This was compelling computational evidence for the language model divergence.

However, there was perhaps somewhat of a missed opportunity with respect the human behavior measure that I used for this experiment. Namely, the dataset was constructed with items as sets of four sentences with similar meanings. Indeed, my coauthors' original motivation for collecting this data was to explore why language producers would elect one encoding over another. As such, these G-Maze response times contain robust patterns among the four encoding choices.

In this chapter, I explain several statistical properties of the data that my coauthors collected. Some of these properties were discovered jointly, and some I discovered on my own after the official collaboration ended. From these properties, I developed a simple computer-based Maze task and a suite of 19 "tests" based on significant patterns in the G-Maze response times. I ran these tests on the language model surprisals, hypothesizing that due to the language model divergence, the test results would align more closely to the psychometric predictive power values than the perplexity values. For the 19 test suite, this is exactly what happened, which gives linguistic motivation and explanation for the language model divergence.

11.1 Background

In this section, I first review the literature for constructing syntactic tests for evaluating language models. Then, I explain the properties of the G-Maze response times that my suite of tests looks for. On the surface, my tests are wholly innovative, but they do rely on an idea that fueled the existing literature: assigning a low surprisal to a word can be interpreted as the model *choosing* that word.

11.1.1 A review of syntactic tests for language model evaluation

Linzen, Dupoux, and Goldberg (2016) was one of the first works that compared language model predictions against human judgements of grammaticality (acceptability). They tested subject-verb agreement because they argued that it was an especially clear case for a task that needs knowledge of linguistic hierarchy. They designed four tasks with increasing difficulty. For the easiest task, they trained binary classifiers to take in the beginning of a sentence and output whether the verb should be singular or plural. For the second task, they trained binary classifiers to take in a complete sentence and output whether it had proper subject-verb agreement. For the third task, they trained the model to predict the correct inflected form for a verb in context given the base form. For the last task, they trained normal language models, computed the surprisal for both grammatical numbers of a verb in context, and counted the item as correct if the correct version of the word had lower surprisal than the incorrect version. The first three tasks had good accuracy, but the fourth task did not.

In their analysis, Linzen, Dupoux, and Goldberg (2016) noted that for the verb inflection task, the system could in principle rule out intervening nouns if they do not fit thematically with the verb. Gulordava et al. (2018) also argued this. They used the example sentence "dogs in the neighborhood often bark". "dogs" as the agent of "bark" has high thematic fit, while "neighborhood" has low thematic fit in that role. So, in theory, the system could use this information (just as a parser or semantic role labeler would) to "guess" the correct subject, rather than apply hierarchical knowledge to determine it. So, Gulordava et al. (2018) extended the agreement tests to "meaningless" sentences, such as Noam Chomsky's famous "Colorless green ideas sleep furiously." They generated such "nonce" sentences by substituting out all content words with other content words that preserve the inflection. They further extended the work to three languages other than English, they did not fine-tune to the agreement task, and even replicated Linzen, Dupoux, and Goldberg (2016) using a better language model than the original work had. They found that LSTMs trained on the usual objective perform consistently well on the task across all four languages "not far from human performance in Italian." They correlated the perplexities of the 68 models per language they obtained during a hyperparameter search against the agreement accuracies. The Pearson correlations ranged from -0.55 (Hebrew) to -0.78 for English, which was the expected direction. Lastly, they argued that the number of "attractors", nouns with the other grammatical number that intervene between the true subject and verb, most dramatically reduces performance on this task by both humans and language models.

Kuncoro et al. (2018) trained an LSTM with no syntactic knowledge, an LSTM that had access to parse trees, and an LSTM that explicitly composes representations according to the parse tree. Only the latter displayed a benefit for their subject-verb agreement task. Since the compositional model had worse perplexity, they claimed that they found evidence of a dissociation between perplexity and a more human notion of language model quality. They wrote, "while perplexity can be a useful diagnostic tool, it may not be sensitive enough for comparing models in terms of how well they capture grammatical intuitions."

Tran, Bisazza, and Monz (2018) did a simple experiment in which they compared an LSTM against an early Transformer model on the tasks of logical inference and subject-verb agreement using data from Linzen, Dupoux, and Goldberg (2016). Their LSTM had a perplexity of 67 and their Transformer had a perplexity of 69. Despite being just slightly better in terms of perplexity, the LSTM was much better at subject-verb agreement and logical inference. So, they, too, claimed that their results were evidence of a dissociation.

The unique contribution of Marvin and Linzen (2018) is that their dataset of syntactic tests comprises pairs of sentences that "differ minimally from each other" (this is in the spirit but distinct from the formal linguistic notion of minimal pair). For each pair, one sentence was grammatical and the other was not. The language model computed the total surprisal for both sentences and was judged correct if the grammatical sentence had lower surprisal. They tested subject-verb agreement, reflexive anaphora, and negative polarity items. They found that as perplexity decreased, the performance of the models on the task increased. However, Marvin and Linzen (2018) claimed that this was because "each model was conditioned on richer information than the previous one" rather than just merely perplexity.

Linzen and Leonard (2018) characterized subject-verb agreement errors made by humans and by RNNs. They found two main divergences. First, attractors within relative clauses caused more errors for the RNNs than attractors within prepositional phrases, while humans had the opposite behavior. They argued that the humans leveraged the comparatively larger amount of implicit syntactic structure in a relative clause to make the correct agreement decision. Second, RNNs exhibited a cumulative effect from all attractors and seemed to consider distance between the subject and the attractors, while humans only showed an effect based on the distance between the most recent attractor and the verb. The authors proposed a "faulty" RNN heuristic that expects relative clauses to be short.

Ettinger (2020) obtained cloze and N400 data and compared these against completions supplied by BERT. They reasoned that if there was a divergence between the N400 and cloze profiles, this completion was especially challenging or interesting from a psycholinguistic perspective. This allowed them to create a number of psycholinguistically-inspired "tests". The first dataset of completions was designed to test "commonsense and pragmatic inference". They tested sensitivity by exploring differences within the "appropriate" semantic category. BERT did decently well on these tests. The second dataset of completions was designed to test "event knowledge and semantic role interpretation". They tested sensitivity by analyzing the effect of reversing word order of the role-fillers, which often reverses the roles and appropriateness of the completions. Again, BERT performed well, but less so on the sensitivity part. The third and last dataset tested noun hypernyms both with and without negation. The sensitivity part had to do with the factual accuracy of the statements, rather than the Cloze probability. BERT did well on the hypernyms but quite poorly with respect to negation. The most important limitation of this study was that the language in these tests was purposefully abnormal. So, tasks like these do not characterize the "normal" behavior of the language model.

Hu et al. (2020) found that for syntactic tests, model architecture matters more than amount of training data. Their syntactic test suite was of considerable scale: 34 types of tests, nearly all in a 2×2 design. Further, they did not perform any kind of fine-tuning for the syntactic tests. "Most of these test suites and criteria are designed so that n-gram models cannot perform above chance for $n = 5$ (sometimes greater)." The syntactic tests covered subject-verb agreement, negative polarity item licensing, reflexive pronoun licensing, main verb / reduced relative garden paths, NP/Z garden paths, "gross syntactic expectation", center embedding, filler-gap dependencies, and a novel suite involving clefts. They continued to use the paradigm of assigning sentences with higher surprisal to an ungrammatical class.

Wei et al. (2021) performed a very careful series of experiments on subject-verb agreement using BERT's masking pipeline. They masked out the main verb, obtained the surprisal for the singular and plural forms, and considered the lower surprisal form to be the model's prediction. They evaluated on natural stimuli from Linzen, Dupoux, and Goldberg (2016) and nonce stimuli from Gulordava et al. (2018). By manipulating the training data, specifically removing sentences that contained test verbs, Wei et al. (2021) found that BERT generalized to unseen words pretty well and claimed that this is evidence that BERT learns rules symbolically but can make "noisy observations". This hypothesis is situated between two extremes of an "idealized symbolic learner" and an "item-specific learner". Further, Wei et al. (2021) found two insights into the limitations of BERT's "knowledge". First, BERT can fail to learn the agreement features for infrequent verb forms. Second, imbalance in the verb forms can lead to strong priors that override the grammatical context. So, BERT can pick the wrong form when it is more frequent than the correct form.

11.1.2 Properties of the German G-Maze data

Recall the 2×2 structure of the items in the Sikos et al. (2017) dataset via a reproduction of the example item from Chapter 10:

(1)	The journalist published…	predictive context (`pred`)	
a.	…[the carefully written essay].	pre-nominal modification	(`pre`)
b.	…[the essay that was carefully written].	post-nominal modification	(`pos`)
(2)	The man evaluated…	non-predictive context (`non`)	
a.	…[the carefully written essay].	pre-nominal modification	(`pre`)
b.	…[the essay that was carefully written].	post-nominal modification	(`pos`)

For ease of discussion, my coauthors and I decided to partition these sentences into 6 distinct regions. Region 0 is the sentence-initial determiner ("The"). Region 1 is the subject noun ("journalist" or "man"). Region 2 is the main verb ("published" or "evaluated"). Region 3 is the determiner following the main verb ("the"). Region 4 is the chunk that immediately follows that determiner, either an object noun ("essay") or a modifier phrase ("carefully written"). Region 5 is the chunk that immediately follows Region 4, either a modifier phrase (“that was carefully written”) or an object noun (“essay”). The items technically contained a region following Region 5, but neither my coauthors nor I used it in analyses. My coauthors made the decision to represent the G-Maze response time of a region as the average of the response times for the words in the region. I also discarded Region 0 following the best practice for reading behavior measures.

One of the biggest results from Sikos et al. (2017) is that my coauthors and I observed what seemed to be significantly different response times for all four conditions at Region 4. There are six ways to choose two conditions from four, so in theory, this gave six testable properties of the G-Maze data. With more careful significance testing that did not aggregate over the participants in the data collection experiment, I found that four of the six properties were statistically significant. One of these was that the G-Maze response times at Region 4 for the non-predictive context, post-nominal modification condition were significantly higher than those for the predictive context, post-nominal modification condition. I abbreviate this property by listing the area of the sentence under consideration (Region 4), plus the condition with the higher response times (nonpost), plus the condition with the lower response times (predpost). Thus, this property is named “4 nonpost predpost”. I found three other significant properties at Region 4 and three significant properties at Region 5, given in Section 11.3.

Note that it was somewhat artificial to compare the response times at Region 4 and Region 5 because certain pairings compared the response time for an object noun against the response time for a modifier phrase. So, I developed four additional properties that compare object nouns against object nouns, and I developed four additional properties that compare modifier phrases against modifier phrases. I explain just one of each type of the properties here and refer to Section 11.3 for the complete list. The object noun (n) had a higher response time in the non-predictive context, post-nominal modification condition than in the non-predictive pre-nominal modification condition (n nonpost nonpre). The modifier phrase (m) had a higher response time in the non-predictive context, pre-nominal modification condition than in the non-predictive context, post-nominal modification condition (m nonpre nonpost).

