Handling long-term dependencies and rare words in low-resource language modelling

A dissertation submitted towards the degree of

Doctor of Engneering of the

Faculty of Mathematics and Computer Science of

Saarland University

by

Mittul Singh, (M.Sc.)

Saarbrücken / June, 2017

Date of the colloquium	31st August 2017
Dean of the Faculty	Univ.-Prof. Dr. Frank-Olaf Schreyer
Chair of the Committee	Prof. Dr. Vera Demberg
Reporters	
First Reviewer	Prof. Dr. Dietrich Klakow
Second Reviewer	Prof. Dr. Josef van Genabith
Academic Assistant	Dr. Alvaro Torralba

Abstract

For low resource NLP tasks like Keyword Search and domain adaptation with small amounts of in-domain data, having well-trained language models is essential. Two major challenges faced while building these language models for such tasks are 1) how the models handle the long-term dependencies, and 2) how to represent the words which occur with a low frequency (rare words) in the text. To handle long-term dependencies in the text, we compare existing techniques and extend these techniques for domain adaptation for small corpora in Speech Recognition, leading to improvements in word error rates. Further, we formulate a new language model architecture to capture long-term dependencies, helping us understand the extent to which enumeration of dependencies can compare to more popular neural network techniques for capturing such dependencies. Next, to handle rare words in the text, we propose an unsupervised technique of generating rare-word representations, which is more general and requires less mathematical engineering than comparable methods. Finally, embedding these representations in a language model shows significant improvements in rare-word perplexity over other such models.

Kurzzusammenfassung

Für Spracherkennungsaufgaben mit geringen Ressourcen wie Babel Keyword Search und Domainadaptation mit geringen Mengen an Daten aus einem spezifischen Gebiet sind gut trainierte Sprachmodelle essenziell. Zwei wesentliche Herausforderungen bei der Erstellung dieser Sprachmodelle sind der Umgang dieser Modelle a) mit langreichweitigen Abhängigkeiten sowie b) mit Wörtern, die eine niedrige Häufigkeit in Texten aufweisen (seltene Wörter). Um die langreichweitigen Abhängigkeiten in Texten zu untersuchen, werden bestehende Methoden verglichen und diese für Domainadaptationsverfahren für kleine Korpora zur Spracherkennung erweitert. Dieses Vorgehen führt zur Verbesserung der Wortfehlerraten. Weiterhin wird ein neues Sprachmodell entwickelt, um langreichweitige Abhängigkeiten ausfindig zu machen, das im Hinblick auf die Berücksichtigung langreichweitiger Abhängigkeiten hilft zu verstehen, wie sich deren Auflistung im Vergleich zu modernen Verfahren mittels Neuronaler Netze verhält. Was den Umgang mit seltenen Wörtern in Texten angeht, wird ein unüberwachtes Verfahren zur Erzeugung von Vektordarstellungen seltener Wörter eingesetzt. Dieses Verfahren ist allgemeiner und erfordert weniger mathematische Berechnungen als vergleichbare Methoden. Wenn diese Vektordarstellungen in ein Sprachmodell miteinbezogen werden, lassen sich signifikante Verbesserungen gegenüber herkömmlichen Modellen bei der Perplexität von seltenen Wörtern feststellen.

Acknowledgment

First, I would like to thank my advisor, Dietrich Klakow for his help, advice and support. I am greatly indebted to him for all that I have learned from him about doing good research and life. I am also thankful to Michelle Carnell and Saarbrucken Graduate School of Computer Science for this opportunity to come to Saarbrucken. I am also grateful to Avishek Anand for his valuable guidance.

I have been very fortunate to collaborate with several fantastic researchers during my graduate career. I would like to thank my co-authors, Arunav Mishra, Clayton Greenberg, Klaus Berberich and Youssef Oualil. Their immense energy and dedication have fueled my enthusiasm and our various collaborative efforts. I am also fortunate to have the opportunity to mentor some very talented students, Anu, Ilyas, Philipp, and Siwen.

I am greatly indebted to Thomas and Marc for their support and encouragement during the ups and downs of my graduate school time. I would also like to thank my current and former members of Spoken Language Systems Group (LSV), in particular, Anna, Clayton, Volha and Youssef for helping to make LSV a lively and exciting place to work.

I am thankful to the support staff at LSV, Adrin, Ashkan, and Dietmar, for their prompt and sincere efforts. I would also like to thank Angelika, Diana and Jaqueline for their help with administrative matters.

I would like to thank my parents Anil and Renu, and my sister Aparna for always inspiring and supporting me to follow my heart. I am glad to acknowledge my close friendship with Prabhav and Rohit. Their advice and support helped immensely. Finally, I am deeply thankful to my wife, Juhi for all her love and support. I could not have completed this without her help.

To Renu, Anil, Aparna and Juhi.

Contents

Part I

Introduction and Background

Chapter 1

Introduction

Natural languages can be differentiated by the amount of resources available. High-resource languages like English, German and even certain dialects of Chinese, have an ever growing amount of data available on the Internet. Such big data facilitates natural language processing (NLP) based studies for these languages. On the other hand, there exist low-resource languages like Tamil, Zulu etc. which do not have enough data available, such that standard statistical methods may be applied.

In order to process any of these languages, one can either leverage a structured view of the data or use the unstructured view of the data. In a structured view, information is represented as a database capturing abstract and concrete relationships in the language explicitly. The latter would just be the free form with no restrictions on the representation of information. In the context of our work, we concentrate on processing unstructured data, leveraging structure only to enhance the processing of low-resource languages.

As part of this work, we develop NLP systems for Automatic Speech Recognition (ASR) and Keyword Search (KWS) on unstructured language data. A central of part of these tasks is to make sequential word predictions, performed using lan-

guage models.

Various versions of these language models are available, like conventional n-gram based models, recurrent neural networks etc. for such NLP tasks. At a high level, these approaches can be classified as being discrete space or continuous space models. Discrete space models like n-gram based models explicitly enumerate any kind of dependencies in text and use them to make word predictions. On the other hand, the continuous space models perform word predictions by mapping words to an abstract representation space, learning dependencies in this space and then emitting words back to text space to make predictions. As we apply these approaches to different NLP tasks, we face two major challenges (Jozefowicz et al. (2016)):

Handling long-term information in languages

Discrete models explicitly enumerate dependencies between words and then predict using statistics of such enumerations. In contrast, continuous models use vector-based word representations to implicitly capture these dependencies and have mostly enabled predictive systems to outperform the discrete versions. However, most of these continuous models have limits to the extent of long-term information that can be utilized, leaving room for improvement.

Handling rare words

For handling rare words (words which occur with low frequency), discrete models use a näive approach by treating all words as the same. However, this approach can be improved upon, in the case of the continuous-space language models, by effectively leveraging information learned from higher frequency words. This leveraging allows the continuous space models to form good representations even for out-of-vocabulary words — an important set of rare words. However, for low-resource languages, such strategies can be difficult to implement without the help

of structured knowledge.

In this dissertation, we concentrate on processing low-resource language corpora, firstly by focusing on understanding the long-term information in text and by better handling such information using continuous-space models on such languages. Secondly, we address out-of-vocabulary words for low-resource languages and enable better representations for language modelling tasks. In both of these cases, we evaluate our models on a downstream NLP task.

To handle these two challenges, we structure the thesis as follows. In Chapter 2, we discuss the preliminary background on language models, providing details of various language models considered in this thesis, providing an overview of existing state-of-the-art methods. This chapter also provides a discussion on how these models have handled these two challenges and what still remains to be tackled within this scope.

Next in Chapter 3, we compare alternatives for capturing long-range dependencies in natural language. Specifically, we choose recurrent neural networks (RNNs) as a baseline and compares to log-linear interpolation of Skip models (i.e. Skip bigrams and Skip trigrams). The method as such has been published earlier but, we investigate the impact of different smoothing techniques on the Skip models as a measure of their overall performance on four Babel languages: Cantonese, Pashto, Tagalog and Turkish. Furthermore, we apply these Skip models to a Keyword Search task evaluating their performance against conventional n-gram models

In Chapter 4 we further analyze the impact of using long-term information early in Automatic Speech Recognition pipeline. We apply powerful continuous-space models, like long-short-term-memory neural network based language models (LSTMs), in early phases of the decoding of the audio signal, and evaluate it in a language model adaptation-based task. Our experiment had only a small amount of in-domain text and we explore enhanced versions of LSTMs to enable long-term

information capturing on such low-resource in-domain corpora.

Furthermore, to analyze and understand the different long-term dependencies in a text, we develop our own language model in Chapter 5. Using a combination of matrix-weighted formulation to represent long-term information combined with conventional n-gram models for short-range information, we compare our language model with RNNs.

After discussing handling long-term dependencies in a text, in Chapter 6 we analyze handling out-of-vocabulary words in language corpora. We propose an algorithm to generate representations for such words and also induce better representations for words with not enough training data (rare words). These representations are qualitatively analyzed against the state-of-the-art and finally applied in a language modelling framework to observe their performance in predicting rare words.

Additionally, we combine the techniques proposed in this thesis to evaluate handling long-term and rare-word information simultaneously on a Keyword Search task in Chapter 7. We analyze the evaluations using intrinsic and extrinsic metrics, and also examine the limits of using these techniques in such a downstream task.

Finally, we conclude by discussing the following main contributions of this dissertation in Chapter 8:

- A comparison of discrete and continuous models that capture long-term information on low-resourced languages (Chapter 3)
- A novel application of state-of-the-art methods to perform first-pass decoding with long-short-term-memory neural network language models in language model adaptation task (Chapter 4)
- A comparison and evaluation of LSTMs for language model adaptation during a Speech Recognition task (Chapter 4)

- A new neural-based continuous language model to capture different long-term dependencies (Chapter 5)
- A new simple and fast method to handle rare word representations in low-resourced languages (Chapter 6)
- A combined evaluation long-term and rare-word techniques on Keyword Search (Chapter 7)

Parts of this dissertation have been published in the following research papers:

- Mittul Singh and Dietrich Klakow. Comparing RNNS and Log-linear interpolation of improved Skip-model on four Babel languages: Cantonese, Pashto, Tagalog, Turkish. ICASSP 2013
- Mittul Singh, Clayton Greenberg and Dietrich Klakow. The Custom Decay Language Model for long range dependencies. TSD 2016
- Mittul Singh, Clayton Greenberg, Youssef Oualil and Dietrich Klakow. Sub-Word Similarity based Search for Embeddings: Inducing Rare-Word embeddings for Word Similarity Tasks and Language Modelling. COLING 2016
- Mittul Singh, Youssef Oualil and Dietrich Klakow. Approximated and domain-adapted LSTM language models for first-pass decoding in Speech Recognition. INTERSPEECH 2017

Chapter 2

Background

Language modelling has played a key role in building systems for traditional NLP tasks such as Speech Recognition (Mikolov et al. (2010); Arisoy et al. (2012)), Machine Translation (Schwenk et al. (2012); Vaswani et al. (2013)) or Text Summarization (Rush et al. (2015); Filippova et al. (2015)). In most cases, training better language models have improved overall performance of these systems, providing the impetus to training better language models.

Recently, language modelling research has been given a new shot of life by the introduction of continuous-space models like recurrent neural network language models (Cho et al. (2014); Sundermeyer et al. (2012); Oualil et al. (2016)). These models have alleviated the data sparsity issues faced by discrete models and outperformed these discrete models significantly. Hence, leading to better-trained language models.

In this chapter, we take a closer look at both these language model categories and compare the models' design choices, while discussing the models' effectiveness when handling long-term dependencies and handling rare words.

2.1 Handling Long-Term Dependencies

In NLP tasks, significant improvements in performances (Momtazi and Klakow (2011); Sundermeyer et al. (2012)) has been shown by handling long-term dependencies. Both discrete-space models and continuous-space models handle such dependencies differently. We provide an overview of various state-of-the-art methods from both these categories and discuss their approach to handling long-term dependencies in the following sections:

2.1.1 Discrete-Space Models

Discrete space models are the oldest set of language models that are still applied in various language modelling tasks. This is because of their ease-of-use and simplicity that these models continue to be used.

These discrete space models are constructed by enumerating different-length dependencies explicitly. To enumerate dependencies explicitly, discrete models leverage the Markov assumption and express these dependencies as word n-grams. The frequencies of these n-grams are then accumulated and used to make predictions in a general text. As these n-grams are chosen only from a training set, the n-grams are not representative of the underlying data distribution. To overcome such overfitting of the training set, n-gram model distributions are smoothed using various techniques like Kneser-Ney smoothing and Dirichlet smoothing (Ney et al. (1994); MacKay and Peto (1994)).

In particular, Kneser-Ney smoothing has performed well in comparison to other available backing-off smoothing methods (Zhai and Lafferty (2004)). Apart from this smoothing method, Dirichlet smoothing has been found to be quite useful in information retrieval tasks.

Smoothing methods allow for a better generalization of n-gram-based methods but, these methods are still restricted by the size of underlying n-grams. Increas-

ing the n-gram size does allow for a longer range but creates data sparsity issues which make such language models hard to use. So, we also utilize skip-gram features to control the data sparsity and still have a long range. Next, we discuss these skip-gram features and other above-mentioned smoothing methods in detail.