Before Region 4, the modification did not occur yet, so there were only two conditions to track, the predictive context and the non-predictive context. The predictive context subject nouns were *designed* to be more surprising than the non-predictive context subject nouns, leading to the property “1 pred non”. Finally, I compared the main verbs to the subject nouns (area 21). The main verbs had significantly higher response times than the subject nouns in three cases, leading to my last three properties.

11.2 Methods

When a human participant completes a G-Maze task, they must make a decision between two words to continue the sentence grammatically. In the spirit of the literature on syntactic tests for language models, it is natural to envision a task in which a language model computes the surprisal of the true word and the foil word in context, and then it predicts that the true word is the word with the smaller surprisal. I ran this task on the English G-Maze dataset from Chapter 9 and the German G-Maze dataset from Chapter 10.

Then, I converted my list of 19 properties into a suite of 19 tests. The first step was to explore the extent to which the properties are true under different variations of the data. Since different conditions within the same item were clearly not independent, it was most appropriate to average over the participants (who never saw an item in multiple conditions) and then compute a paired t-test across the two conditions, pairing on the items.

I did this first with the log-transformed, centered, and scaled response times. I call this the "RT" model. Then, I computed a linear mixed effects model using the same model structure as the one used in Chapter 10, but without a language model. I called the predicted values from this model the "bs" model, and the residuals from this model the "rT" model. The point of doing this is that I wanted to see whether base factors like length, frequency, and random variation in the participants were ultimately responsible for the properties that I observed. I chose the final list of 19 properties for the test suite based on whether the property held for the rT model with a p-value from the t-test at most 0.05.

I tested all 16 language models from Chapter 10, using linear mixed effects models and best practices from Chapter 9 and the methods for obtaining word-by-word surprisals from Chapter 8. In particular, I centered and scaled the surprisals, predicted the response times by inputting the scaled surprisals into the linear mixed effects models, and then subtracted the bs model values to get language model-informed residuals.

I also tested the checkpoints from the BERT model that I trained from scratch (ScratchBERT). The model started passing tests at the third epoch and the number of tests passed stabilized at the eighth epoch.

11.3 Results

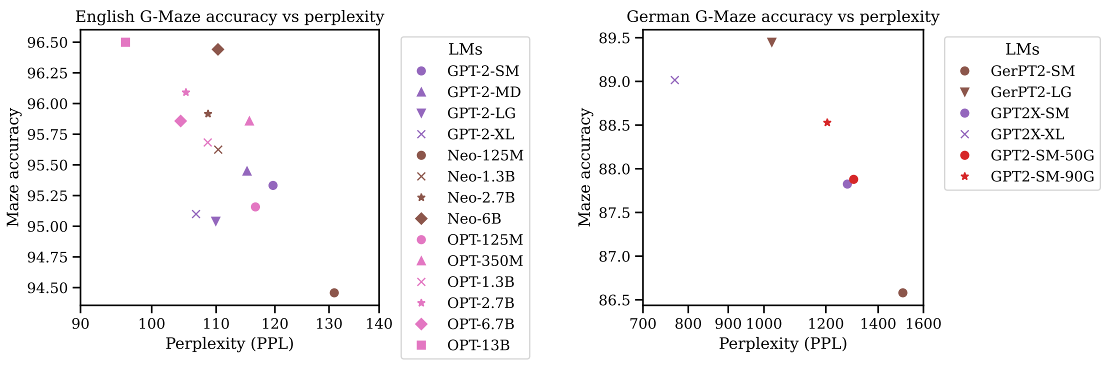


Figure 11.1: Accuracy versus perplexity for Transformer language models on the G-Maze for language models task. The left panel gives the results for the English G-Maze dataset and the right panel gives the results for the German G-Maze dataset. The perplexity axis is on a log scale.

Figure 11.1 presents the results from the G-Maze for language models task that I designed. Note that the range of accuracies is much higher and the range of perplexities is much lower for English. I interpret this to mean that the German G-Maze was more difficult, likely due to the uncommon words that my colleagues selected when designing the items to avoid ceiling effects.

The English accuracies were very correlated with perplexity:
Spearman $\rho = -0.56$, $p = 0.036$. With six data points, it did not seem appropriate to compute a correlation for German. Note though that the pair of GPT-2-SM models that only differed in terms of the amount of training data exhibited the expected trend with respect to G-Maze accuracy, as opposed to the unexpected trend that they had with psychometric predictive power. Further, the English data points clearly do not show the pattern that they had for psychometric predictive power.

I conclude that the G-Maze for language models task was well-formed, but gave results that were too close to perplexity to be independently informative. Further, the task is rather easy for language models. The ceiling of 100% accuracy is not very far off, especially for the English models. This made sense because the foil words in a G-Maze task are designed to be completely unacceptable in context. Therefore, the language model does not need to make that fine-grained of a distinction to identify the foil word.

Property	**rT**	**RT**	**bs**	**KN2**	**KN3**	**KN4**	**KN5**	**LSTM**	**LSTMd**	**LSRC**	**RNN**
1 pred non	*	**	***	***	***	***	***				*
21 pred pred	*	*	X	***	***	***	***	***	**	***	***
21 pred non	***	***	***	***	***	***	***	***	***	***	***
21 non non	***	***	***	***	***	***	***	***	***	***	***
n nonpost nonpre	***	***	X	X	X	X	X		X	X	X
n nonpost predpre	***	***	X	X	X	X	X	*	X		X
n nonpre predpost	*	**		***	***	***	***	*	**	*	**
n predpost predpre	*	.	X	X	X	X	X	X	X	X	X
m nonpre nonpost	***	***	**	*	.			***	***	***	***
m nonpre predpost	***	***		*	.			***	***	***	***
m predpre nonpost	***	***		**	*			***	***	***	
m predpre predpost	***	***	**	**	*			***	***	**	*
4 nonpost predpost	***	***	X	X	.	.	.	***	*	**	*
4 nonpre predpost	***	***	X	**	**	*	*	**	***	***	**
4 nonpre predpre	***	**								.	*
4 predpre predpost	***	***	X	**	**	**	**	**	***	***	*
5 nonpost predpre	**		X	X	X	X	X	X	X	X	X
5 nonpre predpre	***	***						.	.		
5 predpost predpre	*	X	X	X	X	X	X	X	X	X	X

Table 11.1: Left half of the results from the novel test suite built for the German G-Maze corpus. The symbols in the table correspond to significance levels. X : wrong direction, . : $p < 0.1$, * : $p < 0.05$, ** : $p < 0.01$, *** : $p < 0.001$.

Table 11.1 and Table 11.2 present the results from applying the novel test suite over 16 language models that generated surprisals for the German G-Maze corpus. Note that the adjusted response times (the residuals from a linear mixed effects model with only baseline predictors) symbolized by rT passed all 19 tests, but no language model did. Further, there was no test that all language models passed and there was no test that no language models passed.

The tests are grouped by the areas of the sentence that they examine. n-gram models performed well with areas 1 and 21 because the context is so small that it fits within the n. The recurrent models (LSTM, LSTMd, LSRC, RNN) additionally performed well on the modifier tests, and slightly better on Region 4. The non-Transformer models mostly failed the object noun tests and Region 5 tests. It seems that even though the recurrent models have the capacity to store long-term information, they did not do it well enough on this dataset.

BERT	BERTd	gerpt2-SM	gerpt2-LG	gpt2X-SM	gpt2X-XL	GPT2-50G	GPT2-90G
X	X	**	***	**	**	***	**
**	***	.				X	X
*	***	***	***	***	***	*	**
.	***	***	***	***	***	**	***
***	***	***	***	***	***	***	***
***	***	***	***	***	***	***	***
X	*	**	**	.	***	**	**
***				*			
***	***		*		***		
***	***				**		X
***	***	X	X	X	X	X	X
***	***	X	X	X	X	X	X
***	***	***	***	***	***	***	***
***	***	**	***	**	***	**	*
*		***	**	***	***	***	***
***	***	*	**	.	***	*	
***	***	**	***	***	***	***	***
***	***	***	***	***	***	***	***
***	***	***	***	***	***	***	***

Table 11.2: Right half of the results from the novel test suite built for the German G-Maze corpus. The symbols in the table correspond to significance levels. X : wrong direction, . : $p < 0.1$, * : $p < 0.05$, ** : $p < 0.01$, *** : $p < 0.001$.

Contrast this with the patterns for the Transformer models in Table 11.2. The Transformers did very well with Regions 4 and 5, but GPT models struggled with the modifier tests. I assert that this was because the subject and verb were already enough to trigger the modifiers with or without the object noun, representing a pattern that the GPT models learned but humans did not use. Indeed, for the predictive context, the property fully reversed for the GPT models, meaning that the models became more surprised after seeing more of the sentence. The models could have been triggered more strongly by the predictive subject and verb than a human would. Or, they could have assigned a lower surprisal to the pre-nominal modification structure because that structure is more common. Further work is required to tease apart these possibilities, but it is very promising that these syntactic tests have captured the inhuman trends that I hypothesized.

The property "4 predpre predpost" was the one test that GPT2-50G passed but GPT2-90G did not, giving a clue as to why the smaller training set led to a higher psychometric predictive power, as discussed in Chapter 10. This property states that the object noun in a predictive context should be less surprising than the modifiers. But the model with the larger training corpus may have learned a triggering relationship between the subject noun / main verb and the modifiers directly, which would counter the trend.

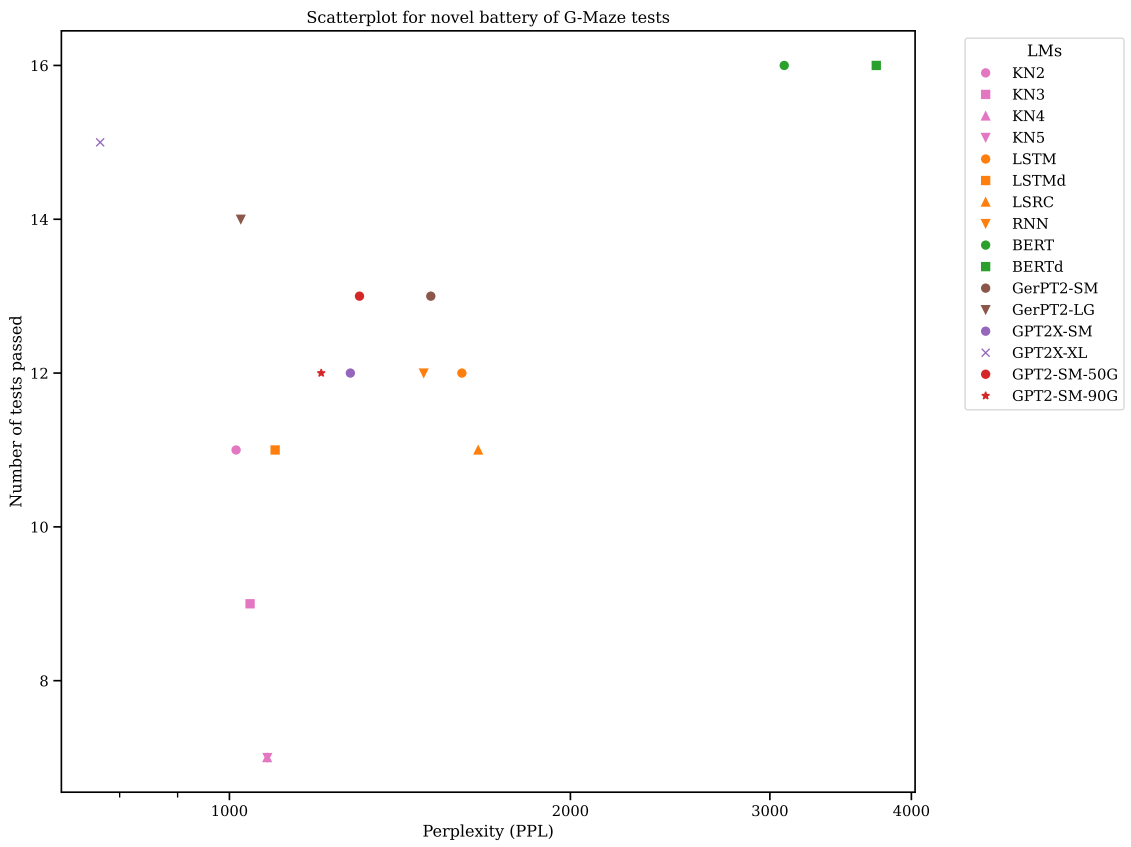


Figure 11.2: Scatter plot for the number of tests passed versus perplexity for the 16 examined language models. The perplexity axis is on a log scale.

Figure 11.2 shows the relationship between perplexity and the number tests passed in my novel suite for the 16 language models that I examined. The correlation was significantly positive: Pearson $\rho = 0.57$, $p = 0.02$. As such, this is rather strong evidence that the number of tests passed in the test suite is not as related to perplexity as is the G-Maze for language models accuracy. So, it just might capture something separate from perplexity and be a worthwhile language model evaluation metric in its own right.

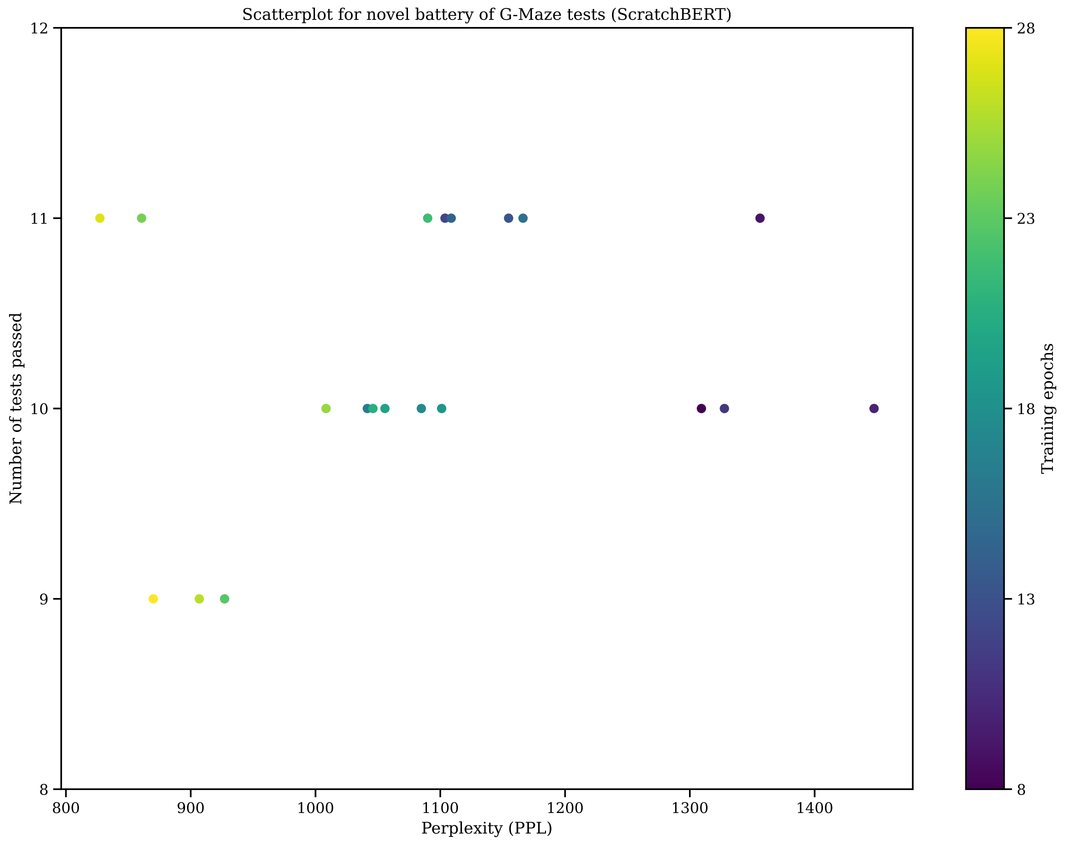


Figure 11.3: Scatter plot for the number of tests passed versus perplexity for the checkpoints of the BERT model that I trained from scratch. The perplexity axis is on a log scale.

As shown in Figure 11.3, the trend was far less strong for the BERT models trained from scratch (Pearson $\rho = 0.22$, $p = 0.35$), as nearly all of the models passed between 9 and 11 tests. Upon inspecting the results, I noticed that the specific tests that the ScratchBERT models passed were most similar to those that the LSTM passed. I find this compelling because the training data and model size were similar for these two architectures.

In the bigger picture, even a not significant result was a good result in this case because it means that the test suite did not mirror perplexity on these 30 models either.

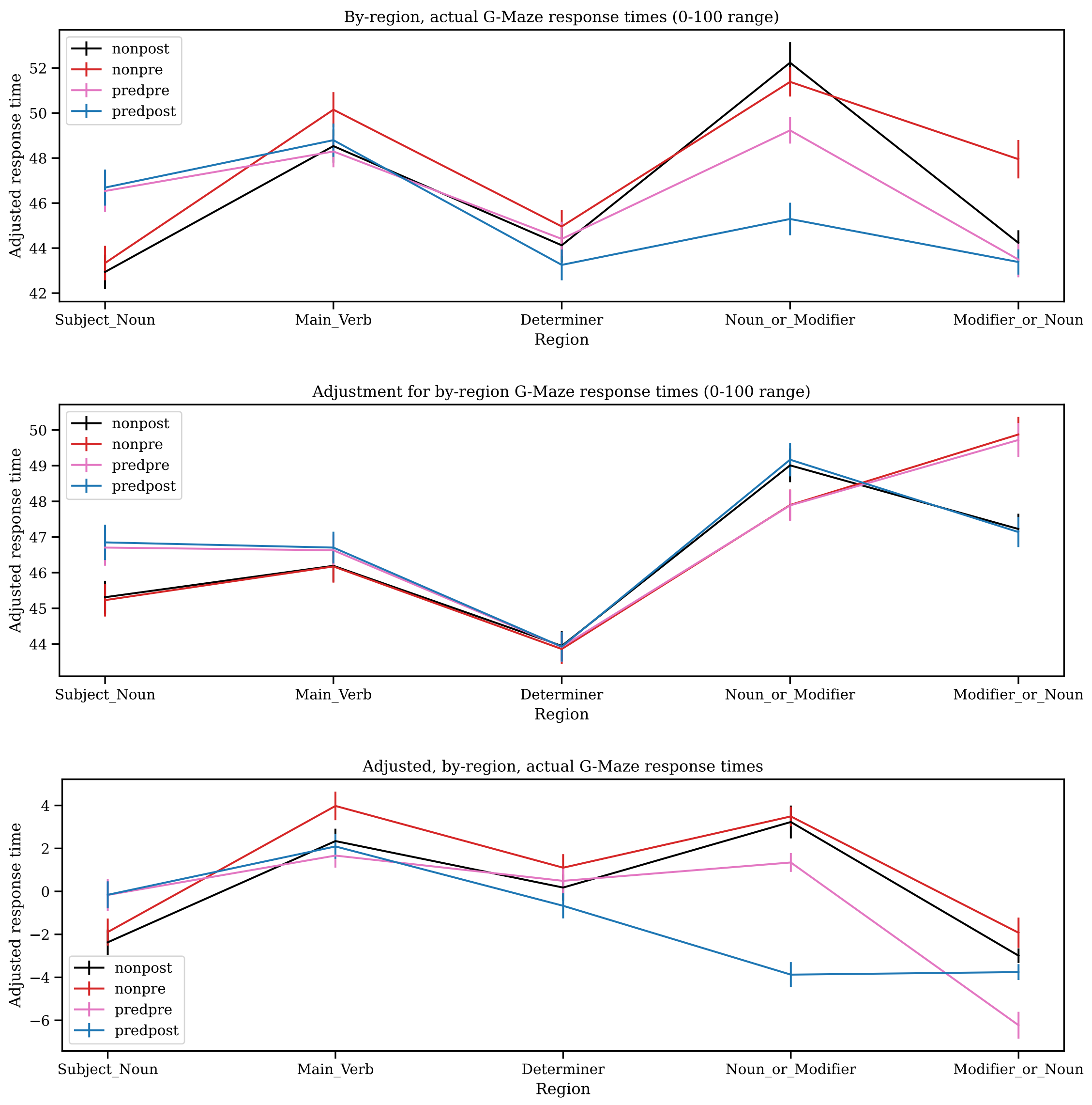


Figure 11.4: Plots of actual average G-Maze response times by region. The top panel is the centered and scaled response times, the middle panel is the predicted values from the baseline linear mixed effects model, and the bottom panel is the residuals from the baseline linear mixed effects model.

In the original publication, my colleagues and I found it illuminating to plot the average G-Maze response times by region for a more intuitive comparison. In particular, Region 4 in the top panel of Figure 11.4 illustrates several of the properties that were featured in Sikos et al. (2017) and became part of my novel test suite.