Kneser-Ney Smoothing

On sentence-level word prediction tasks, Kneser-Ney improved performance among different backing-off smoothing methods is due to its ability to model words based on the diversity of their contexts. This dependence on diversity of context also means that these models are good for short ranges only and overfit when the size of the n-grams is increased.

In practice to overcome this bias towards short-range dependencies, such backing-off models are combined with corpus-level-statistics-based language models, which provide information with long-range dependencies avoiding overfitting.

Dirichlet Smoothing

Although Kneser-Ney performs well on sentence-level language modelling tasks like KWS and ASR, we also apply smoothing using Dirichlet priors ($p_T(w|h)$) as described below:

$$p(w|h) = \frac{c(w;h) + \mu p_T(w|h)}{c(h) + \mu} \tag{2.1}$$

where for a given word (w) and its context (h), we train priors ($p_T(w|h)$) on a big training corpus and smooth the corpus counts with these priors using the parameter μ.

For information retrieval related tasks, Dirichlet smoothing priors are able to capture a document collection statistics better than representing distributions generating single sentences. Hence, allowing Dirichlet-based smoothing in sentence-level oriented tasks, like ASR and KWS, enables smoothing short context-based

language models with the help of a corpora-level language model and hence, providing a larger range of dependencies to work with.

Skip-gram based Models

In the above-described smoothing methods, we have assumed the n-gram tuple as a continuous sequence of words. However, this assumption is an unnecessary hurdle to capturing long-range dependencies. Instead, skipping words in an n-gram can capture even longer context than regular n-grams. These skip n-gram features can be used to construct Skip-gram language models. Extending range using these language models also allows controlling parametric growth. Moreover, such language models have been used successfully for intrinsic evaluation in Momtazi et al. (2010a).

Earlier work (Peters and Klakow (2000)) has also experimented with different frameworks to employ these skip-gram features and log-linear interpolation of such features has been shown to capture long-range dependencies better than other frameworks like the linear interpolation. An example formulation using bigram-based skip-gram features on a window of five words $(w_1, w_2, w_3, w_4, w_5)$ is described below:

$$p(w_5|w_1, w_2, w_3, w_4) = \frac{1}{Z(h)} \prod_{i=1}^{4} p^{\lambda_i}(w_5|w_i) \tag{2.2}$$

Here, h is defined to be (w_1, w_2, w_3, w_4), $Z(h)$ stands for the normalization constant dependent on the context (h), λ_is' are the parameters of the log-linear interpolation and $p(w_5|w_i)$ form the distance bigrams with $5-i-1$ representing the number of words skipped in the middle.

Prior work has assessed the usefulness of these models on intrinsic language modelling tasks like perplexity-based experiments but these models have not been applied to extrinsic evaluations on tasks like KWS and ASR. As part of this work, we apply these skip models to extrinsic evaluations and compare them to contemporary neural network-based language models.

2.1.2 Continuous-Space Models

Unlike discrete models, continuous space models represent different dependencies in text implicitly. These continuous models represent words in an abstract word space, where words in a similar context are clustered close to each other. This context and its representation might differ for different continuous space model. Once this model establishes the word space, it utilizes this space to predict the words for any newer context.

Few of such models are applied as part of our study, namely, class-based language models and recurrent neural network language models. These are briefly described in the following sections.

Class-based Language Models

Language models like n-gram models through explicit formulation work well on short-range tasks, but miss out on abstract information contained in the text. To model this abstract information class-based n-gram techniques (Brown et al. (1992); Saul and Pereira (1997)) are utilised.

Most of these class-based methods group words in the text, depending on their context, together in a class. These classes form an abstract representation of words, sharing this information with the class to make predictions on the word level. But, this clustering only captures high-level corpus phenomenon and hence, work well in tandem with conventional n-gram language models.

Recurrent Neural Network Language Models

A major drawback of n-gram- and class-based LMs is data sparsity problem caused by increasing their range and hence, shorter versions of these model end up being used. However, limiting range in such a manner discards a lot of long-range dependencies, which can be beneficial for a predictive system.

Language	Vocabulary Size	Rare Words
Tagalog	22K	11K
Turkish	25K	14K
Vietnamese	6K	1K

Table 2.1: This table reports the statistics for different low-resource language corpora used for language modelling. The second column shows the vocabulary size and the last column shows the rare words (words in vocabulary with frequency ≤ 1 in the training set).

Recently, with the advent of recurrent neural networks (RNNs) for language modelling (Mikolov et al. (2010)), capturing long-range dependencies while avoiding data sparsity issues has become simpler. This effect is due the recurrent connections in RNNs, which allows the context information to cycle longer than an ordinary neural network or an n-gram model.

However, these RNNs suffer from instabilities during training leading to leaking long-term information (Sundermeyer et al. (2012)). To fix this instability issue, long-short-term-memory (LSTM) based units are applied to RNNs and LSTM-based models improve the performance significantly over RNNs. Further improvements (Oualil et al. (2016)) have been made to LSTM which have led to a more efficient use of information in the memory units of the recurrent neural networks.

LSTM and most of its extensions have focussed on re-purposing within sentence information for tasks like ASR. Though, for tasks like Question Answering using across sentence information can be beneficial (Momtazi and Klakow (2011)).

2.2 Handling Rare Words

Apart from research in capturing long-range dependencies, last decade has seen an undeniable growth in NLP research towards resource poor languages. Cieri

et al. (2016) states that such growth has been due to impetus by US and EU funded programs like LORELEI, Babel, METANET, which are only focussed on low-resource language-based systems.

METANET, an EU-funded program, states that *"The majority of European languages are severely under-resourced"* and suggests that a *"coordinated, large-scale effort has to be made in Europe to create the missing technologies and transfer this technology to the languages faced with digital extinction"*. The motivations of METANET focus on improving information access and collaboration across multilingual Europe. Unlike METANET, US National Science Foundation's Documenting Endangered Languages program (2014) is differently motivated and they propose that *"We must do our best to document living endangered languages and their associated cultural and scientific information before they disappear"*. These motivations lead us to not only work on low-resource languages but also handle key language modelling challenges these on low-resource languages.

Specifically, we tackle the problem of providing good representations for words with not enough training data, such words are referred to as rare words. These rare words can form up to 50 % portion of the language's vocabulary, as seen in the Table 2.1. Though, we address this issue in a language modelling framework, handling rare words is important to various other NLP tasks.

Most recent work (Luong et al. (2013); Botha and Blunsom (2014); Soricut and Och (2015)) on handling rare words have worked with morphologically-rich languages. These prior work can be divided into two categories supervised (Luong et al. (2013); Botha and Blunsom (2014)) and unsupervised (Soricut and Och (2015)). In supervised methods existing linguistic knowledge like morpheme-based analysis is applied to generate good representations of the rare words. On the contrary, the unsupervised methods use automatic analysis to obtain good rare-word representations.

One such unsupervised method which enables generation of rare-word represen-

tations is Soricut and Och (2015). In their work, a word-to-word based transformation graph is built, where each such transformation represents a morphological operation to convert one word into another. And as these transformations are represented in a high-dimensional semantic vector space, the rare-word representations generated perform well qualitatively.

Such unsupervised approaches are beneficial in a low-resource scenario, where good representations can be generated without the need of more resources. However, the above graph-based formulation can lead to tuning and implementation overhead in an NLP application. As part of our work, we overcome these overheads by developing a faster and simple-to-implement technique.

2.3 Combining Long-Term and Rare-Word information

Apart from handling long-term and rare-word information separately, efforts have also been made to develop language models which address these issues simultaneously. Character-based neural network language models (Kim et al. (2015); Jozefowicz et al. (2016)) have shown impressive performance gains by capturing long-term dependencies and also handling rare words simultaneously. However, these models have not been applied to downstream tasks like Speech Recognition and Keyword Search.

Previous language model comparison studies have only used long-range n-gram based back-off language models (Hartmann et al. (2014)), where using long-range models have shown promising results. We further this comparison by combining techniques discussed in our work and enhancing existing techniques for application to a Keyword Search task.

Part II

Handling long-term dependencies

Chapter 3

Alternatives to RNNLM

3.1 Introduction

Language modelling research has been uprooted with the arrival of neural network based language models (Bengio et al. (2003); Mikolov et al. (2010); Schwenk et al. (2012)). This change is due to their ability to model low-level language phenomena effectively, which can then be leveraged to build classifiers. Such learning machines have been successfully applied to language modelling tasks (Bengio et al. (2003); Mikolov et al. (2010); Sundermeyer et al. (2012); Kim et al. (2015)) and other tasks.

A side effect of this good performance of neural networks is increased application of such models to train better LMs but at the expense of not much research to understand the underlying data. A few language models have approached solving tasks using Skip-gram based features in a more understanding-driven way (Shazeer et al. (2015); Chelba and Shazeer (2015)), however, unlike neural network research have lacked widespread application. In this chapter, we focus on exploring impor-

tant differences between such Skip-gram-based alternatives and an effective version of neural networks.

Specifically, we choose to compare to recurrent neural network language models (RNNLMs). These models obtain the state-of-the-art performance in language modelling tasks due to their ability to capture long-span information. Recently, these have also given rise to well-performing modifications (Sundermeyer et al. (2012); Kim et al. (2015); Oualil et al. (2016)) but we concentrate on comparing to their easy-to-train version (Mikolov et al. (2011)), described in Section 3.3.1.

While constructing Skip-gram based language models (Section 3.3.2), we embed the Skip-gram features in a log-linear interpolation framework enabling it to capture the long-span information allowing them to compete with recurrent neural networks. We also discuss improved smoothing methods to enhance capturing long-span information in Skip-gram based models. To compare these different language models, we evaluate their perplexity on various low-resourced language corpora (Section 7.5.2). Following these perplexity-based experiments, we perform an extrinsic evaluation of such conventional n-gram approaches on Keyword Search task on a few IARPA Babel program's languages datasets (Section 3.5.1).

3.2 Related Work

Apart from Skip-grams-based language models as a long-span information capturing model, there exist other language models like across-sentence-based models (Momtazi et al. (2010a)). These language models, however, have the advantage of looking into the previous sentence context which can bolster the language model prediction of words in the present sentence. Such an advantage is not available to RNNLMs and hence, we only compare the RNNLMs with regular Skip-gram language models.

There exist other matrix-based variants of using Skip-gram features (Chelba

and Shazeer (2015)) for language modelling. Unlike Momtazi et al. (2010a) formulation using a few parameters in log-linear interpolation, this matrix-based formulation uses a lot more parameters to leverage the Skip-gram features. This increase in parameters also implies a harder to tune method. Thus, we stick to using a more amenable way of optimizing the use of Skip-gram features i.e. the log-linear interpolation of Skip-gram features.

Apart from the model variants being discussed here, we explore evaluating the models on low-resourced languages. To perform this evaluation, we use the low-resourced language modelling corpora available from the IARPA Babel program. Prior work (Gandhe et al. (2014)) has utilized these datasets to evaluate neural network language models. Though in the context long-span language models such a comparison has not been done to the best of our knowledge.

3.3 Language Models

We want to compare two methods that go beyond the trigram to capture long-span information: recurrent neural networks (RNN) and the log-linear interpolation of Skip bigrams and Skip trigrams. In the past years, Mikolov et al. (2010) used RNNs based language model (RNNLM) to capture long range dependencies. This model did not have any limits on the size of the context. Recurrent connections of the neural network allowed it to cycle context information for an arbitrarily long time and provide contexts of arbitrary lengths. This led the RNN based language models to show great improvement in performance over the previous state-of-the-art language models (further discussed in Section 3.3.1).

An alternative is the use of Skip models constructed using log-linear interpolation (Momtazi et al. (2010a)). Unlike RNNs, in this model, all long range dependencies are enumerated explicitly using Skip bigrams and trigrams. Normally these Skip models are smoothed using standard off-the-shelf smoothing techniques like

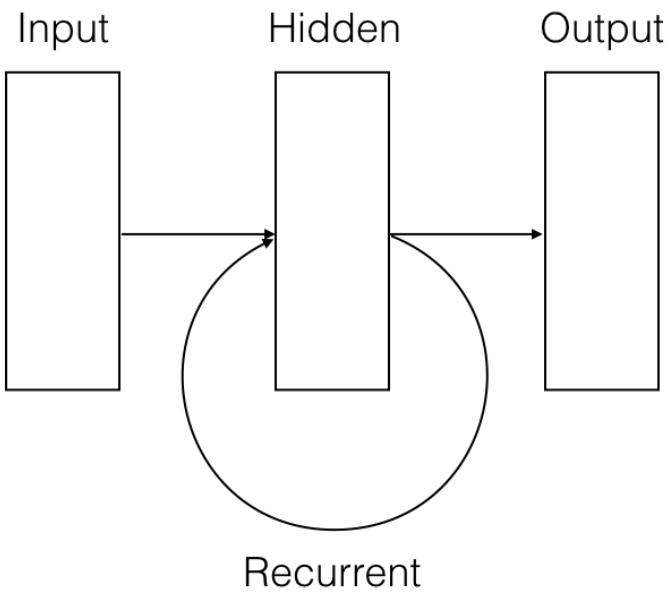


Figure 3.1: The Elman network which forms the basis of RNNLM

the Absolute Discounting variant suggested by Kneser and Ney (Ney et al. (1994)) or the Dirichlet smoothing (MacKay and Peto (1994)). Dirichlet smoothing is very successful over long contexts and frequently used in information retrieval applications (Zhai and Lafferty (2004)).

In our experiments, we found that the constituents of Skip n-grams: bigrams performed well with Kneser-Ney smoothing whereas distance bigrams performed well with Dirichlet smoothing. Hence, we tried a unification of these techniques which improved over the RNN based language model. Section 3.3.2 describes this unification of smoothing techniques in detail. To further improve the performance of the Skip n-gram model, we applied them in a word cluster based language framework. Section 3.3.3 briefly discusses the clustering methods used to develop the above-mentioned application. Section 3.3.4 details the combination of these techniques generated to form word-cluster based modified Skip n-gram model.

3.3.1 RNNLM

RNNLMs and its modification have shown state-of-the-art performances for sequential prediction task like language modelling. RNNLMs use the Elman architecture shown in figure 3.1. A special feature of this architecture is the recurrent connection, through which the context size for these models is essentially infinite, or at least, formally unconstrained. This makes them especially suitable for long range dependencies.

The RNNLMs have also given rise to other more complex architecture like GRU, LSTMs and LSRC (Cho et al. (2014); Sundermeyer et al. (2012); Oualil et al. (2016)). These modifications are able to overcome the vanishing gradient problem of RNNLMs but increase training complexity of these model. In this work, we focus on studying the properties of the fastest of these models, the Elman network.

While applying RNNLMs, training them 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 (2001)). Unfortunately, using classes for normalization complicates the training process, since it creates a particularly volatile meta-parameter. This can be observed in Fig. 5.2, where even for a small variation in classes, RNNLMs show unstable variation in perplexity.

For our experiments, we employ a widely used class-based RNNLM implementation (Mikolov et al. (2011)), which require $H^2 + 2HV + HC$ parameters, where H is the number of hidden units and C is the number of normalization classes. To produce better RNNLMs, we can increase the hidden layer size by one which in turn increases the number of parameters linearly in vocabulary size ($O(V+H+C)$).

3.3.2 Skip n-grams

To combine the Skip n-gram based features various language modelling frameworks can be used, like linear interpolation, log-linear interpolation. We apply the log-linear formulation for its known long-span information capturing abilities (Peters and Klakow (2000)) in comparison to linear interpolation, which is not well suited for such long-span information.

While constructing Skip-gram language models using log-linear interpolation, we combine Skip bigrams, also known as distance bigrams, with conventional unigrams. Constructing Skip-gram language models in such a way leads to a quadratic growth in parameters with respect to the vocabulary size. To add more document level context to these language models, class-based distance bigrams can be interpolated within the log-linear framework. Adding class-based distance bigrams in this mix leads to a linear increase in the number of parameters with the increase in context size.

Smoothing Skip n-grams

Using proper smoothing methods for distance bigrams can lead to a large impact on their performance. We propose to combine the Kneser-Ney smoothed Skip model (p_{KN}) and the Dirichlet smoothed Skip model (p_{Dir}). Table 1 gives a summary of the various smoothing techniques used. A combination of these techniques is carried out using the Jelinek-Mercer interpolation method (Jelinek and Mercer (1980)). The unified smoothing based language model ($UniSt$) thus obtained is described as:

$$p_{UniSt}(w|h) = \frac{\lambda_1 p_{Dir}(w|h) + \lambda_2 p_{KN}(w|h) + p_{BG}(w|h)}{\lambda_1 + \lambda_2 + 1}$$

A unified smoothing performed in such a manner shows performance gains when compared to those obtained by individual smoothing. This is also clear from the results of the experiments as discussed in Section 3.4.2.

Method	$p(w\|h)$
Dirichlet	$\frac{c(w;h)+\mu p_{BG}(w\|h)}{c(h)+\mu}$
Kneser-Ney	$\frac{\max(c(w;h)-\delta,0)}{c(h)}+\frac{\delta c_u(h)}{c(h)}p_{BG}(w\|h)$
Jelinek-Mercer	$(1-\lambda)p_1(w\|h)+\lambda p_2(w\|h)$

Table 3.1: Summary of the smoothing methods used to smooth bigrams and distance bigrams. Here p_{BG} is the background language model to which the smoothing methods back-off to. For a detailed description of these techniques refer to Zhai and Lafferty (2004)

3.3.3 Word Cluster-based Models

To enhance Skip-gram features, we also include distance bigram based class information with the features. To include such class information we apply brown clustering and aggregate Markov model based classes. We describe them as follows:

Brown Clustering

Brown clustering as described in Brown et al. (1992) maximizes the average mutual information over words. Here the mutual information between two words w_i and w_j in the data is defined as follows:

$$MI = P(w_i, w_j) \log \frac{P(w_i, w_j)}{P(w_i)} P(w_j)$$

This is then averaged over all the words in the vocabulary to obtain the average mutual information (AMI).

To derive the class using this formulation, each cluster is initialized with having a single word. Thereafter, pairs of clusters with a minimum decrement in AMI are merged. This goes on until a predefined number of clusters is reached. As a final step to increase the AMI, each term is moved to that cluster for which

the resulting partition has the greatest AMI. This formulation of grouping words together forces the words in similar context to be in the same cluster. Each word's cluster membership defines the word representation of it. Hence, many words share the same representation over the same dataset. Such a representation also forms the major drawback of this approach, assigning explicit cluster membership to each word, leading to an inflexible formulation.

After estimating the word clusters, where $c(w)$ represents the class of a given word w, it can be used to predict the probability of a bigram as follows:

$$p(w_2|w_1) = p(w_2|c(w_2))p(c(w_2)|c(w_1)) \tag{3.1}$$

Aggregate Markov Model

In contrast to Brown clustering, aggregate Markov model (AMM) relaxes the explicit membership assigned by Brown clustering. Thus, one word can belong to more than one cluster. This form of word membership is formulaized by defining the probability of the bigram pair as follows:

$$p(w_2|w_1) = \sum_{c=1}^{C} p(w_2|c)p(c|w_1) \tag{3.2}$$

where $(w_1\ ,\ w_2)$ is the bigram pair, c represents the class variable and C is the number of clusters. In the above model, $p(c|w_1)$ is the probability of w_1 being mapped to class c (conceptualizing context) and $p(w_2|c)$ denotes the probability of w_2 appearing, after a word from class c has appeared. For a word w, aggregate Markov models use $p(c|w)$ to define similarity to a context c. Thereby, allowing a w to belong to different contexts with varying degree. Here, $p(c|w)$ forms a real valued vector representation for each word, with each element of the vector constrained to the interval $[0, 1]$

To solve for different parameters for the model, we use the EM algorithm (Saul and Pereira (1997)) to evaluate the following steps:

E-step

$$p_i(l|w_2, w_1) = \frac{p(w_2|l)p(l|w_1) + \eta}{\sum_{k=1}^{C} p(w_2|k)p(k|w_1) + C\eta} \tag{3.3}$$

M-step

$$p_i(w_2|l) = \frac{\sum_{w_1 in V} N(w_1, w_2)p(l|w_2, w_1) + |V|\eta}{\sum_{w_2'} \sum_{l'} N(w_1, w_2')p(l'|w_2, w_1') + |V|^2\eta} \tag{3.4}$$

where $N(w_1, w_2)$ denotes the absolute frequency of bigram (w_1, w_2) in the corpus and $|V|$ is the vocabulary size of the corpus. Here, we modify the E-step and M-step by smoothing the individual steps using additive smoothing (Lidstone (1920)), which is controlled by the parameter η. Steps (2.3)-(2.4) are iterated until the algorithm converges.

3.3.4 The Combined Model

The Skip bigrams can now be reformulated by applying the techniques already explained in Sections 3.3.2 and 3.3.3. The Skip bigram's constituent models: bigrams ($p(w_1|w_2)$) and distance bigrams $\{p(w_1|w_j) : j = 2, 3, \ldots, n\}$, are individually smoothed using the p_{UniSt} (see section 3.3.2). Simultaneously, word clusters are evaluated over bigrams and distance bigrams (see section 3.3.3). The *UniSt* bigrams and distance bigrams are then combined with their clustering-based counterparts ($p_{soft}(w|h), p_{hard}(w|h)$) through the Jelinek-Mercer interpolation method ($p_c(w|h)$), described as follows:

$$\begin{aligned} p_c(w_i|w_j) = \sigma_1 p_{UniSt}(w_i|w_j) + \sigma_2 p_{soft}(w_i|w_j) \\ + \sigma_3 p_{hard}(w_i|w_j) \end{aligned}$$

where $\sum_{i=1}^{3} \sigma_i = 1$. A final log-linear interpolation yields the following modified Skip (MS) bigram model:

$$\begin{aligned} p_{MS}(w_1|h) = \frac{1}{Z_\lambda(h)} p_c(w_1|w_2)^{\lambda_u} \\ \times \prod_{i=2}^{n} \left(\frac{p_c(w_1|w_i)}{p(w_1)} \right)^{\lambda_i} \end{aligned}$$

where λ_is are the log-linear interpolation parameters.

	Cantonese	Pashto	Tagalog	Turkish	Assamese
Training data					
Words	410536	379596	320841	329380	74775
Sentences	34890	28180	32851	41668	10999
Test data					
Words	23071	19261	14670	15020	72105
Sentences	1768	1313	1298	1784	10310
Vocabulary	9932	9361	13431	23794	7675

Table 3.2: A statistical summary of language datasets used for the experiments in this chapter. (Vocabulary is measured in number of words)

3.4 Perplexity Experiments

In this section, we compare perplexity of Skip n-gram language models against other conventional n-grams language models and RNNLMs on various Babel language corpora.

3.4.1 Data

We used Cantonese, Pashto, Tagalog and Turkish language datasets in our experiments. The datasets include transcriptions of phone conversations collected under the IARPA Babel Program language collection releases babel101-v0.4c, babel104b-v0.4aY, babel106b-v0.2f and babel105-v0.5. Cantonese is a particular dialect of Chinese spoken in large parts of southern China. It is segmented on a character level. The Pashto language (also known as Afghani and Pathani) is mainly spoken in Afghanistan and Pakistan. Tagalog is one of the main languages spoken in the Philippines, and Turkish is the predominantly spoken in Turkey with smaller groups located in Europe and central Asia.

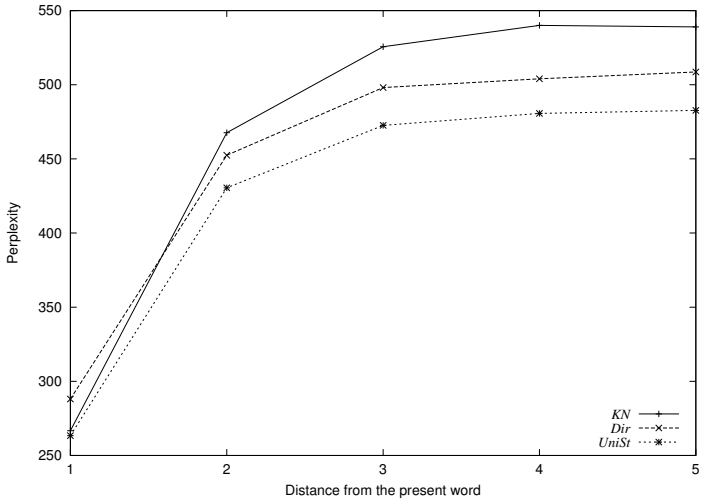


Figure 3.2: Variation of perplexity for different distances used to construct the distance bigrams

To evaluate language models these datasets were divided into training and test data. We used perplexity as a performance measure. First 200 words of the test set were used as the development set to tune the parameters involved in language models. Training and test corpora sizes and respective vocabulary sizes are summarized in Table 3.2.

3.4.2 Smoothing Techniques For Skip Bigrams

We use the unified smoothing technique (*UniSt*) described in Section 3.3.2 to compare the performance of bigrams and distance bigrams over a window of five previous words on the Turkish language dataset. Figure 3.2 shows the variation of performance of distance bigrams on the test set for different distances. As can be seen from the figure, for a smaller context we note that the bigrams and distance bigrams smoothed using the Kneser-Ney smoothing (*KN*) generally performs better than the ones smoothed using the Dirichlet technique (*Dir*). However, for larger contexts *Dir* ($d > 1$)performs better than *KN*. The combined smoothing technique (*UniSt*) takes advantage of both these methods and is thus able to outperform

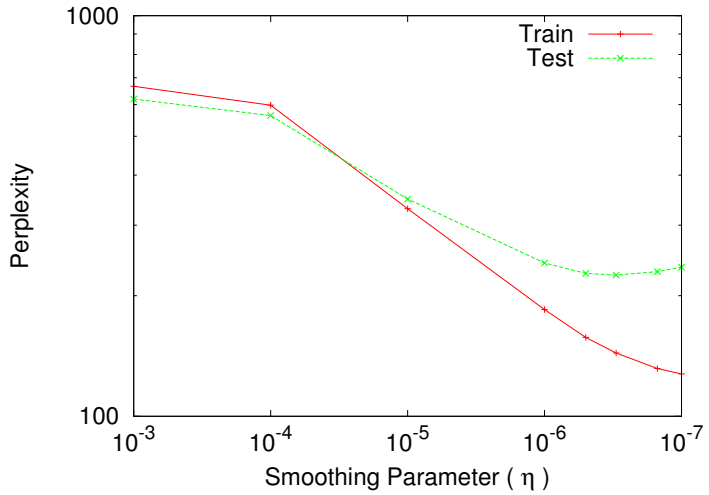


Figure 3.3: Perplexity of aggregate Markov model based LMs on test and training set vs smoothing parameter (η) used in the steps of the EM algorithm plotted on doubly log scale.

its component smoothing techniques. A similar trend was observed for the other languages demonstrating the robustness of the technique.