As shown by the predicted values (middle panel), length and frequency applied to the predictive and non-predictive contexts differently in the first half of the sentence and pre-nominal and post-nominal modification differently in the second half of the sentence. This was to

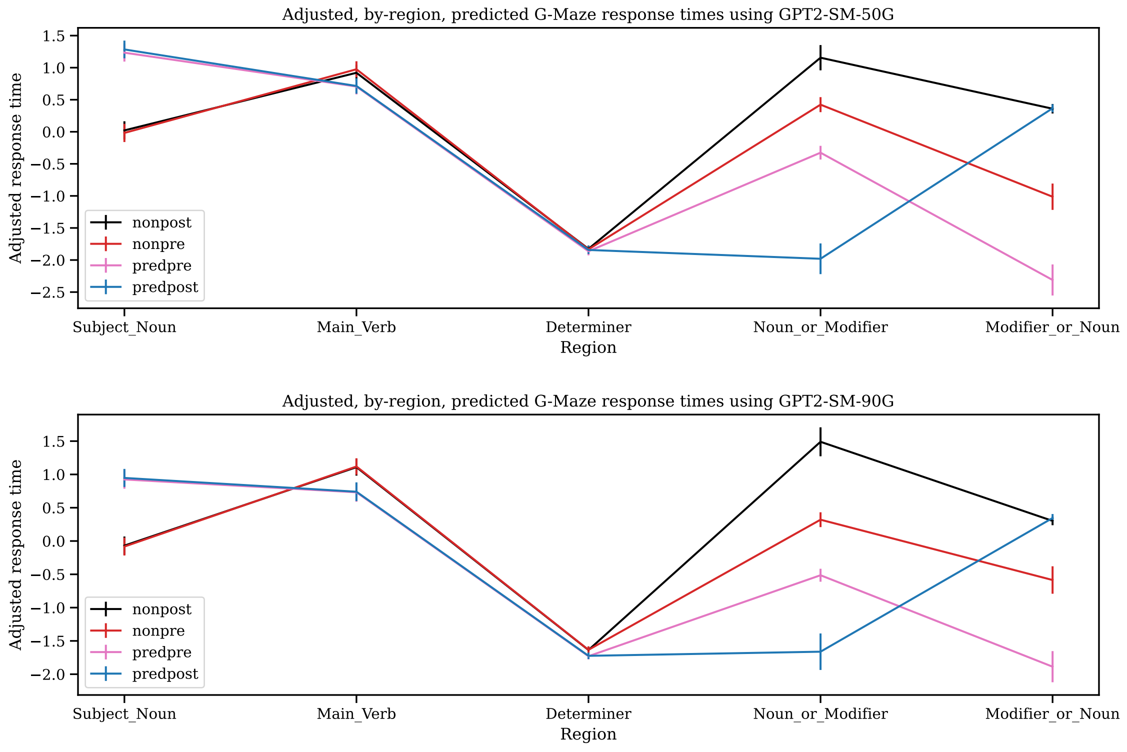
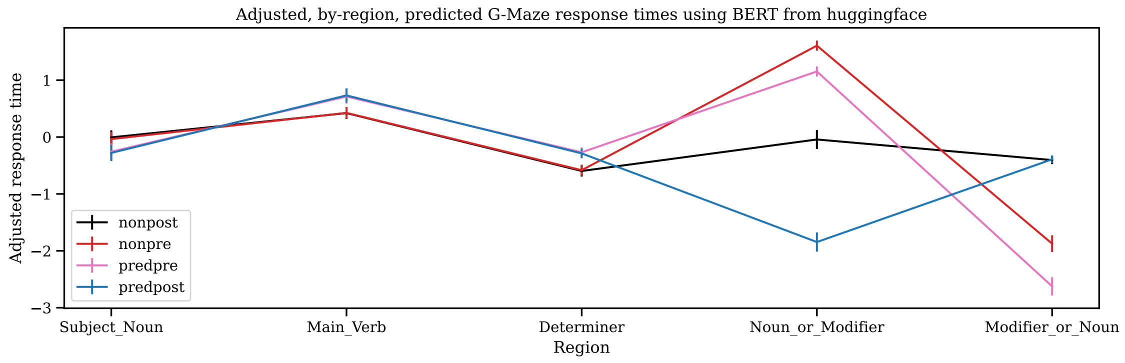


Figure 11.5: Plots of adjusted, by-region, predicted G-Maze response times for the GPT2-SM models.

be expected as length and frequency are entirely local effects. Also note that since nonpost (black) and nonpre (red) were not significantly different at Region 4, this combination did not enter the test suite.

Since the GPT2-SM models produced the most interesting results in other analyses, I produced by-region plots to compare for these two models. As shown in Figure 11.5, the plots are quite similar. The non-predictive context verb had a significantly higher surprisals than the predictive context verb just in the 90G version, although this does not correspond to a test in the suite because the human response times did not have any sensible significant patterns at this region. Still it was not intuitive to assign a lower surprisal to the predictive verb as these verbs were less common (frequent).

Also, the 50G version more correctly captured that the predpost condition (blue) was far easier than the other conditions. In all, the test suite was a more systematic way to evaluate the surprisal profiles from these models since inspection did not easily reveal the most important trends.



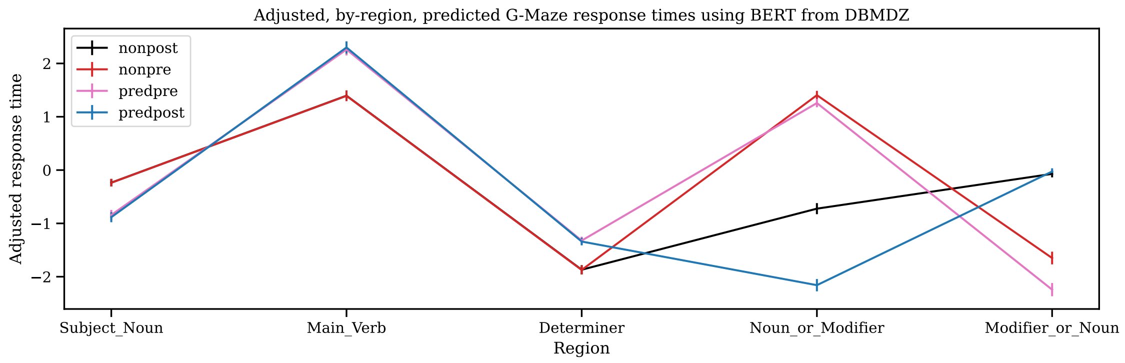


Figure 11.6: Plots of adjusted, by-region, predicted G-Maze response times for the pre-trained BERT models.

The pre-trained BERT models passed the largest number of tests in the suite despite not having the best perplexities. According to their dedicated by-region plots in Figure 11.6, it is not very clear why. In particular, the BERT models made some faulty predictions at Region 4, which was the most important region in Sikos et al. (2017).

While they did correctly predict that the predpost condition was the easiest, they erroneously predicted that the modifiers would always be more surprising than the object noun. This is sensible in that modifiers are optional while the object noun is not, especially following a determiner. It also could have been an artifact of the "no_future" technique, since the model was trained only on complete "sentences", but then it had to assign a surprisal to an adjective that followed a determiner as the last words of the text to be scored.

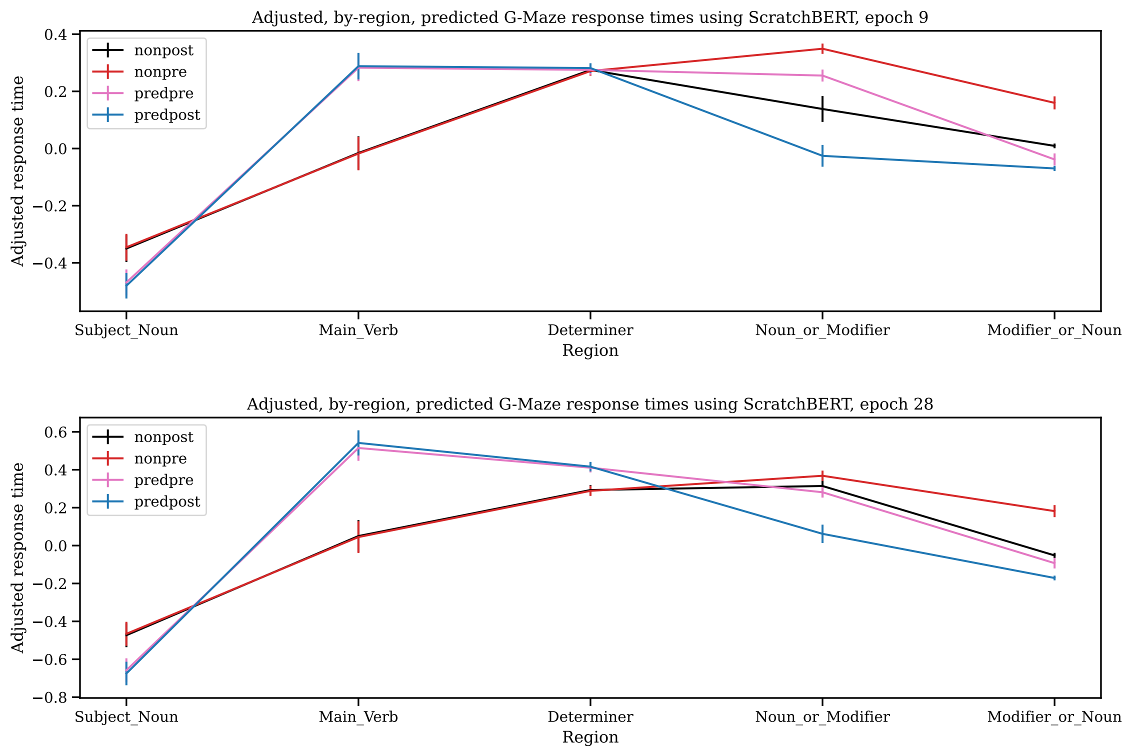


Figure 11.7: Plots of adjusted, by-region, predicted G-Maze response times for the BERT models trained from scratch.

Finally, I compare the underfitted ScratchBERT model with high perplexity and high psychometric predictive power against the possibly overfitted ScratchBERT model with lower perplexity and lower psychometric predictive model in Figure 11.7. Looking particularly at Region 4, it is interesting that the underfitted model looks more like the BERT model while the possibly overfitted model looks more like the human behavior measures and the GPT models.

This evokes an important limitation to my theory. It would be too strong to say that after a language model divergence, the language model fully stops learning good patterns and only learns bad patterns. Even if this were true, it would not be provable. Instead, my theory of the language model divergence argues that the language model begins learning noticeably more bad patterns but could continue to learn more good patterns as well.