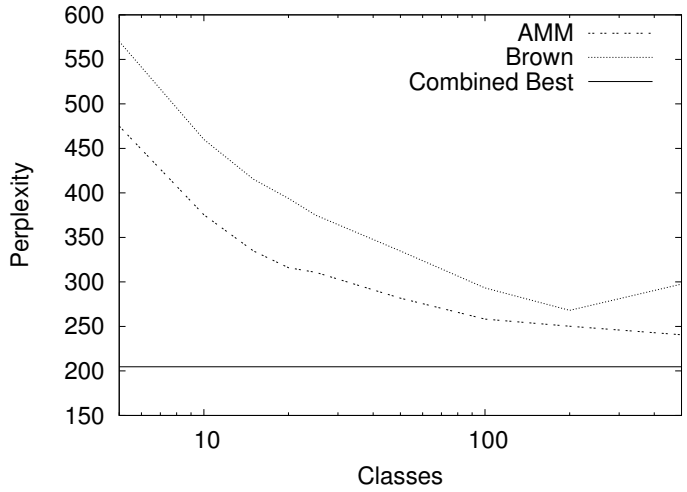


Figure 3.4: Perplexity of AMM and Brown clustering based LM on the test set as a variation of classes on the x axis. Best result is marked by the combination which uses $C = 500$ for both the models.

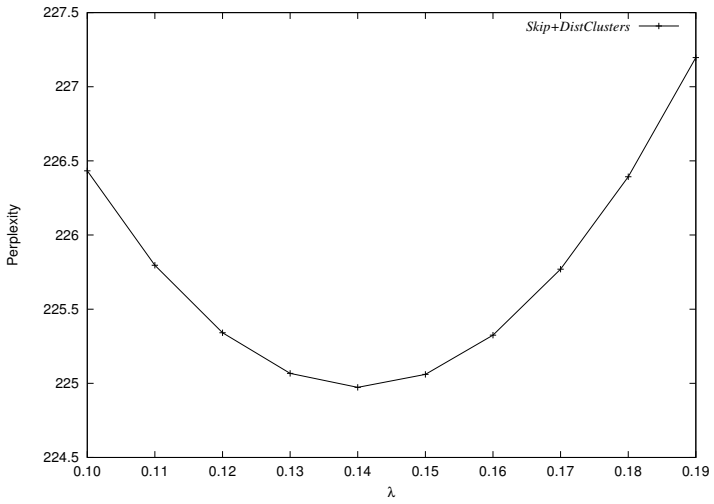


Figure 3.5: Variation of test set perplexity of *Skip+DistClusters* with log-linear interpolation parameter. The least value is observed at $\lambda_4 = 0.138$

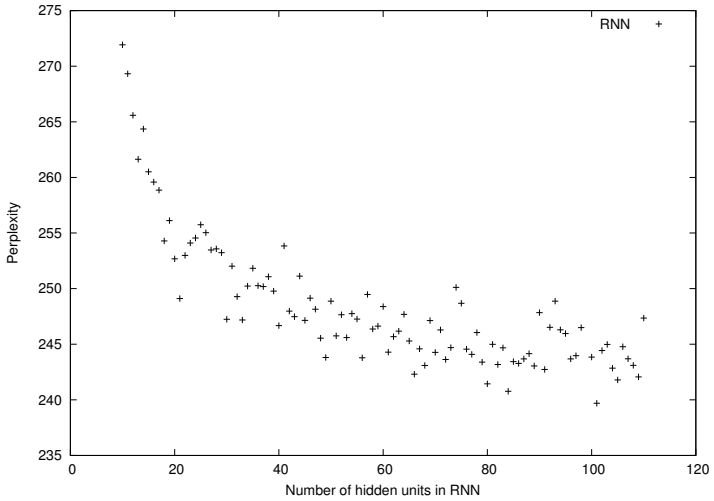


Figure 3.6: Variation of test set perplexity for a particular instance of *RNN* for different number of hidden units

3.4.3 Sensitivity Analysis of Meta-Parameters

We analsye the impact of smoothing parameter on the EM algorithm by varying it in the range $[10^{-7}, 10^{-3}]$. It displays a generally robust variation of perplexity as shown in the Figure 3.3. The best performance of AMMs is noted for $\eta = 3\times]$.

LM	Cantonese	Pashto	Tagalog	Turkish
KN3	85.206	125.853	144.83	266.322
RNN	85.012	122.432	129.150	245.016
$Skip_{KN}$	83.457	124.657	130.031	236.871
$Skip_{UniSt}$	78.563	118.405	123.957	224.277
Skip+DistClusters	77.586	116.382	122.558	223.255
Skip+Clusters	**75.481**	**112.529**	**117.456**	**216.711**

Table 3.3: Perplexity results for different models over Babel language modelling corpora

Similarly, varying the number of classes from 5 to 500 for AMM and Brown clustering based LMs displays a robust performance, shown in Figure 3.4. Though the best performing Brown clustering LM ($C^{best}_{brown} = 200$) and AMM ($C^{best}_{AMM} = 500$) differ in number of classes, their combination performs best when 500 classes are used for each. We attribute the increase in Brown classes for best performance in the above combination to AMM's ability to alleviate the data sparseness better than Brown clustering.

Furthermore, we look at the variation of perplexity with meta parameters of both the Skip n-gram model and RNN based language model. Figure 3.5 and 3.6 show the variation of perplexity as a function of each of these methods' meta parameters. As seen in figure 3.5 and 3.6, we observe that perplexity varies smoothly with λ_4 for the Skip model, whereas the RNN based model shows large variations even for small changes in its number of hidden units. These large variations make the Skip model easier to tune than the RNN based model. Tuning the RNN's meta parameters on the development set can be done using a grid-search based algorithm. However, even this might not be enough to obtain a good performance on the test set.

3.4.4 Results

To evaluate performance of the various methods discussed in this work, we use the language datasets from Cantonese, Pashto, Tagalog and Turkish. A Kneser-Ney smoothed trigram (KN3) is used as the baseline for comparisons with other language models. We report the perplexity results of the *RNN* based language model and the Skip n-gram model smoothed using Kneser-Ney smoothing ($Skip_{KN}$). We compare these results with the unified smoothed Skip model ($Skip_{UniSt}$). For further comparisons, we construct another two versions of the modified Skip model by adding cluster information to its constituent models: one includes cluster information only in the distance bigrams (*Skip+DistClusters*) and the other adds word clusters to both bigrams and distance bigrams (*Skip+Clusters*).

The perplexity numbers of the above language models are summarized in the Table 3.3. As shown in this table, we observe the following trends: both $Skip_{KN}$ and *RNN* show on-par results on the various datasets used. $Skip_{UniSt}$ outperforms the baseline (*KN3*) on different language datasets by about 8-16 %. In comparison to *RNN*, it gives a performance improvement of 3-8 %, whereas with respect to $Skip_{KN}$ it shows an improvement of about 5 %. Adding cluster information to unified-smoothed Skip model's component distance bigrams only improves its performance by 1 %. Additionally, if the word clusters are combined with both the component models of Skip n-gram an improvement of 4 % is observed.

3.5 Keyword Search Experiments

In this section, we compare the Skip-grams with other traditional n-gram models when applied in IARPA Babel program's Keyword Search (KWS) task. We first describe the task briefly and then discuss application of language models within the task.

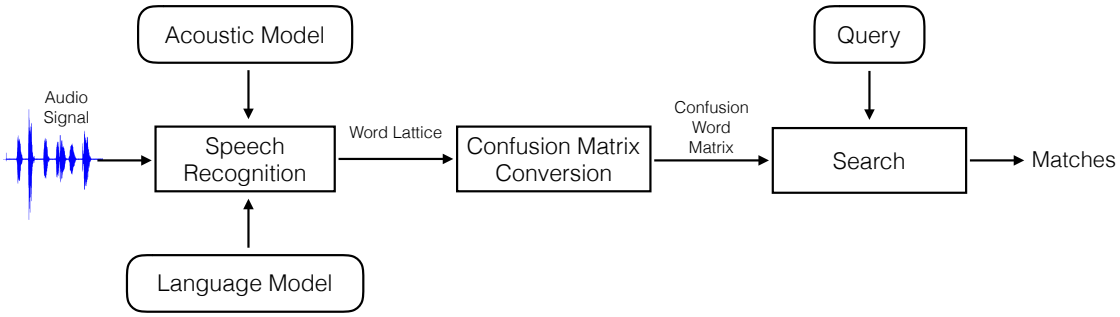


Figure 3.7: An overview of Keyword Search System used for evaluating language models described here.

3.5.1 Keyword Search Task

Keyword Search Task (KWS), also known as the Keyword Search (KWS) task, aims to find exact matches of queries in an audio signal. Previously, NIST 2006 KWS had performed such a task on high-resource languages. With this iteration of KWS, the focus shifts to developing such systems for low-resource languages like Turkish, hence aiming to develop Keyword Search capabilities for various low-resourced languages.

A high-level visualization of a KWS system is displayed in the Figure 3.7. This pipeline displays the audio signal being passed through a large-vocabulary continuous Speech Recognizer (LVCSR), outputting a lattice of words with timestamps. This word lattice is then be converted to a confusion matrix of words, using approach described in Hakkani-Tur and Riccardi (2003). A search is then performed over this word confusions matrix for query terms and the matches are returned.

Applying Language Models in KWS

Language models are used in two different ways as part of this KWS pipeline. First, while performing Speech Recognition a language model is used for decoding

audio signals and producing word lattices. Usually a second pass is performed on this word lattice to rescore the lattice paths with an advanced language model.

Evaluation Metrics

To evaluate this system, Actual Term Weighted Value (ATWV) is used as the primary metric (Gandhe et al. (2014)). ATWV combines missed detections and false alarms as follows:

$$ATWV = \frac{1}{N_{terms}} \sum (1 - P_{Miss}(term) - \beta P_{FA}(term)) \tag{3.5}$$

$$P_{Miss}(term) = 1 - \frac{N_{correct}(term)}{N_{true}(term)} \tag{3.6}$$

$$P_{FA}(term) = 1 - \frac{N_{spurious}(term)}{T_{speech} - N_{true}(term)} \tag{3.7}$$

where,

- $\beta = 999.9$ as described in Mamou et al. (2007)
- $N_{correct}$ is the number of correct detections retrieved by the system
- $N_{spurious}$ is the number of spurious detections retrieved by the system
- N_{true} is the true number of occurrences of the terms in the system
- T_{speech} is the total amount of speech in seconds
- N_{terms} is the total number of terms to detected

In ATWV, every term is weighted equally but missing a rare term is more expensive than missing a frequent term due to P_{Miss}'s dependence on number of *true* terms. Also, in ATWV detecting a term falsely is highly discouraged.

This trade-off of misses and false alarms makes this metric fundamentally different from perplexity used in our experiments. In fact, perplexity is agnostic to

Language Model	Tagalog		Assamese	
	Recall	ATWV	Recall	ATWV
KN3	0.40	0.13	0.32	0.14
$Skip_{UniSt}$	0.40	0.14	0.32	0.13
KN3 + $Skip_{UniSt}$	0.41	0.14	0.32	0.13

Table 3.4: Lattice Recall and ATWV results for a kneser-ney trigram (KN3), Skip n-gram smoothed using unified smoothing ($Skip_{UniSt}$) and also compared to a combination of these two models

these detection events and does not correlate well with ATWV. As we use language models optimized for perplexity and not for ATWV, we also evaluate their impact on lattice recall after a second pass with Skip n-gram language models described previously. Lattice recall measures the number of correct term detections in the all the hypotheses of the lattice. This helps measures how the quality of the lattices change with rescoring with long-span language models.

3.5.2 Keyword Search System Description

As our baseline model, we trained Kneser-Ney smoothed trigram language models (KN3). Skip n-grams are constructed in house and then converted to backing-off ARPA formats to be used as part of the Speech Recognition pipeline.

This Speech Recognition pipeline is built using the 10 hour LimitedLP audio data, available as part of the IARPA Babel program, to train the Janus Speech Recognition toolkit based system (Soltau et al. (2001)). This system applies the above discussed ARPA format language models to the Keyword Search task.

3.5.3 Results

For our experiments with Skip n-grams in KWS, we evaluate ATWV and lattice recall for two Babel language datasets, LimitedLP Assamese (collection release babel102b-v0.5a) and LimitedLP Tagalog, details available in Table 3.2.

We compare the unified smoothing based Skip n-grams (Skip_{UniSt}) against a Kneser-Ney trigram language model. As noted by the performances on these Babel datasets in Table 3.4, the Skip_{UniSt} performs marginally better on Tagalog in terms of ATWV and a reverse trend is seen in case of Assamese, where KN3 performs better than Skip_{UniSt}. Similarly, when combining the two language models (KN3 + Skip_{UniSt}), a performance similar to Skip n-grams in terms of ATWV is observed.

However, no variation is seen in lattice recall for both these languages when performing lattice rescoring with Skip n-gram models. We attribute this lacklustre performance of Skip n-grams to dissimilar training objectives for the language modelling task and the Keyword Search task.

We also note that interpolating neural network based language models along with KN3 has been shown to perform significantly better than KN3 in terms of ATWV (Gandhe et al. (2014)). As our interpolated combinations of Skip n-grams did not perform better than the simple baseline of KN3, we refrained from comparing the combination with other long-span neural network based language models.

3.6 Summary

At the time of this work, Skip n-grams had already inspired longer range language models such as Momtazi et al. (2010a). These particular models proved very beneficial for Question Answering systems (Momtazi and Klakow (2011)). These models, however, used across sentence statistics which are unavailable to RNNLM explicitly and hence, we do not compare to them.

More recent and previous work have utilized Skip n-gram-based features in newer and novel language modelling frameworks (Shazeer et al. (2015)) to improve performance on language modelling tasks. This is contrast to this work where we employ existing smoothing methods to improve the performance of Skip n-grams for longer range dependencies.

In our perplexity based experiments, our proposed unified-smoothed Skip model was able to outperform state-of-the-art language models. Moreover, it outperformed a recurrent neural network based language model. A simple unification of the smoothing techniques gave 3-8 % improvement over a connectionist-based language model across all the four Babel languages. We achieved this by using a lower training complexity model than RNN.

We also applied the word cluster information at a word level to this model which further improved its performance. An addition of word clusters to only the distance bigrams in the Skip model showed a minor improvement, whereas adding them to both the bigrams and distance bigrams showed a greater improvement.

Sensitivity analysis over meta parameters of Skip n-grams and RNN based language model showed that the former is more robust towards small changes in parameters than the latter one. Smooth variation of perplexity in Skip models also makes them easier to tune than a RNN based technique.

Even though, Skip n-grams were able to perform well on perplexity-based language modelling experiments, these models performed comparably with Kneser-Ney trigram models with respect to ATWV in a Keyword Search task. This is in contrast to neural network performance, which through its distributed representation is able to improve upon the KN3 performance in this task. We attribute this comparable performance of Skip n-grams to KN3, to Skip-grams unfocussed training with respect to the perplexity measure. Gandhe et al. (2014) details out one such way of focussed training of language models for Keyword Search task.

In the next chapter, we shift our focus from Skip n-grams to a more contempo-

rary set of long-span language models, the long-short-term-memory neural network models, for improving language model adaptation performance in an Automatic Speech Recognition system.

Chapter 4

Improving applicability of RNNLMs for Speech Recognition

4.1 Introduction

In the previous chapter, while comparing Skip n-gram models with RNNLMs, we observed that Skip n-grams models can be easier to tune and even outperform RNNLMs in terms of perplexity. However, when applying the Skip n-grams in a Keyword Search task, it is unable to outperform a regular Kneser-Ney smoothed trigram model. Moreover, linear combinations of these two models performs comparably with respect to the trigram model, when previously it has been shown that similar combinations of trigram model with neural network-based model outperform the conventional trigram models (Gandhe et al. (2014)).

Hence, we switch to using recurrent neural networks instead of Skip-gram models to harness long-span information in languages for a downstream task. Among the variants of such recurrent neural network-based language models, we exper-

iment with long-short-term-memory (LSTM) LMs (Sundermeyer et al. (2012)) because of their ability to handle the vanishing gradient problem obtaining state-of-the-art performance on various language modelling tasks (Graves et al. (2013); Sutskever et al. (2014); Sundermeyer et al. (2014)).

However, applying such a long-span neural model for in a downstream task like first-pass decoding in a Speech Recognition pipeline can be difficult leading to loss of relevant information at an early stage. Improving applicability of such LMs for first-pass decoding becomes the focus of this chapter.

Here, we explore different approximations of LSTMs enabling the application to first-pass decoding in Speech Recognition (Section 4.4) and also introduce faster ways of achieving such approximations. To evaluate these approximations, we apply these models to Metalogue[1] Speech Recognition task.

Due to little audio training data available in the Metalogue task, we apply our approximation method to help develop two approximate LSTM variants for language model adaptation (Section 4.3). In the context of language model adaptation, such an application of approximated and domain-adapted long-span models to first-pass decoding has not been studied earlier. This study becomes the focus of this chapter, where we compare different adaptation techniques for the LSTM-based long-span model in conjunction with approximation techniques for first-pass decoding in a Speech Recognition task.

4.2 Prior Work

4.2.1 Approximating LSTM language models

Earlier, different techniques for approximating recurrent neural network language models using n-gram language models have been compared. These include variational approximation methods (Deoras et al. (2011a)), probability-based conversion

[1] http://www.metalogue.eu/

(Adel et al. (2014)) and iterative conversion (Arisoy et al. (2014)). Comparison of these techniques has shown that the iterative conversion method has performed best. Though, while using this method smaller bigram-based approximations of RNNLMs have outperformed the trigram-based approximation on Speech Recognition tasks. This behaviour of smaller context bigrams performing better than larger trigrams is unfavourable for including longer range dependencies through this approximation method. Hence, as part of this work, we only apply the variational approximation method and a variant of probability conversion method.

In the probability conversion method, RNNLM probabilities are collected for every word of the training text and these probabilities are assigned to n-grams related to this word. Then the probabilities of n-grams appearing more than once are averaged together, before being normalized and smoothened to produce a backing-off n-gram language model. In contrast, we collect the probability on single representative n-grams instead of the words from the text and thus, skip the averaging step.

4.2.2 First-pass Decoding using RNNLMs

The above-described approximation methods have mostly been developed for first-pass decoding. These methods can be classified as an off-line way of using recurrent neural network language models for decoding, as these methods are first used to approximate RNNLMs to produce an off-line copy of n-gram-based language models, which are later used for decoding purposes.

Another category is the on-line methods for approximating RNNLMs, which directly apply RNNLMs and perform approximation for decoding dynamically (Lecorvé and Motlicek (2012); Huang et al. (2014)). These on-line methods employ a cache-based mechanism, storing RNNLM states in caches and pruning the number of caches as they perform decoding. A more detailed comparison of different cache-based mechanisms has been studied earlier in Huang et al. (2014). In the

context of our work, we only experiment with off-line first-pass decoding methods.

4.2.3 Language Model Adaptation with LSTMs

Recently, recurrent neural network-based architectures have also been applied to an adaptation task (Deena et al. (2016)). Deena et al. (2016) describes feature-based adaptation and model-based adaptation of Elman network-based RNNLMs (Mikolov et al. (2010)), the former includes the auxiliary features from the data for a single training pass and the latter performs a two-pass training. In Deena et al. (2016), these models are applied in a multi-domain adaptation setting where the auxiliary features are either available or can be estimated readily. However, we concentrate on small-sized single-domain adaptation data where only model-based adaptation scheme is applicable for adaptation of LSTMs.

To apply these adapted LSTM, we approximate this model to an n-gram LM for first-pass decoding. Apart from performing adaptation as part LSTM training, we also explore directly applying language model adaptation techniques to approximated regular LSTMs.

4.3 Language Model Adaptation for LSTMs

In this section, we describe the application details of two techniques for language model adaptation techniques used for LSTMs.

4.3.1 Output Adapted Long-Short-Term-Memory Network

Deena et al. (2016) develop a specialized LSTM-based neural network with four layers, one of which is an adaptation layer for domain adaptation. This adaptation layer is placed between the LSTM-based hidden layer (second layer) and the output layer. After training the whole network on background data, the weights between

adaptation layer and output layer are re-trained on the adaptation training corpus. For simplicity, we construct a three-layered LSTM-based neural network, where similarly to the Deena et al. (2016) the first layer forms the projection layer encoding word representations; the next forms the recurrent hidden layer cycling the long-span information; and the final layer outputs probability based on information from the first two layers.

Similarly to Deena et al. (2016), we train this neural network on background corpus, however, re-train only the weights between the hidden layer and the output layer on the adaptation data. This output-adapted LSTM LM can then be used to estimate probabilities for adaptation data. And to use this the output adapted LSTM in first-pass decoding the model is approximated using above techniques discussed in Section 4.4.

4.3.2 Fast Marginal Adaptation for Long-Short-Term-Memory-based n-gram LM

The above method forms a more intrinsic way of adapting an LSTM, we can also directly use the approximated LSTM LM trained on background corpus in the fast marginal adaptation framework i.e. applying the approximated LSTM LM trained on the background corpus as P_{back} as described in Kneser et al. (1997) (a brief introduction is given in Chapter 8.2).

4.4 Approximations to LSTM-based LMs

To perform Speech Recognition decoding with a long-span language model like LSTM LM, we approximate the language model to an n-gram LM. To create such an approximation we use a variant of the probability conversion method. Previously, Adel et al. (2014) collect RNNLM probabilities for every word of the training text and assign these probabilities to n-grams related to this word. Then the probabilities of n-grams appearing more than once are averaged together, be-

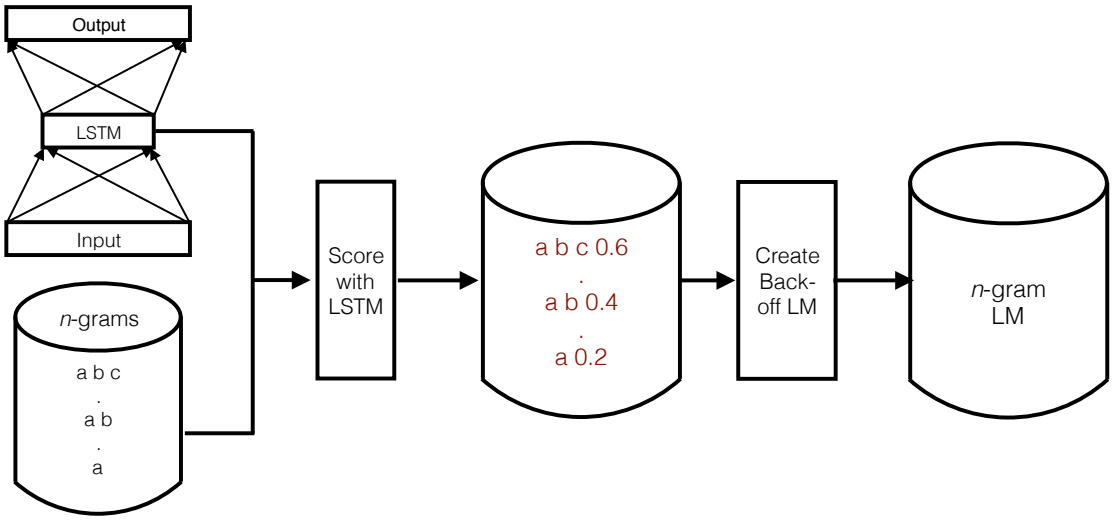


Figure 4.1: The figure shows the steps that are followed to convert n-grams from a text to an n-gram LM to approximate an LSTM LM

fore being normalized and smoothened to produce a backing-off n-gram language model (stored in an ARPA format). When applying this method in our experiments, we directly collect the probabilities on different n-grams instead of the word and hence, skip the averaging step. These steps are outlined in the Figure 4.1. To keep these approximated LSTMs tractable for decoding, we only score n-grams up to the size of five.

Switching from scoring words to n-grams allows us to explore different methods of instantiating n-grams set. First, where n-grams are extracted from the data used for training the language models for the Speech Recognition system.

This method, however, might not cover most n-grams present in the test set. To increase coverage, we apply the second method where we sample text from a Kneser-Ney trigram LM (Kneser and Ney (1995)). As suggested by the infinite monkey theorem (Infinte monkey theorem (2002)), given enough time such a trigram LM will produce all the n-grams present in the test set, hence, providing a better coverage of this set .

Still these methods described above lack the information about the speech data

being input into the Speech Recognizer. To alleviate this deficiency, we use the N-best list produced by a well-performing baseline of the Speech Recognizer. This N-best list is then converted to n-grams, which can then be scored using an LSTM.

4.5 Data

As part of the Metalogue Project, the participants have access to educational debate sessions' speech dataset. As this dataset includes only small amounts of language modelling text, we perform language model adaptation to build an Automatic Speech Recognizer, using the language modelling text from 1996 English Broadcast News Speech corpus (Graff et al. (1997)) as the background text. Further details about these datasets are presented in the next few sections.

4.5.1 Metalogue Speech Data

The speech corpus was collected during the Metalogue project, which aims to develop a dialogue system to monitor, teach and interact with participants, who are debating a multi-issue bargaining topic in order to improve their negotiation skills. The data used in this paper includes debates regarding the implementation of new anti-smoking regulations. The speech data used in this paper contains 6 undergraduates of age between 19 and 25, 4 males and 2 females. All the participants are non-native English speakers. The data comprises 6 different debate sessions, in the English language, with a total duration of 1 hour of speech.

For our language model adaptation experiments we use the transcriptions 5 of these sessions as our adaptation training data (18k tokens) and the 6^{th} (11k tokens) is used for evaluating perplexity and word error rates.

4.5.2 1996 English Broadcast News Speech Corpus

To perform language model adaptation, we use the language modelling text provided by 1996 English Broadcast News Speech Corpus (HUB4) as background information. As the Metalogue corpus uses conversational speech, the HUB4 transcriptions of News speech, also conversational speech, forms a good source of background information.