11.4 Discussion and Conclusion

This chapter considers the language model divergence from a linguistic standpoint. I formalized and tested a straightforward extension of the G-Maze paradigm for evaluating language models. Then, when this did not prove useful, I developed a suite of 19 tests that quantify the ways in which language model surprisals can be like or not like human G-Maze response times. The number of tests passed is clearly dissociated from perplexity and highlights a promising avenue for experimental methods of language model evaluation.

In the bigger picture, Chapter 10 and Chapter 11 illustrated a rather self-contained data science process. I collected a small dataset focusing on things that I considered important, controlled many more variables than could be controlled in a more "natural" linguistic setup, and derived results that were remarkably clean. Although the dangers of big data and overly powerful models are quite present in the collective minds of researchers at the moment, it is important to consider that small, controlled datasets have their pitfalls, too. In particular, there are increased risks for confirmation bias, over-representing the known, and under-representing the unknown.

Chapter 12

Conclusion and outlook for language modeling research

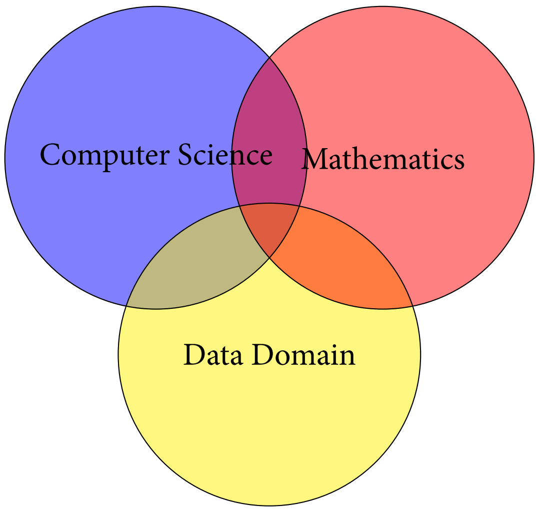


Figure 12.1: The "parent" fields of data science.

To evaluate the humanness of language models, I drew on three academic fields. In the first part of this thesis, I characterized the types of knowledge that language models do and do not capture well. Building these models was computer science research (blue in Figure 12.1). Throughout the thesis, I needed and took advantage of knowledge of the data domain (yellow in Figure 12.1). In particular, human behavior measures are complex, so the field has developed specific methodologies, conventions, and techniques for them. And lastly, the main tools with which I related language model predictions and measures of human behavior belonged to statistics, commonly situated within mathematics (red in Figure 12.1). In short, this thesis is a work of the triple intersection: data science.

12.1 Situating my contributions within CERBA

Accordingly, I will interpret the main contributions of my thesis in terms of my novel formulation of a data science pipeline. Each item in the pipeline is accompanied by a small graphic of the parent fields that I consider to be especially relevant for it.

CS **Collect high-quality data in an organized fashion.**
The Maze study in Chapter 10 produced a carefully manipulated corpus of response times that captured very sophisticated aspects of human reading behavior. I endeavored to make test corpora for language modeling clean while preserving comparability to work by other researchers. Our LM Turing Test study (Chapter 7) obtained introspections into humanness.

M **Extrapolate from exploring representative samples.**
I investigated the robustness of the trends that I found, such as larger Transformers are less correlated with word length and frequency (Chapter 8), the unexpected role of part of speech when analyzing psychometric predictive power and perplexity (Chapter 9), and abilities of language models to reproduce intricate Maze behavior (Chapter 11) so that I may make inferences beyond my specific corpora.

D **Recognize and describe useful patterns in data.**
Powerful linguistic features in Wiegand et al. (2018) outperformed fine-tuned word embeddings. I recognized and explored the interplay between reading time, length, and language model surprisal in Chapter 8 and Chapter 9. I evaluated the longevity of triggering effects in language (Chapter 1). And I used language models with varying degrees of knowledge of the future to characterize the depth of prediction while reading (Chapter 11).

Build appropriate models of data.
CDLM (Singh, Greenberg, and Klakow, 2016) was designed to reproduce the triggering effects recognized in Chapter 1. SRNN (Oualil et al., 2016a) created specialized syntactic and semantic embeddings as these are somewhat separable aspects. LSRC (Oualil et al., 2016b) was designed to maintain a slow-moving meaning representation of the whole document. SWORDSS (Singh et al., 2016) exploited the generalizability of orthography to generate meaning representations for rare words.

Apply models to relevant tasks.
Overall, my work contributed to state-of-the-art language model performance put forward a specific theory about the future of language modeling. I provided data-driven evidence and solutions, such as building test suites for evaluating language models that reward things other than the most probable text. The lexicon from Wiegand et al. (2018) can be used to help keep people safe online. And I suggested desirable qualities of datasets to be collected in the future, turning the pipeline into a cycle.

12.2 On the future of language modeling

When a language model generates text, it generates the most probable text. This text is not necessarily correct, interesting, human, natural, or any other quality that is not encoded in the language model's objective function. I showed some of the mechanism behind this in the first part of this thesis. Namely, specific architecture choices enabled language models to capture long-distance triggering effects because that is, mathematically, what they were missing.

But, the argument can be made that language models do this too well now. In the second part of this thesis, I presented evidence for a language model divergence between the most probable text and the most human-like text that the language models sought to mimic. The result is that Bender and Gebru et al.'s (2021) term "stochastic parrot" is becoming even more apt for contemporary language models. My most concrete prediction for the future of language modeling is that the divergence will widen, resulting in text generation that is more unnatural and easy to identify as artificially created. A promising recourse for this is to adopt evaluation methods of the kind that I developed and used in this work's final chapters. Comparison against measures of human behavior, whether they are introspective ratings, reading times, response times, or curated lists of statistical properties, are a way to reward further advancements without also rewarding banality. For now, the world can take heart that artificial intelligence is not vying to capture some essence of humanity. At best, artificial intelligence can echo it.

Bibliography

Al-Rfou', Rami, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *CoNLL 2013, Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Association for Computational Linguistics, Sofia, Bulgaria.

Alumäe, Tanel and Mikko Kurimo. 2010. Efficient estimation of maximum entropy language models with n-gram features: an SRILM extension. In *INTERSPEECH 2010, Proceedings of the 11th Annual Conference of the International Speech Communication Association*, pages 1820–1823, Makuhari, Chiba, Japan.

Anastasakos, Tasos, Young-Bum Kim, and Anoop Deoras. 2014. Task specific continuous word representations for mono and multi-lingual spoken language understanding. In *ICASSP 2014, Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3246–3250, Florence, Italy.

Aurnhammer, Christoph and Stefan L. Frank. 2019. Evaluating information-theoretic measures of word prediction in naturalistic sentence reading. *Neuropsychologia*, 134:107198.

Baayen, R H, R Piepenbrock, and L Gulikers. 1995. CELEX2 LDC96L14. Web Download. Philadelphia: Linguistic Data Consortium.

Baker, Collin F., Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *ACL 1998, Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montréal, Quebec, Canada.

Bárány, Imre and Van Vu. 2007. Central limit theorems for Gaussian polytopes. *The Annals of Probability*, 35(4):1593–1621.

Baroni, Marco, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky Wide Web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Bellegarda, Jerome R. 1998a. Exploiting both local and global constraints for multi-span statistical language modeling. In *ICASSP 1998, Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 677–680, Seattle, Washington.

Bellegarda, Jerome R. 1998b. A multispan language modeling framework for large vocabulary speech recognition. *IEEE Transactions on Speech and Audio Processing*, 6(5):456–467.

Bender, Emily M, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *FAccT 2021, Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, Virtual Event, Canada.

Bengio, Samy and Georg Heigold. 2014. Word embeddings for speech recognition. In *INTERSPEECH 2014, Proceedings of the 15th Annual Conference of the International Speech Communication Association*, pages 1053–1057, Singapore.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Bengio, Yoshua, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML 2009, Proceedings of the 26th Annual International Conference on Machine Learning*, page 41–48, Association for Computing Machinery, Montréal, Quebec, Canada.

Bengio, Yoshua and Jean-Sébastien Senécal. 2003. Quick training of probabilistic neural nets by importance sampling. In *AISTATS 2003, Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*, pages 17–24, Key West, Florida.

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Bird, Steven, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Black, Sid, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Boston, Marisa F, John T Hale, Shravan Vasishth, and Reinhold Kliegl. 2011. Parallelism and syntactic processes in reading difficulty. *Language and Cognitive Processes*, 26(3):301–349.

Boston, Marisa Ferrara, John Hale, Reinhold Kliegl, Umesh Patil, and Shravan Vasishth. 2008. Parsing costs as predictors of reading difficulty: An evaluation using the Potsdam Sentence Corpus. *Journal of Eye Movement Research*, 2(1):1–12.

Botha, Jan A. and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *ICML 2014, Proceedings of the 31st International Conference on International Conference on Machine Learning*, pages 1899–1907, Association for Computing Machinery, Beijing, China.

Bowman, Samuel R., Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *CoNLL 2016, Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Association for Computational Linguistics, Berlin, Germany.

Brants, Thorsten and Alex Franz. 2006. Web 1T 5-gram Version 1 LDC2006T13. Web Download. Philadelphia: Linguistic Data Consortium.

Brown, Peter F., John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai, and Robert L. Mercer. 1992. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40.

Burnap, Pete and Matthew L Williams. 2015. Cyber hate speech on Twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy and Internet*, 7(2):223–242.

Cameron, A Colin and Frank AG Windmeijer. 1996. R-squared measures for count data regression models with applications to health-care utilization. *Journal of Business and Economic Statistics*, 14(2):209–220.

Charniak, Eugene. 2001. Immediate-head parsing for language models. In *ACL 2001, Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 124–131, Toulouse, France.

Chelba, Ciprian, Tomáš Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Chen, Stanley F. and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *ACL 1996, 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California.

Cheng, Wei-Chen, Stanley Kok, Hoai Vu Pham, Hai Leong Chieu, and Kian Ming A. Chai. 2014. Language modeling with sum-product networks. In *INTERSPEECH 2014, Proceedings of the 15th Annual Conference of the International Speech Communication Association*, pages 2098–2102, Singapore.

Choi, Yoonjung and Janyce Wiebe. 2014. +/-EffectWordNet: Sense-level lexicon acquisition for opinion inference. In *EMNLP 2014, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1181–1191, Association for Computational Linguistics, Doha, Qatar.

Clarkson, Philip and Tony Robinson. 1999. Towards improved language model evaluation measures. In *EUROSPEECH 1999, Proceedings of the 6th European Conference on Speech Communication and Technology*, pages 1927–1930, Budapest, Hungary.

Collobert, Ronan. 2011. Deep learning for efficient discriminative parsing. In *AISTATS 2011, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 224–232, Fort Lauderdale, Florida.

Collobert, Ronan and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML 2008, Proceedings of the 25th International Conference on Machine Learning*, page 160–167, Association for Computing Machinery, Helsinki, Finland.

Cop, Uschi, Nicolas Dirix, Denis Drieghe, and Wouter Duyck. 2017. Presenting GECO: An eyetracking corpus of monolingual and bilingual sentence reading. *Behavior Research Methods*, 49(2):602–615.

Cotterell, Ryan, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *ACL 2016, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1651–1660, Berlin, Germany.

Cover, Thomas and Roger King. 1978. A convergent gambling estimate of the entropy of English. *IEEE Transactions on Information Theory*, 24(4):413–421.

Creutz, Mathias and Krista Lagus. 2005. *Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora using Morfessor 1.0*. Helsinki University of Technology, Helsinki, Finland.

Dai, Zihang, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL 2019, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy.

Davidson, Thomas, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *ICWSM 2017, Proceedings of the International AAAI Conference on Web and Social Media*, volume 11, pages 512–515, Montréal, Quebec, Canada.

Davies, Mark. 2008. Word frequency data from The Corpus of Contemporary American English (COCA). `https://www.wordfrequency.info`.

Demberg, Vera and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

Deng, Lingjia and Janyce Wiebe. 2016. Recognizing opinion sources based on a new categorization of opinion types. In *IJCAI 2016, Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2775–2781, New York, New York.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dias, Gaël, Dinko Lambov, and Veska Noncheva. 2009. High-level features for learning subjective language across domains. In *ICWSM 2009, Proceedings of the International AAAI Conference on Web and Social Media*, pages 199–202, San José, California.

Drieghe, Denis, Keith Rayner, and Alexander Pollatsek. 2005. Eye movements and word skipping during reading revisited. *Journal of Experimental Psychology: Human Perception and Performance*, 31(5):954–969.

Duchi, John, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Dugan, Liam, Daphne Ippolito, Arun Kirubarajan, and Chris Callison-Burch. 2020. RoFT: A tool for evaluating human detection of machine-generated text. In *EMNLP 2020, Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 189–196, Association for Computational Linguistics, Online.

Ehrlich, Susan F and Keith Rayner. 1981. Contextual effects on word perception and eye movements during reading. *Journal of Verbal Learning and Verbal Behavior*, 20(6):641–655.

Eisape, Tiwalayo, Noga Zaslavsky, and Roger Levy. 2020. Cloze distillation: Improving neural language models with human next-word prediction. In *CoNLL 2020, Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 609–619, Association for Computational Linguistics, Online.

Emami, Ahmad and Frederick Jelinek. 2004. Exact training of a neural syntactic language model. In *ICASSP 2004, Proceedings of the 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–245, Montréal, Quebec, Canada.

Engbert, Ralf, André Longtin, and Reinhold Kliegl. 2002. A dynamical model of saccade generation in reading based on spatially distributed lexical processing. *Vision research*, 42(5):621–636.

Ettinger, Allyson. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Filimonov, Denis and Mary Harper. 2009. A joint language model with fine-grain syntactic tags. In *EMNLP 2009, Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1114–1123, Association for Computational Linguistics, Singapore.

Finkelstein, Lev, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Flekova, Lucie and Iryna Gurevych. 2016. Supersense embeddings: A unified model for supersense interpretation, prediction, and utilization. In *ACL 2016, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2029–2041, Berlin, Germany.

Forster, Kenneth I, Christine Guerrera, and Lisa Elliot. 2009. The Maze task: Measuring forced incremental sentence processing time. *Behavior Research Methods*, 41(1):163–171.

Gage, Philip. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Gers, Felix A, Nicol N Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3(Aug):115–143.

Gitari, Njagi Dennis, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230.

Glorot, Xavier and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS 2010, Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256, Sardinia, Italy.

Goodkind, Adam and Klinton Bicknell. 2018. Predictive power of word surprisal for reading times is a linear function of language model quality. In *CMCL 2018, Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics*, pages 10–18, Association for Computational Linguistics, Salt Lake City, Utah.

Goodman, Joshua T. 2001a. A bit of progress in language modeling, extended version. *Computer Speech and Language*, 15(4):403–434.

Goodman, Joshua T. 2001b. Classes for fast maximum entropy training. In *ICASSP 2001, Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 561–564, Salt Lake City, Utah.

Graff, David and Christopher Cieri. 2003. English Gigaword LDC2003T05. Web Download. Philadelphia: Linguistic Data Consortium.

Greenberg, Clayton, Vera Demberg, and Asad Sayeed. 2015a. Verb polysemy and frequency effects in thematic fit modeling. In *CMCL 2015, Proceedings of the 6th Workshop on Cognitive Modeling and Computational Linguistics*, pages 48–57, Association for Computational Linguistics, Denver, Colorado.

Greenberg, Clayton, Vera Demberg, and Asad Sayeed. 2015b. Verb polysemy and frequency effects in thematic fit modeling. In *AMLaP 2015, Architectures and Mechanisms of Language Processing*, page 158, Valletta, Malta.

Greenberg, Clayton, Asad Sayeed, and Vera Demberg. 2015. Improving unsupervised vector-space thematic fit evaluation via role-filler prototype clustering. In *NAACL-HLT 2015, Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 21–31, Denver, Colorado.

Gulordava, Kristina, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *NAACL 2018, Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana.

Gurevych, Iryna. 2005. Using the structure of a conceptual network in computing semantic relatedness. In *IJCNLP 2005, Second International Joint Conference on Natural Language Processing: Full Papers*, Jeju Island, South Korea.

Guthrie, David, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. 2006. A closer look at skip-gram modelling. In *LREC 2006, Proceedings of the Fifth International Conference on Language Resources and Evaluation*, European Language Resources Association (ELRA), Genoa, Italy.

Gutmann, Michael and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS 2010, Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 297–304, Sardinia, Italy.

Hale, John. 2001. A probabilistic Earley parser as a psycholinguistic model. In *NAACL 2001, Second Meeting of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, Pennsylvania.

Hamilton, William L., Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *EMNLP 2016, Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Association for Computational Linguistics, Austin, Texas.

Hatzivassiloglou, Vasileios and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *ACL 1997, 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 174–181, Madrid, Spain.

Heafield, Kenneth. 2011. KenLM: Faster and smaller language model queries. In *WMT at EMNLP 2011, Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Association for Computational Linguistics, Edinburgh, Scotland.

Heafield, Kenneth, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *ACL 2013, Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria.

Hermann, Karl Moritz and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *ACL 2014, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 58–68, Baltimore, Maryland.

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hohenstein, Sven, Hannes Matuschek, and Reinhold Kliegl. 2017. Linked linear mixed models: A joint analysis of fixation locations and fixation durations in natural reading. *Psychonomic Bulletin and Review*, 24(3):637–665.

Hollenstein, Nora, Maria Barrett, and Lisa Beinborn. 2020. Towards best practices for leveraging human language processing signals for natural language processing. In *LiNCr 2020, Proceedings of the Second Workshop on Linguistic and Neurocognitive Resources*, pages 15–27, European Language Resources Association, Marseille, France.

Hollenstein, Nora and Lisa Beinborn. 2021. Relative importance in sentence processing. In *ACL 2021, Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 141–150, Online.

Hollenstein, Nora, Emmanuele Chersoni, Cassandra Jacobs, Yohei Oseki, Laurent Prévot, and Enrico Santus. 2022a. CMCL 2022 shared task on multilingual and crosslingual prediction of human reading behavior. In *CMCL 2022, Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 121–129, Association for Computational Linguistics, Dublin, Ireland.

Hollenstein, Nora, Emmanuele Chersoni, Cassandra L. Jacobs, Yohei Oseki, Laurent Prévot, and Enrico Santus. 2021a. CMCL 2021 shared task on eye-tracking prediction. In *CMCL 2021, Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 72–78, Association for Computational Linguistics, Online.

Hollenstein, Nora, Itziar Gonzalez-Dios, Lisa Beinborn, and Lena Jäger. 2022b. Patterns of text readability in human and predicted eye movements. In *CogALex 2022, Proceedings of the Workshop on Cognitive Aspects of the Lexicon*, pages 1–15, Association for Computational Linguistics, Taipei, Taiwan.

Hollenstein, Nora, Federico Pirovano, Ce Zhang, Lena Jäger, and Lisa Beinborn. 2021b. Multilingual language models predict human reading behavior. In *NAACL 2021, Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 106–123, Online.

Hollenstein, Nora, Jonathan Rotsztejn, Marius Troendle, Andreas Pedroni, Ce Zhang, and Nicolas Langer. 2018. ZuCo, a simultaneous EEG and eye-tracking resource for natural sentence reading. *Scientific Data*, 5(180291):1–13.

Hollenstein, Nora, Marius Troendle, Ce Zhang, and Nicolas Langer. 2020. ZuCo 2.0: A dataset of physiological recordings during natural reading and annotation. In *LREC 2020, Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 138–146, European Language Resources Association, Marseille, France.

Hu, Jennifer, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020. A systematic assessment of syntactic generalization in neural language models. In *ACL 2020, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, Online.

Huffman, David A. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

Ippolito, Daphne, Reno Kriz, João Sedoc, Maria Kustikova, and Chris Callison-Burch. 2019. Comparison of diverse decoding methods from conditional language models. In *ACL*

2019, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3752–3762, Florence, Italy.

Jaccard, Paul. 1912. The distribution of the flora in the Alpine Zone. *New Phytologist*, 11(2):37–50.

Jaffe, Evan, Byung-Doh Oh, and William Schuler. 2021. Coreference-aware surprisal predicts brain response. In *EMNLP 2021, Findings of the Association for Computational Linguistics*, pages 3351–3356, Association for Computational Linguistics, Punta Cana, Dominican Republic.

Jaffe, Evan, Cory Shain, and William Schuler. 2020. Coreference information guides human expectations during natural reading. In *COLING 2020, Proceedings of the 28th International Conference on Computational Linguistics*, pages 4587–4599, International Committee on Computational Linguistics, Barcelona, Spain (Online).

Jaro, Matthew A. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420.

Jindal, Nitin and Bing Liu. 2008. Opinion spam and analysis. In *WSDM 2008, Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230, Palo Alto, California.

Joachims, Thorsten. 1999. Making large-scale support vector machine learning practical. In Christopher J.C. Burges, Bernhard Schölkopf, and Alexander J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*. MIT Press, pages 169–184.

Joulin, Armand, Edouard Grave, Piotr Bojanowski, and Tomáš Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL 2017, Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain.