The HUB4 text is divided into three parts, training (152M tokens) and validation (23M tokens) sets. The vocabulary size of these sets was restricted to around 80K most frequent words in the transcriptions, with all the out-of-vocabulary words (OOV) being replaced by a predefined unknown word symbol. Any new words in the transcriptions of the Metalogue corpus were also added to this vocabulary, making it a closed vocabulary with zero OOVs in the data. Also, any optimization of the background corpus-based LMs was performed using the validation set.

4.6 Perplexity Experiments

To create approximated and adapted LSTMs, we first train LSTMs using in-house GPU enabled tools on the background training corpus and then process this model using the techniques described in Section 4.3 and Section 4.4. In this section, we compare different approximated and adapted models, evaluating these models on test set *perplexity*.

4.6.1 n-gram Sources for Approximated LSTMs

The LSTM model are approximated to n-gram LMs using variational approximation and probability-conversion methods. We design the latter method to create an approximated LSTM LM independently of n-grams' sources, whereas the former inherently produces text to create n-gram LMs. Constructing approximated LSTM-

n-grams' Source	PPL	Size
Variational Approximation		
Sampled from LSTM	327.5	5M
Probability Conversion		
Sampled from KN3	371.6	5M
Sampled from KN3	302.3	250M
HUB4	292.4	150M
N1000	**238.4**	21M

Table 4.1: Approximated *3g*-LSTM models' perplexity on the adaptation test set. These approximated LSTM-based LMs are constructed using different sources of n-grams displayed in the first column, followed by perplexity in the second column and size of data in millions (M) of tokens

based trigram (*3g*-LSTM) language models using these approximation methods, we compare these methods on the usage of different n-grams' sources and the language model *perplexities* are reported in Table 4.1.

In Table 4.1 for a five million token-size corpora, using sampled text from LSTM obtains a better perplexity than sampled text from a Kneser-Ney smoothed trigram model (KN3). Though obtaining the latter text with KN3 is much faster than the former. For instance, due to the constrained size of GPU memory, the text sampler took nearly a week to sample five million tokens from an LSTM. Whereas, using KN3 to sample a similar number of tokens took half an hour as a single-threaded job. So instead of slowly improving perplexity by sampling more text from LSTM, we sampled up to 250 million tokens (time to sample $\sim$ 1 day) from KN3 to improve the approximated LSTM's perplexity, which is reflected by a 7% improvement in perplexity over 327.5 obtained using five million tokens sampled from an LSTM.

As an alternative source of n-grams, we also used the background corpora (HUB4 with 150 M tokens) and the 1000-best lists (N1000 with 21 M tokens) cre-

LM	HUB4	SAMPLED	N1000
n-gram LMs			
KN3	198.1	-	247.3
FMA3	202.6	-	**169.5**
Original LSTMs			
$\text{LSTM}_{\text{adapt}}$	433.3	-	-
LSTM_{bg}	181.1	-	-
OA-LSTM	**166.5**	-	-
Approximated and Adapted LSTM LMs			
3g-OA-LSTM	258.4	265.9	224.0
3g-LSTM_{bg} + *3g*-$\text{LSTM}_{\text{adapt}}$	184.2	202.6	**180.1**
FMA-*3g*-LSTM	263.4	261.6	185.7
Interpolation of FMA3 and Approximated LSTMs with KN3			
FMA3 + KN3	184.4	-	166.1
3g-OA-LSTM + KN3	174.6	178.0	**141.4**
3g-LSTM_{bg} + *3g*-$\text{LSTM}_{\text{adapt}}$ + KN3	164.4	180.3	148.5
FMA-*3g*-LSTM + KN3	192.1	191.7	153.0

Table 4.2: Perplexity results on different adapted LSTM-based trigram models, constructed using various n-gram sources.

ated using KN3 for first-pass of decoding (Section 4.7.1). Using these alternate sources, approximated *3g*-LSTM models further improved the perplexity over the sampled-text versions. As these alternate sources are more easily available, using these sources instead of sampled text to construct approximated LSTMs allows for a faster application of language models to first-pass decoding.

4.6.2 Approximate LSTMs for Language Model Adaptation

In this section, we compare the following language model adaptation techniques for LSTMs in terms of perplexity. As baselines for this comparison, we chose the Kneser-Ney smoothed trigram model (KN3) and its fast marginal adaptation (FMA3).

LSTM$_{\text{bg}}$ and **LSTM**$_{\text{adapt}}$ represent LSTMs trained on background corpus HUB4 and adaptation Metalogue training corpora respectively. To perform first-pass decoding with these LMs, we construct approximate trigram versions of these LMs (as described in Section 4.3), labeled with a prefix ***3g-***. To apply a simple baseline, we combine these approximated LMs linearly to perform language model adaptation, forming *3g*-LSTM$_{\text{bg}}$ + *3g*-LSTM$_{\text{adapt}}$.

OA-LSTM are the output-adapted LSTMs trained on HUB4's training set and re-trained on adaptation training set as described in Section 4.3.1. To perform first-pass decoding with this LM, we construct an approximate trigram version of this LM, represented by *3g*-OA-LSTM.

FMA-3g-LSTM are constructed using *3g*-LSTM$_{\text{bg}}$, embedded in the fast marginal adaptation framework (FMA).

Among the baselines and original LSTM LMs, OA-LSTM has the best perplexity value. However, applying an LSTM directly to first-pass decoding is prohibitively expensive and hence, we approximate the LSTM LMs using three different sources of n-grams, namely, the background corpus (HUB4), the text sampled using KN3 (SAMPLED) and the 1000-best lists (N1000).

Comparing the approximated adapted LSTMs, *3g*-LSTM$_{\text{bg}}$ + *3g*-LSTM$_{\text{adapt}}$ obtains the lowest perplexity among the approximated LSTM LMs. However, FMA3 an adapted n-gram LM scoring N1000 n-grams obtains a lower perplexity. Only after linear interpolation with KN3 the approximated LSTM LMs are able to improve upon the perplexities with *3g*-OA-LSTM + KN3 achieving the lowest

perplexity.

Moreover, for the different source of n-grams, 1000-best list based corpora shows the best results across because of the extra decoding-pass information already contained in these lists.

4.7 Speech Recognition Experiments

We compare the different approximated and adapted LSTM models on Metalogue Speech Recognition Task. In this section, we describe the relevant system details and Speech Recognition experiments using these LSTM models.

4.7.1 Metalogue Speech Recognition System

The Metalogue corpus mainly contains non-native English speech with many spontaneous speech phenomena such as repetitions, hesitations, etc and this corpus does not offer much audio training data for a Speech Recognizer. Therefore, we train a multi-style Speech Recognition system on a collection of corpora, including the 1996 English Broadcast News Speech Corpus, Voxforge[2], LibriSpeech (Panayotov et al. (2015)) and WSJ0 (Pallett et al. (1994); Kubala (1995)), which amounts to a total training duration of $\approx$ 1200 hours of training data for the acoustic model. The acoustic model was trained using the standard GMM/HMM model combined with Linear Discriminant Analysis (LDA), Maximum Likelihood Linear Transform (MLLT) estimation and Speaker Adaptive Training (SAT). All these models were trained and applied using Kaldi toolkit (Povey et al. (2011)) by conveniently modifying some of its training recipes. Training other types of acoustic models such as subspace Gaussian mixture models and deep neural networks did not lead to any noticeable improvement.

[2]http://www.voxforge.org

LM	HUB4	N1000
Baselines		
3g-LSTM$_{\text{adapt}}$	50.0	45.9
3g-LSTM$_{\text{bg}}$	55.7	41.0
FMA3	36.5	**36.1**
Approximated and Adapted LSTM LMs		
3g-OA-LSTM	42.9	39.0
3g-LSTM$_{\text{bg}}$ + *3g*-LSTM$_{\text{adapt}}$	39.2	37.8
FMA-*3g*-LSTM	38.8	**37.3**
Interpolation of Approximated LSTM models with KN3		
FMA3 + KN3	35.6	35.0
3g-OA-LSTM + KN3	36.0	36.3
3g-LSTM$_{\text{bg}}$ + *3g*-LSTM$_{\text{adapt}}$ + KN3	36.2	**34.7**
FMA-*3g*-LSTM + KN3	35.2	34.8

Table 4.3: WERs on Metalogue Speech Recognition task for adapted and approximated *3g*-LSTMs on n-grams from HUB4 and 1000-best lists (N1000)

4.7.2 Speech Recognition Experiments with *3g*-LSTMs

The aforementioned adapted and approximated LSTM-based trigram models are applied in the above described Metalogue Speech Recognition system (Section 4.7.1). Table 4.3 reports *word error rate* (WER) results of these experiments using HUB4 and N1000 as n-gram sources.

For these experiments, we use *3g*-LSTMs trained on the background (bg) and adaptation (adapt) training sets as the simple baselines. As a more competitive baseline, we also apply the fast-marginal-adaptation-based trigram language model to the Metalogue Speech Recognition task.

Among the LSTM-based models, FMA-based LSTM (FMA-*3g*-LSTM) obtains

the lowest WERs across different sources of n-grams, in contrast to *3g*-LSTM$_{\text{bg}}$ + *3g*-LSTM$_{\text{adapt}}$ that showed a better perplexity value. Until these approximated LSTM LMs are interpolated with a KN3, these LMs are all outperformed by the FMA3 model. This is quite similar to results obtained during our perplexity experiments.

As KN3 is interpolated with approximated LSTM models and FMA3, we observe a higher rate of improvement for LSTM models than the FMA3 model. Moreover, the KN3-interpolated version of LSTMs obtains a lower word error rate than the KN3-interpolated FMA3 model. We attribute this observation to KN3's modelling short-context information that is more complementary to LSTM's capabilities to leverage longer-context information than in comparison to a short-context FMA3 model and hence, the interpolation with KN3 benefits the LSTM models more.

In Table 4.3, we also observe the impact of using different n-gram sources. 1000-best list based n-grams, which are rich in first-pass decoding information, obtain lower word error rates in comparison to n-grams from HUB4.

Also, among each of the subcategory of baseline, approximated & adapted and KN3-interpolated language models, FMA-based language models generally perform best for different sources of n-grams. Previously, it has been shown that fast marginal adaptation generally outperforms simple linear interpolation of language models trained on background and adaptation training sets (Kneser et al. (1997)). In most cases, we observe a similar trend. When comparing *3g*-OA-LSTM to FMA-*3g*-LSTM word error rates, the former performs worse whereas in terms of perplexity the non-approximated version performs much better than the FMA-*3g*-LSTM version. This degradation in performance, we suspect is due to losses in the approximation process.

We note that, while using sampled text from KN3 as a source we observed similar language model trends, however, the sampled text as the source was out-

LM	N1000
5g-OA-LSTM + KN3	35.9
5g-LSTM$_{\text{bg}}$ + *5g*-LSTM$_{\text{adapt}}$ + KN3	35.0
FMA-*5g*-LSTM + KN3	**34.4**

Table 4.4: *5g*-LSTMs interpolated with kneser-ney trigram (KN3) on Metalogue Speech Recognition task with 1000-best list as the n-grams source performed by n-grams from HUB4 and N1000 and hence, are not reported.

4.7.3 Speech Recognition Experiments with *5g*-LSTMs

In all the above experiments, we chose n-grams up to a size three to be scored by LSTMs. As LSTMs are long-span models, using a short-range n-grams can be a hindrance in the models performing well. To alleviate this issue, we score n-grams up to five built on a 1000-best list with adapted LSTMs (*5g*-LSTM) to be used in decoding. We only consider 1000-best list n-grams as these n-grams obtain the best results in our previous experiments and report the results in the Table 4.4. As shown in this table, the larger context mostly helps improve the Speech Recognition performance, with the FMA-based *5g*-LSTM interpolated with KN3 (*5g*-LSTM+KN3) performing the best on the Speech Recognition task.

4.8 Conclusion

In this chapter, we applied approximate LSTMs to first-pass decoding as part of the Metalogue Speech Recognition pipeline. To create these approximate LSTMs, we scored the n-grams from different texts using an LSTM to create different variants of approximated LSTMs.

Among these methods, we found that using N-best list based n-grams for LSTM scoring led to the best performance on the Metalogue Speech Recognition task. The other n-gram instantiations for LSTM scoring didn't perform as well but

increased the overall correct hypotheses recall of the *1000*-best list in comparison to regular models. In our future work, we plan to explore N-best rescoring methods along with approximated methods to improve the correct hypotheses scores.

Due to only a small amount of in-domain training data available for Metalogue Speech Recognition task, we also applied model-based adaptation directly to LSTMs and n-gram style language model adaptation to approximate LSTM. This allowed us to use approximate-adapted LSTMs for first-pass decoding. In all our experiments, most approximate-adapted LSTMs outperformed the non-adapted versions. Moreover, when we increase the context size of approximate-LSTM n-gram LM, we observed further reduced word error rates, which also formed the best result obtained in our experiments.

In summary, we were able to faster approximate LSTMs to a better performance than variational-approximated LSTM and also as a first we were able to use their adapted versions for first-pass decoding in Speech Recognition. Though we applied these fast approximation methods to LSTM, these techniques are generic and can also be applied to other large-context neural network model. Hence, making long-span neural models more applicable at the early stages of decoding in Speech Recognition.