Jozefowicz, Rafal, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Just, Marcel A, Patricia A Carpenter, and Jacqueline D Woolley. 1982. Paradigms and processes in reading comprehension. *Journal of Experimental Psychology: General*, 111(2):228–238.

Kang, Jun Seok, Song Feng, Leman Akoglu, and Yejin Choi. 2014. ConnotationWordNet: Learning connotation over the Word+Sense network. In *ACL 2014, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1544–1554, Baltimore, Maryland.

Katz, Slava. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401.

Kennedy, Alan and Joël Pynte. 2005. Parafoveal-on-foveal effects in normal reading. *Vision Research*, 45(2):153–168.

Kim, Yoon, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI 2016, Thirtieth AAAI Conference on Artificial Intelligence*, pages 2741–2749, Phoenix, Arizona.

Kirchenbauer, John, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *ICML 2023, Proceedings of the 40th International Conference on Machine Learning*, Honolulu, Hawaii.

Klakow, Dietrich. 1998. Log-linear interpolation of language models. In *ICSLP 1998, Proceedings of the 5th International Conference on Spoken Language Processing*, pages 1695–1698, Sydney, Australia.

Klakow, Dietrich and Jochen Peters. 2002. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1-2):19–28.

Kliegl, Reinhold, Ellen Grabner, Martin Rolfs, and Ralf Engbert. 2004. Length, frequency, and predictability effects of words on eye movements in reading. *European Journal of Cognitive Psychology*, 16(1):262–284.

Kliegl, Reinhold, Antje Nuthmann, and Ralf Engbert. 2006. Tracking the mind during reading: the influence of past, present, and future words on fixation durations. *Journal of Experimental Psychology: General*, 135(1):13–35.

Kneser, Reinhard and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *ICASSP 1995, International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184, IEEE, Detroit, Michigan.

Kuhn, Roland and Renato De Mori. 1990. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583.

Kuncoro, Adhiguna, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *ACL 2018, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia.

Kutas, Marta and Steven A Hillyard. 1980. Reading senseless sentences: Brain potentials reflect semantic incongruity. *Science*, 207(4427):203–205.

Le, Hai Son, Alexandre Allauzen, and François Yvon. 2012. Measuring the influence of long range dependencies with neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 1–10, Association for Computational Linguistics, Montréal, Quebec, Canada.

Levshina, Natalia. 2022. Frequency, informativity and word length: Insights from typologically diverse corpora. *Entropy*, 24(2):280.

Likert, Rensis. 1932. A technique for the measurement of attitudes. *Archives of Psychology*, 22(140).

Linzen, Tal, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *TACL, Transactions of the Association for Computational Linguistics*, 4:521–535.

Linzen, Tal and Brian Leonard. 2018. Distinct patterns of syntactic agreement errors in recurrent networks and humans. In *CogSci 2018, Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 692–697, Madison, Wisconsin.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692.*

Lodhi, Huma, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444.

Logan, Beth, Pedro Moreno, Jean-Manuel Van Thong, Ed Whittaker, Jean manuel Van, and Thong Whittaker. 1996. An experimental study of an audio indexing system for the web. In *ICSLP 1996, Proceedings of the 4th International Conference on Spoken Language Processing*, pages 676–679, Philadelphia, Pennsylvania.

Logan, Beth, Jean-Manuel Van Thong, and Pedro J. Moreno. 2005. Approaches to reduce the effects of OOV queries on indexed spoken audio. *IEEE Transactions on Multimedia*, 7(5):899–906.

Luke, Steven G and Kiel Christianson. 2018. The Provo Corpus: A large eye-tracking corpus with predictability norms. *Behavior Research Methods*, 50(2):826–833.

Luong, Thang, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL 2013, Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Association for Computational Linguistics, Sofia, Bulgaria.

Mahoney, Matt. 2011. Large text compression benchmark. `http://mattmahoney.net/dc/textdata.html`.

Marshall, Sandra P. 2002. The index of cognitive activity: measuring cognitive workload. In *Proceedings of the IEEE 7th Conference on Human Factors and Power Plants*, 7, pages 5–9, Scottsdale, Arizona.

Marvin, Rebecca and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *EMNLP 2018, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Association for Computational Linguistics, Brussels, Belgium.

McCandless, Michael, Erik Hatcher, and Otis Gospodnetic. 2010. *Lucene in Action, Second Edition: Covers Apache Lucene 3.0.* Manning Publications Co., Greenwich, Connecticut.

Meng, Fandong, Zhengdong Lu, Zhaopeng Tu, Hang Li, and Qun Liu. 2016. A deep memory-based architecture for sequence-to-sequence learning. In *ICLR 2016, Workshop Proceedings of the International Conference on Learning Representations*, San Juan, Puerto Rico.

Merity, Stephen, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Meylan, Stephan C and Thomas L Griffiths. 2021. The challenges of large-scale, web-based language datasets: Word length and predictability revisited. *Cognitive Science*, 45(6):e12983.

Mikolov, Tomáš, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR 2013, Workshop Proceedings of the International Conference on Learning Representations*, pages 3111–3119, Scottsdale, Arizona.

Mikolov, Tomáš, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan Černocký. 2011a. Empirical evaluation and combination of advanced language modeling techniques. In *INTERSPEECH 2011, Proceedings of the 12th Annual Conference of the International Speech Communication Association*, pages 605–608, Florence, Italy.

Mikolov, Tomáš, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černocký. 2011b. Strategies for training large scale neural network language models. In *ASRU 2011, Proceedings of the 2011 IEEE Workshop on Automatic Speech Recognition Understanding*, pages 196–201, Waikoloa, Hawaii.

Mikolov, Tomáš, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, Proceedings of the 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048, Makuhari, Chiba, Japan.

Mikolov, Tomáš, Stefan Kombrink, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2011c. Extensions of recurrent neural network language model. In *ICASSP 2011, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5528–5531, Prague, Czech Republic.

Mikolov, Tomáš and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *SLT 2012, Proceedings of the 2012 IEEE Spoken Language Technology Workshop*, pages 234–239, Miami, Florida.

Miller, George A, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244.

Minixhofer, Benjamin. 2020. GerPT2: German large and small versions of GPT2.

Minixhofer, Benjamin, Fabian Paischer, and Navid Rekabsaz. 2022. WECHSEL: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models. In *NAACL 2022, Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3992–4006, Seattle, United States.

Mnih, Andriy and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *ICML 2007, Proceedings of the 24th International Conference on Machine Learning*, page 641–648, Association for Computing Machinery, Corvalis, Oregon.

Mohammad, Saif. 2012. Portable features for classifying emotional text. In *NAACL 2012, Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 587–591, Montréal, Quebec, Canada.

Mohammad, Saif M and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Momtazi, Saeedeh, Friedrich Faubel, and Dietrich Klakow. 2010. Within and across sentence boundary language model. In *INTERSPEECH 2010, Proceedings of the 11th Annual Conference of the International Speech Communication Association*, pages 1800–1803, Makuhari, Japan.

Ney, Hermann, Sven Martin, and Frank Wessel. 1997. Statistical language modeling using leaving-one-out. In *Corpus-based methods in Language and Speech processing*. Springer, pages 174–207.

Nobata, Chikashi, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *WWW 2016, Proceedings of the 25th International Conference on World Wide Web*, pages 145–153, Geneva, Switzerland.

Norvig, Peter. 2009. Natural language corpus data. In Toby Segaran and Jeff Hammerbacher, editors, *Beautiful Data*. O'Reilly Media, pages 219–242.

Oh, Byung-Doh, Christian Clark, and William Schuler. 2021. Surprisal estimators for human reading times need character models. In *ACL 2021, Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3746–3757, Online.

Oh, Byung-Doh, Christian Clark, and William Schuler. 2022. Comparison of structural parsers and neural language models as surprisal estimators. *Frontiers in Artificial Intelligence*, 5:777963.

Oh, Byung-Doh and William Schuler. 2023. Why does surprisal from larger transformer-based language models provide a poorer fit to human reading times? *Transactions of the Association for Computational Linguistics*, 11:336–350.

Osterhout, Lee and Phillip J Holcomb. 1992. Event-related brain potentials elicited by syntactic anomaly. *Journal of Memory and Language*, 31(6):785–806.

Oualil, Youssef, Clayton Greenberg, Mittul Singh, and Dietrich Klakow. 2016a. Sequential recurrent neural networks for language modeling. In *INTERSPEECH 2016, Proceedings of the 17th Annual Conference of the International Speech Communication Association*, pages 3509–3513, San Francisco, California.

Oualil, Youssef, Mittul Singh, Clayton Greenberg, and Dietrich Klakow. 2016b. Long-short range context neural networks for language modeling. In *EMNLP 2016, Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1473–1481, Association for Computational Linguistics, Austin, Texas.

Pascanu, Razvan, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to construct deep recurrent neural networks. In *ICLR 2014, Conference Proceedings of the International Conference on Learning Representations*, Banff, Canada.

Pavlopoulos, John, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deeper attention to abusive user content moderation. In *EMNLP 2017, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135, Association for Computational Linguistics, Copenhagen, Denmark.

Piantadosi, Steven T, Harry Tily, and Edward Gibson. 2011. Word lengths are optimized for efficient communication. *Proceedings of the National Academy of Sciences*, 108(9):3526–3529.

Radford, Alec, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. *OpenAI*.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Rayner, Keith, Jane Ashby, Alexander Pollatsek, and Erik D Reichle. 2004. The effects of frequency and predictability on eye fixations in reading: implications for the EZ reader model. *Journal of Experimental Psychology: Human Perception and Performance*, 30(4):720–732.

Rayner, Keith and Arnold D Well. 1996. Effects of contextual constraint on eye movements in reading: A further examination. *Psychonomic Bulletin and Review*, 3(4):504–509.

Razavi, Amir H, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive language detection using multi-level classification. In *In Proceedings of the Canadian Conference on Artificial Intelligence*, pages 16–27, Ottawa, Canada.

Rill, Sven, Jörg Scheidt, Johannes Drescher, Oliver Schütz, Dirk Reinel, and Florian Wogenstein. 2012. A generic approach to generate opinion lists of phrases for opinion mining applications. In *WISDOM 2012, Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*, pages 1–8, Beijing, China.

Rosenfeld, Ronald. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10:187–228.

Rosenfeld, Ronald. 2000. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning internal representations by error propagation. In David E Rumelhart and James L McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge, pages 318–362.

Ruppenhofer, Josef, Michael Wiegand, and Jasper Brandes. 2014. Comparing methods for deriving intensity scores for adjectives. In *EACL 2014, Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, Volume 2: Short Papers*, pages 117–122, Gothenburg, Sweden.