Chapter 5

Capturing Long Range Dependencies

5.1 Introduction

In our foregoing experiments, we saw that the Skip n-grams outperformed RNNLMs with respect to perplexity on low resourced languages. However, when applied to downstream tasks like Keyword Search and Automatic Speech Recognition, they performed comparably with Kneser-Ney trigram LM (KN3) only after a combination with this trigram model. Also in our ASR experiments, the approximated LSTM-based LMs combined with KN3 outperform the Skip-gram and KN3 combination. Hence, showing that recurrent neural network based LMs are able to better capture long-span information than Skip n-gram based LMs.

This is attributed to such a neural network's recurrent connections, which can allow old information to cycle in the network. Hence, leading to remember the information state of the data longer than other simpler neural network and conventional n-gram models, which does not have any such information state. Another important difference between recurrent and non-recurrent n-gram models is

the former's ability to model word representations and help in alleviating the data sparseness (Mnih and Hinton (2007); Mikolov et al. (2011)). To understand the limits of the behaviour of these recurrent neural network models, we build simpler but comparable models and evaluate their performances. Section 5.2.3 outlines the development of such a model. As we map long-span information in data, we observe that such information exists beyond the normal sentence boundary. To quantify information in dependencies of long distances, we use a variant of pointwise mutual information. Specifically, for a given pair of words (w_1, w_2) separated over a distance d, we examine the ratio of the actual co-occurrence rate to the statistically predicted co-occurrence rate: $c_d(w_1, w_2) = \frac{P_d(w_1, w_2)}{P(w_1)P(w_2)}$. 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. In Fig. 5.1, we show an example variation of this correlation for pronouns with the distance d on the English Gigawords corpus (Graff and Cieri (2003)).

In this corpus, 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, Fig.5.1 demonstrates that plenty of word triggering information is spread out over long distance dependencies that is typically beyond the reach of N-gram LMs.

Several models, such as the cache-based LM (Kuhn and De Mori (1990)), Skip models (**??**Guthrie et al. (2006); Momtazi et al. (2010b)), and recurrent neural network language models (RNNLMs) (Mikolov et al. (2011)) have been proposed to capture triggering in large contexts, but they usually only handle auto-triggering and/or have too many parameters to scale with vocabulary size. In this paper, we develop a novel modelling scheme, the Custom Decay Language Model (CDLM), which is specifically built to capture long range dependencies while growth in

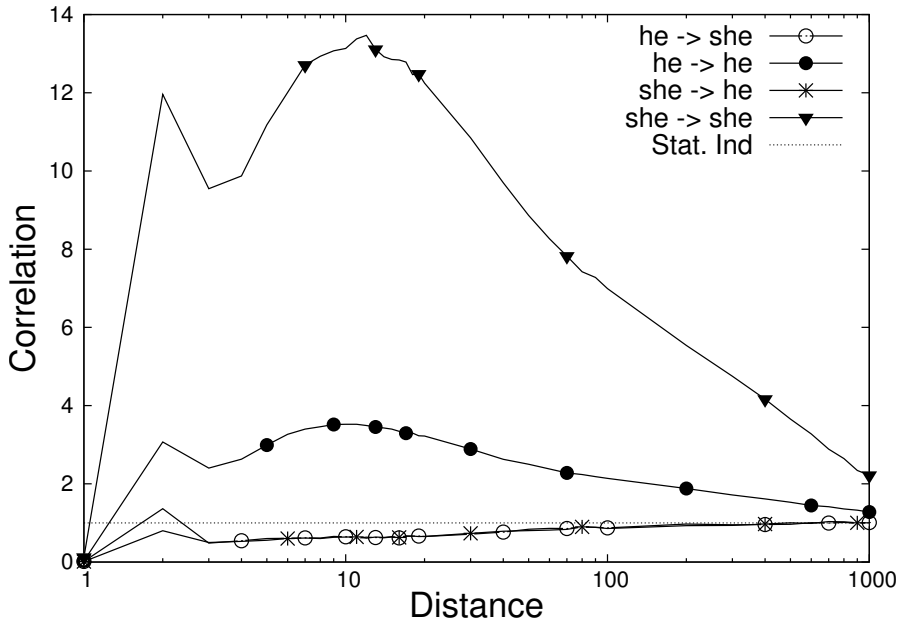


Figure 5.1: Variation of word triggering correlations for pronouns over large distances

number of parameters remains sub-linear in vocabulary size. CDLMs outperform large-context-size Skip models, which are not constrained this way. Additionally, CDLMs show a more robust variation of performance metric against the variation of meta-parameters than RNNLMs, and they allow us to study the sparseness of word representations over different context sizes.

In the rest of the paper, we first briefly describe Skip models and RNNLMs and their limitations in Section 5.2, leading up to the detailed description of our new modelling technique in Section 5.2.3. We then set up experiments to analyze performance of these models in Section 5.3. Section 5.4 gives a robustness analysis of our model in addition to perplexity results for comparing the performance of various LM types and finally, Section 5.5 gives some concluding remarks.

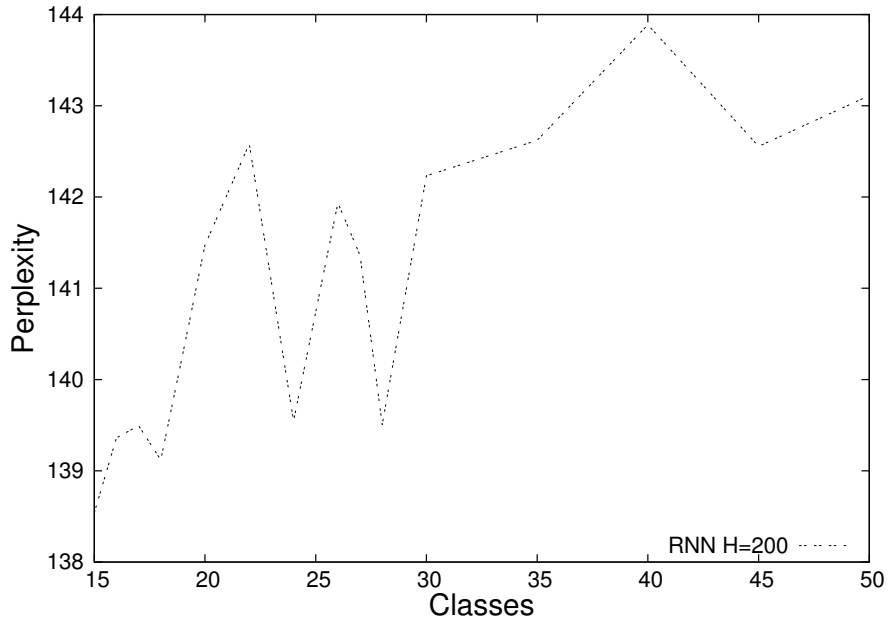


Figure 5.2: Variation of perplexity against the number of classes for a RNNLM with 200 hidden nodes

5.2 Language models

In this section, we first briefly describe and outline the numbers of parameters needed by Skip models and RNNLMs for handling long range dependencies. We then describe our novel CDLM which has been designed to overcome the limitations of skip models by reducing the number of parameters.

5.2.1 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) each time the context size is increased by one, making them computationally inefficient. In addition, the skip modeling framework lacks representational power

when compared to neural network based LMs.

For our experiments, we build skip models by combining unified-smoothing trigrams and distance bigrams, which extend the range. Previously, such a combination has been shown to outperform state-of-the-art smoothed N-gram LMs (Singh and Klakow (2013)).

5.2.2 RNNLMs

RNNLMs provide impressive performance gains when compared to other state-of-the-art 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 (2001)). Unfortunately, using classes for normalization complicates the training process, since it creates a particularly volatile meta parameter. This can be observed in Fig. 5.2, where even for a small variation in classes, RNNLMs show unstable variation in perplexity.

In our experiments, we employ a widely used class-based RNNLM implementation (Mikolov et al. (2011)) 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. To produce better RNNLMs, we can increase the hidden layer size by one which in turn increases the number of parameters linearly in vocabulary size ($O(V + H + C)$).

5.2.3 Custom Decay Language Models

Our new modelling scheme was inspired by log-linear language models, which are characterized by sub-linear growth in the number of parameters with context size (Klakow (1998)). This model consists 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 :

$$\begin{aligned} P(w_{i+1}|h_i) = & \quad \frac{1}{Z(h_i)} \times P_{3\text{-}gram}(w_{i+1}|w_{i-1}, w_i) \\ & \times \quad e^{(E^{w_{i+1}} v_{w_{i-2}} + \sum_{k=i-N+2}^{i-3} E^{w_{i+1}} T_k v_{w_k})} \end{aligned} \tag{5.1}$$

where i is the position in the document, $P_{3\text{-}gram}$ is a standard trigram 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 5.1, 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, we log-linearly combine the long-distance component with an N-gram LM.

In order to estimate the parameters E, v_{w_k} and T_k, we use the stochastic gradient descent algorithm and minimize the training perplexity of CDLM.

5.3 Language Modeling Experiments

5.3.1 Corpus

We trained and evaluated the LMs on the Penn Treebank as preprocessed in Charniak (2001). We used the traditional divisions of the corpus: sections 0-20 for training (925K tokens), sections 21-22 for parameter setting (development: 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 has become 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. We expect our results would scale for larger datasets.

5.3.2 Experimental Setup

In our experiments, we use perplexity as the performance metric to compare the language modelling techniques described in this paper.

In order to establish the most competitive baselines, the RNNLMs trained in our experiment were optimized for a 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. We 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. They were a unified smoothing trigram (*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 trigram 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 is managed by the adaptive gradient methods (Duchi et al. (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, we first trained a CDLM $M = 4$ and reused its con-

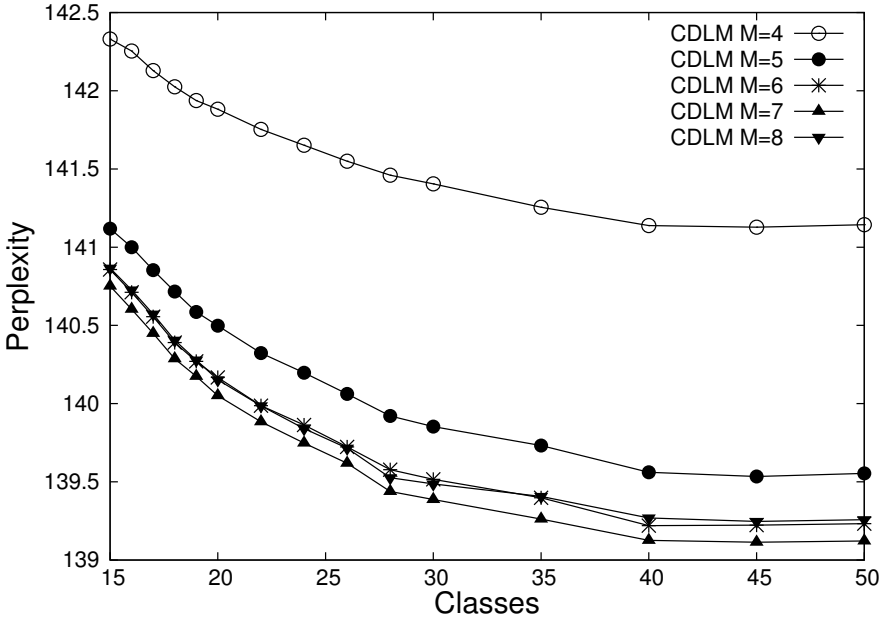


Figure 5.3: Perplexity versus number of classes (C) in CDLM

stituent parameters E and v to build CDLM $M = 5$, only updating T_k while training. This process iterated up to $M = 8$.

5.4 Results and Discussion

5.4.1 CDLM Robustness Analysis

CDLM shows a robust variation of perplexity with changes in classes, as shown in Fig. 5.3. The perplexity values decrease monotonically with increasing classes, as expected since each increase in class creates more parameters that can be tuned. Note that moving from $M = 4$ to $M = 5$ doubles the number of T_k matrices, which caused the large perplexity drop.

Along with the robustness shown by CDLM, the log-linear formulation of CDLM allows us to study and analyze the sparseness of the transformed word space matrices represented by $T_k v_{w_k}$ for different distances. We measure sparse-

Table 5.1: Test set perplexity (PPL) and total number of parameters (PAR) for each language model (LM).

LM	Range	Hidden	PPL	PAR
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

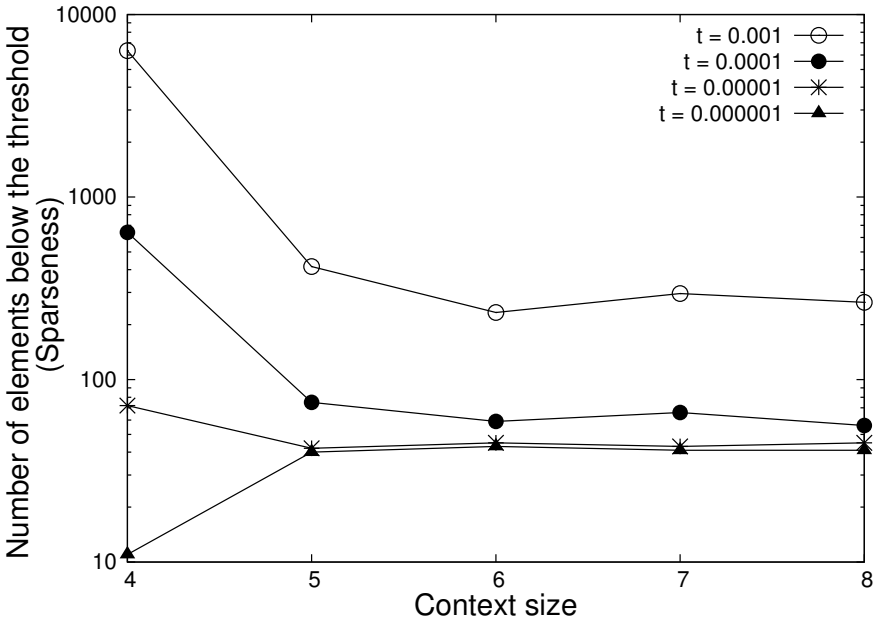


Figure 5.4: Sparseness of CDLM's transformed word space ($T_l v_{w_l}$) measured at different threshold (t) versus its context size

ness by counting the matrix entries below a given threshold. By this measure, a more sparse matrix will have a large number of entries below the threshold than a less sparse matrix. We plot the variation of the sparseness for $T_k v_{w_k}$ matrices for different thresholds against the context size of CDLM in Fig.5.4. In most cases, we observe that as the context size increases the transition matrices have a fewer number of entries below the threshold making them less sparse. Therefore, we believe that this matrix formulation alleviates the sparseness problem and also allows the exponent part to capture latent information.