Saffran, Jenny R, Richard N Aslin, and Elissa L Newport. 1996. Statistical learning by 8-month-old infants. *Science*, 274(5294):1926–1928.

Safi Samghabadi, Niloofar, Suraj Maharjan, Alan Sprague, Raquel Diaz-Sprague, and Thamar Solorio. 2017. Detecting nastiness in social media. In *Proceedings of the First ACL Workshop on Abusive Language Online*, pages 63–72, Association for Computational Linguistics, Vancouver, BC, Canada.

Salicchi, Lavinia, Rong Xiang, and Yu-Yin Hsu. 2022. HkAmsters at CMCL 2022 shared task: Predicting eye-tracking data from a gradient boosting framework with linguistic features. In *CMCL 2022, Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 114–120, Association for Computational Linguistics, Dublin, Ireland.

Saon, George, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, Bergul Roomi, and Phil Hall. 2017. English conversational telephone speech recognition by humans and machines. *arXiv preprint arXiv:1703.02136*.

Sayeed, Asad, Clayton Greenberg, and Vera Demberg. 2016. Thematic fit evaluation: an aspect of selectional preferences. In *RepEval 2016, Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 99–105, Association for Computational Linguistics, Berlin, Germany.

van Schijndel, Marten and Tal Linzen. 2018. A neural model of adaptation in reading. In *EMNLP 2018, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4704–4710, Association for Computational Linguistics, Brussels, Belgium.

van Schijndel, Marten and William Schuler. 2015. Hierarchic syntax improves reading time prediction. In *NAACL 2015, Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1597–1605, Denver, Colorado.

van Schijndel, Marten and William Schuler. 2016. Addressing surprisal deficiencies in reading time models. In *CL4LC 2016, Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity*, pages 32–37, The COLING 2016 Organizing Committee, Osaka, Japan.

van Schijndel, Marten and William Schuler. 2017. Approximations of predictive entropy correlate with reading times. In *CogSci 2017, Proceedings of the 39th Annual Conference of the Cognitive Science Society*, pages 1260–1265, London, United Kingdom.

Schmidt, Anna and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *SocialNLP 2017, Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Association for Computational Linguistics, Valencia, Spain.

Schneider, Walter, Amy Eschman, and Anthony Zuccolotto. 2002. *E-Prime: User's Guide*. Psychology Software Tools, Incorporated, Pittsburgh, Pennsylvania.

Schwenk, Holger and Jean-Luc Gauvain. 2005. Training neural network language models on very large corpora. In *HLT 2005, Proceedings of Human Language Technology Conference* *and Conference on Empirical Methods in Natural Language Processing*, pages 201–208, Association for Computational Linguistics, Vancouver, British Columbia, Canada.

Seker, Sadi Evren, Oguz Altun, Uğur Ayan, and Cihan Mert. 2014. A novel string distance function based on most frequent k characters. *International Journal of Machine Learning and Computation*, 4(2):177–183.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL 2016, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany.

Shannon, Claude E. 1951. Prediction and entropy of printed English. *Bell System Technical Journal*, 30(1):50–64.

Shen, Xiaoyu, Youssef Oualil, Clayton Greenberg, Mittul Singh, and Dietrich Klakow. 2017. Estimation of gap between current language models and human performance. In *INTERSPEECH 2017, Proceedings of the 18th Annual Conference of the International Speech Communication Association*, pages 553–557, Stockholm, Sweden.

Sikos, Les, Clayton Greenberg, Heiner Drenhaus, and Matthew Crocker. 2017. Information density of encodings: the role of syntactic variation in comprehension. In *CogSci 2017, Proceedings of the 39th Annual Conference of the Cognitive Science Society*, pages 3168–3173, London, United Kingdom.

Singh, Mittul, Clayton Greenberg, and Dietrich Klakow. 2016. The custom decay language model for long range dependencies. In *TSD 2016, Proceedings of the 19th International Conference on Text, Speech, and Dialogue*, pages 343–351, Springer International Publishing, Brno, Czech Republic.

Singh, Mittul, Clayton Greenberg, Youssef Oualil, and Dietrich Klakow. 2016. Sub-word similarity based search for embeddings: Inducing rare-word embeddings for word similarity tasks and language modelling. In *COLING 2016, Proceedings of the 26th International Conference on Computational Linguistics*, pages 2061–2070, Osaka, Japan.

Singh, Mittul and Dietrich Klakow. 2013. Comparing RNNs and log-linear interpolation of improved skip-model on four Babel languages: Cantonese, Pashto, Tagalog, Turkish. In *ICASSP 2013, Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8416–8420, Vancouver, British Columbia, Canada.

Smith, Nathaniel and Roger Levy. 2011. Cloze but no cigar: The complex relationship between cloze, corpus, and subjective probabilities in language processing. In *CogSci 2011, Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, pages 1637–1642, Boston, Massachusetts.

Socher, Richard, Eric Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NeurIPS 2011, Advances in Neural Information Processing Systems*, volume 24, Granada, Spain.

Socher, Richard, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP 2012, Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Association for Computational Linguistics, Jeju Island, South Korea.

Soricut, Radu and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *NAACL 2015, Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1627–1637, Denver, Colorado.

Steiger, James H. 1980. Tests for comparing elements of a correlation matrix. *Psychological bulletin*, 87(2):245.

Stolcke, Andreas. 2002. SRILM - an extensible language modeling toolkit. In *ICSLP 2002, Proceedings of the 7th International Conference on Spoken Language Processing*, pages 901–904, Denver, Colorado.

Sundermeyer, Martin, Hermann Ney, and Ralf Schlüter. 2015. From feedforward to recurrent lstm neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):517–529.

Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *INTERSPEECH 2012, Proceedings of the 13th Annual Conference of the International Speech Communication Association*, pages 194–197, Portland, Oregon.

Talukdar, Partha Pratim, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *EMNLP 2008, Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 582–590, Association for Computational Linguistics, Honolulu, Hawaii.

Taylor, Wilson L. 1953. “Cloze procedure”: A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433.

Tian, Edward. 2023a. gptzero update v1. *Substack*.

Tian, Edward. 2023b. new year, new features, new model. *Substack*.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL 2003, Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Tran, Ke, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *EMNLP 2018, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4731–4736, Association for Computational Linguistics, Brussels, Belgium.

Turing, Alan M. 1950. Computing machinery and intelligence. *Mind*, 59(236):433–460.

Tversky, Amos. 1977. Features of similarity. *Psychological review*, 84(4):327–352.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS 2017, Advances in Neural Information Processing Systems*, volume 30, Long Beach, California.

Velikovich, Leonid, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *NAACL 2010, Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 777–785, Los Angeles, California.

Warner, William and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *LSM 2012, Proceedings of the Second Workshop on Language in Social Media*, pages 19–26, Association for Computational Linguistics, Montréal, Quebec, Canada.

Waseem, Zeerak, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First ACL Workshop on Abusive Language Online*, pages 78–84, Association for Computational Linguistics, Vancouver, BC, Canada.

Waseem, Zeerak and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *NAACL SRW 2016, Proceedings of the NAACL Student Research Workshop*, pages 88–93, Association for Computational Linguistics, San Diego, California.

Wei, Jason, Dan Garrette, Tal Linzen, and Ellie Pavlick. 2021. Frequency effects on syntactic rule learning in transformers. In *EMNLP 2021, Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 932–948, Association for Computational Linguistics, Punta Cana, Dominican Republic.

Wiegand, Michael, Christine Bocionek, and Josef Ruppenhofer. 2016. Opinion holder and target extraction on opinion compounds – a linguistic approach. In *NAACL 2016, Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 800–810, San Diego, California.

Wiegand, Michael, Manfred Klenner, and Dietrich Klakow. 2013. Bootstrapping polarity classifiers with rule-based classification. *Language Resources and Evaluation*, 47(4):1049–1088.

Wiegand, Michael, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. 2018. Inducing a lexicon of abusive words – a feature-based approach. In *NAACL 2018, Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1046–1056, New Orleans, Louisiana.

Wiegand, Michael, Marc Schulder, and Josef Ruppenhofer. 2016. Separating actor-view from speaker-view opinion expressions using linguistic features. In *NAACL 2016, Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 778–788, San Diego, California.

Wilcox, Ethan Gotlieb, Jon Gauthier, Jennifer Hu, Peng Qian, and Roger Levy. 2020. On the predictive power of neural language models for human real-time comprehension behavior. In *CogSci 2020, Proceedings of the 42nd Annual Conference of the Cognitive Science Society*, pages 1707–1713, Online.

Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT 2005, Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, Association for Computational Linguistics, Vancouver, British Columbia, Canada.

Winkler, William E. 1990. String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods*, page 354–359, American Statistical Association.

Witzel, Naoko, Jeffrey Witzel, and Kenneth Forster. 2012. Comparisons of online reading paradigms: Eye tracking, moving-window, and maze. *Journal of Psycholinguistic Research*, 41(2):105–128.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP 2020, Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Association for Computational Linguistics, Online.

Wulczyn, Ellery, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *WWW 2017, Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399, Perth, Australia.

Xiang, Guang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 2012. Detecting offensive tweets via topical feature discovery over a large scale Twitter corpus. In *CIKM 2012, Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1980–1984, Maui, Hawaii.

Xiong, Wayne, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. 2016. Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*.

Xu, Peng and Frederick Jelinek. 2007. Random forests and the data sparseness problem in language modeling. *Computer Speech and Language*, 21(1):105–152.

Yang, Yiming, Hanxiao Liu, Jaime Carbonell, and Wanli Ma. 2015. Concept graph learning from educational data. In *WSDM 2015, Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, page 159–168, Association for Computing Machinery, Shanghai, China.

Zesch, Torsten and Iryna Gurevych. 2006. Automatically creating datasets for measures of semantic relatedness. In *Proceedings of the Workshop on Linguistic Distances*, pages 16–24, Association for Computational Linguistics, Sydney, Australia.

Zhang, ShiLiang, Hui Jiang, MingBin Xu, JunFeng Hou, and LiRong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *ACL 2015, Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 495–500, Beijing, China.

Zhang, Susan, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open pre-trained transformer language models. *arXiv preprint: arXiv 2205.01068*.

Zhong, Haoti, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J Miller, and Cornelia Caragea. 2016. Content-driven detection of cyberbullying on the Instagram social network. In *IJCAI 2016, Proceedings of the International Joint Conference on Artificial Intelligence*, pages 3952–3958, New York, New York.

Zipf, George Kingsley. 1936. *The Psychobiology of Language*. Routledge.

Zipf, George Kingsley. 1949. *Human Behavior and the Principle of Least Effort*. Addison-Wesley.

Zouhar, Vilém, Clara Meister, Juan Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. 2023. A formal perspective on byte-pair encoding. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 598–614, Toronto, Canada.