5.4.2 Perplexity results

Table 5.1 presents our comparison of CDLM with different language models on the basis of their total numbers of parameters and their perplexities. As shown, skip models (*Skip*) outperform the unified smoothing trigram (*UniSt3*) as they have more parameters and hence, they better encode information spread over larger

distances.

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

We further compare CDLMs with RNNLMs. An RNNLM with 145 hidden nodes has about the same number of parameters as *CDLM* and performs 0.1 perplexity points worse than *CDLM*. Increasing the hidden units for RNNLM to 500, we obtain the best performing RNNLM. This comes at a cost of using a lot of parameters. To produce better performing LMs with fewer parameters we constructed an RNNLM with 50 hidden units, which when linearly combined with *CDLM* (*CDLM+RNNLM*) outperforms the best *RNNLM* using less than half as many parameters. It even nominally outperforms the combination of *KN* and *RNNLM* using fewer parameters, but this difference is likely not significant. Combinations of the three different LMs do not result in any large improvements, suggesting that there is redundancy in the information spread over these three types of LMs.

Finally, we note that the increase in parameters does not always lead to better performance. We observe this increase while comparing a *Skip* model with *CDLM* and this increase can be attributed to the richer formulation of *CDLM*. Increase in parameters for *CDLM+KN* does not also lead to a better performance against the fewer-parameters *CDLM+RNNLM*. This is also observed when we compare,

CDLM+KN and *KN+RNNLM*. In this case, we suspect that the lower performance is due to CDLM's lack of recursive connections which form an integral part of RNNLMs. But note that CDLMs, which are not recurrent, can capture much of the long-distance information that the recurrent language models can.

5.5 Conclusion

In this paper, we proposed Custom Decay Language Model, inspired by Skip models' log-linear technique of dividing context into smaller bigrams and then recombining them. In contrast with Skip models, CDLM uses a richer formulation by employing a matrix based exponentiation method to capture long range dependencies. Additionally, CDLM model uses an N-gram model to capture the short range regularities.

Perplexity improvements are observed for CDLM even when compared to Skip models with the larger range and Kneser Ney five-grams. This improvement is observed even though CDLM uses fewer parameters compared to larger range Skip model and $KN5$ with more parameters. In conclusion, CDLM provides a rich formulation for language modeling where the growth of the number of parameters is constrained. We look forward to further enhancing CDLM with recurrent connections and analyzing its performance on other language datasets with a focus on ASR tasks.

Part III

Handling rare words

Chapter 6

Sub-Word Similarity-based Search for Rare-Word Embeddings

6.1 Introduction

Previously, we investigated long-span language modelling techniques which enabled coverage of larger histories. These larger histories helped improve performance on intrinsic and extrinsic language modelling tasks. In such tasks on low-resource languages, however, language models face the problem of handling *rare words* (i.e. words that generally occur with low frequency). In this Chapter, we describe methods to handle rare words by generating good representations cheaply which can then be used in language models to be applied in various tasks.

Recently word representations, also known as word embeddings, have been successfully applied to various 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)), having shown competitive performance in comparison to

the state-of-the-art methods. This competitive performance is ascribed to embeddings' ability to substitute common NLP features and leverage large amounts of data for various NLP tasks.

Even though most of these embeddings are trained on large amounts of data allowing them to have good coverage of a language's vocabulary, however, they still miss out on words when these embeddings are used in tasks with a lot of *rare words*. To alleviate this lack of embeddings for rare words, morphology ((Luong et al., 2013; Botha and Blunsom, 2014; Soricut and Och, 2015)) based methods have been used and these have shown impressive performance gains over methods which ignore such rare words.

Among these morphology-based approaches, the ones proposed by (Luong et al., 2013; Botha and Blunsom, 2014) generate features using a morphological analyzer, namely Morfessor ((Creutz and Lagus, 2005)). These Morfessor based features are then combined to form word vector representations. In contrast, (Soricut and Och, 2015) applies an automatic method to induce morphological rules and transformations as vectors in the same embedding space. More specifically, they exploit automatically learned prefix and suffix based rules using the frequency of such transformations in the data and induce a morphological relationship based graph for words. A search is then performed on this graph for rules, which best explain the morphology of a rare word. The embedding is then estimated using these rare-word explaining rules. In this method, creating and tuning this morphological graph can be a big overhead.

In order to overcome this overhead and still be able to automatically induce rare word representations, we propose a sub-word similarity based search. This technique maps a rare word to a set of its morphologically similar words and combines the embeddings of these similar words to generate this rare word's representation (further discussed in Section 6.2). These generated embeddings can then be augmented to existing word embeddings to be applied in various tasks.

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 6.1: This table reports various statistics for different language datasets used for language modelling. The last column shows the coverage of our method in percentage.

To evaluate these augmented embeddings, we apply it to word similarity tasks (Section 6.2.2). We further instantiate log-bilinear language model ((Mnih and Hinton, 2007)) with augmented word embeddings (Section 6.3) and analyze their performance on rare words over various language modelling datasets (Section 6.3.1). Finally, we summarise our findings in Section 6.4.

6.2 Rare-Word Embeddings

Rare words can form a large part of a task's vocabulary. This is illustrated in Table 6.1, which reports the vocabulary size and number of rare words with zero or one training set frequency (RW). As can be seen in this table, these rare words constitute 10%-50% portions of the vocabulary, thus, making it essential for different tasks to handle them to obtain better performance.

In the context of word embeddings related tasks, training good word embeddings can incur huge computational costs (Al-Rfou et al., 2013) and thus, we focus on applying readily available sources rather than creating them. To increase the availability of resources for many languages, Al-Rfou et al. (2013) provide access[1] to pre-trained word embeddings for more than a hundred languages. These

[1] `https://sites.google.com/site/rmyeid/projects/polyglot`

Task	V	#ENF	Coverage
Rare Word Luong et al. (2013)	2951	1073	100
Gur65 Gurevych (2005)	49	4	100
Rare Word + Google News	2951	173	100

Table 6.2: This table reports various statistics of a few language word similarity datasets used in our experiments. The last column shows the coverage of our method in percentage.

pre-trained word embeddings, namely *Polyglot*, are constructed by applying the method outlined in Bengio et al. (2009) on Wikipedia text, which vary in size from millions of tokens to a few billion tokens.

Among other available pre-trained word embeddings, Google also provides word2vec (Mikolov et al., 2013) based embeddings[2] trained on Google's English News dataset (about 100 billion tokens). In our experiments, we apply both these embeddings set to jump start generating the rare word embeddings for different languages.

6.2.1 Inducing Rare-Word Embeddings

The statistics about various language modelling datasets and word similarity tasks we use in our experiments are shown in Table 6.1 and Table 6.2, respectively. In these tables, along with the vocabulary size and number of rare words (RW), we also report the number of words for which the embeddings were not found (ENF = Embedding Not Found) in the pre-trained embedding sets. For most of the language and pre-trained embedding pairs, number of ENFs form a large share of the vocabulary for word similarity tasks and of rare-word set size for language modelling tasks, hence, we estimate the missing word embeddings before using

[2] `https://code.google.com/archive/p/word2vec/`

them in our tasks.

We first provide the high level description of the steps of our algorithm to induce the word embeddings for these missing rare words, followed by detailed description of each step. For a language dataset with finite vocabulary V and a finite set of given rare words $RW = \{w|w \notin V\}$, we apply the following steps:

1. Map every word $w \in V$ to its sub-word features
2. Index $w \in V$ using its sub-word features
3. Search the index for matches of $w' \in RW$
4. For every $w' \in RW$, combine matched words' embeddings to generate its embedding

More details about these steps are discussed next.

Step 1: Map words to sub-words

When rare words are encountered, it is easier to find similar sub-word units than the word itself. Hence, we start by creating such sub-word units by breaking down each word $w \in V$ into it's constituent N-sized sub-word units: $D_N(w)$. For example, given the sub-word size $N = 3$, the $D_N(language)$ is defined as:

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

In our experiments, we worked with value of $N = 3$.

Step 2: Index word using its sub-words

The pre-trained set of embeddings can have a lot words (for example, *Polyglot* embeddings have 100K words in its vocabulary) and can lead to a lot comparisons to look for matching sub-word units of the rare word. To speed up the search for sub-word units of these words, we create an inverted index on words. For each $w \in V$, we treat $D_N(w)$ as a document and feed it into a search engine based

indexer. In this work, we use Lucene[3] McCandless et al. (2010) to index the words.

Step 3: Search for matches of a rare word

Next, we break the rare word $w' \notin V$ into its sub-word units ($D_N(w')$) and search for $D_N(w')$ on the index. We restrict the search results set to top K results, denoted by $R^K(w')$. $R^K(w')$ now contains words having similar sub-word units as w', hence, containing words which are morphologically similar to w'. In our experiments, we fix $K = 10$ to consider only the top ten results.

Step 4: Generating rare-word embeddings

To estimate the word embedding of $w' \in RW$, we perform the weighted average of embeddings (v) of the rare-word matches. To perform the weighted average, we employ a string similarity function S, such that

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

The above method particularly hinges on the third step, where we utilise sub-word similarity of morphologically similar words to search for rare word alternatives, leading to embedding combination in the fourth step. Hence, we refer to the above technique as **S**ub-**Word** **S**imilarity based **S**earch (**SWORDSS**: pronounced swordz). The SWORDSS embeddings ($\{v_{w'} : w' \in RW\}$) are used along with $\{v_w : w \in V\}$ to perform rare word-related tasks.

In the fourth step, we apply different string similarity functions (S) to average different embeddings of matches from the third step, described in the list below. These different similarity functions help provide a more morphologically enriched scoring of matches and eventually are used in combination to generate a 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

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

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) is a variation on Jaro similarity and quite importantly, it deems two strings more similar if a short prefix (at the beginning of the strings) appears on both strings. This is especially relevant for our 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) considers 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 true morphological 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 meta parameters allow the metric some flexibility that the others do not.

In our experiments on rare-word related tasks, we mostly observe that using SWORDSS leads to high coverage rates, also presented in Table 6.1 and Table 6.2. We note that whenever words w' result in zero matches in our experiments, they were either removed completely (in case of word similarity tasks) or substituted with random vectors (in case of language modelling tasks, Section 6.3).

6.2.2 Word Similarity Task

To test the efficacy of SWORDSS embeddings, we evaluate them on standard word similarity tasks. In such tasks, the correlation between the human annotator ratings of word pairs and the scores generated using embeddings, is calculated. A good set of embeddings would lead to high correlation rates.

Specifically, we evaluate the SWORDSS embeddings on Luong et al. (2013)'s English Rare Words dataset with 2034 word pairs (Luong2034) and also evaluate these embeddings on German word similarity task Gurevych (2005) with 65 word pairs (Gur65).

6.2.3 Experimental Setup

For the German word similarity task, we use only *Polyglot* word embeddings, which are 64-dimensional vectors. For English along with *Polyglot* word embeddings, we use the Google News word2vec embeddings, which are 300-dimensional vectors.

As a baseline we use the existing pre-trained word embeddings, which are compared to its augmented SWORDSS version. While augmenting the pre-trained set with the SWORDSS embeddings, we also explore various string similarity functions to be used in the fourth step (Section 6.2.1), namely, Jaccard Index (SWORDSS_{ji}), Jaro similarity (SWORDSS_{jaro}), Jaro-Winkler similarity (SWORDSS_{jw}), Most Frequent K Characters similarity (SWORDSS_{mfk}), Subsequence Kernels (SWORDSS_{ssk}) and Tversky Coefficient (SWORDSS_{tc}).

Word Vectors	Gur65
Polyglot	28.5
Polyglot+SWORDSS$_{ji}$	37.5
Polyglot+SWORDSS$_{jaro}$	37.1
Polyglot+SWORDSS$_{jw}$	37
Polyglot+SWORDSS$_{mfk}$	37.2
Polyglot+SWORDSS$_{ssk}$	36.9
Polyglot+SWORDSS$_{tc}$	**37.6**
Polyglot+SWORDSS$_1$	35.8

Table 6.3: Spearman's rank correlation (%) based evaluation of various string similarity functions used to generate augmented word vectors for German word similarity task (Gur65)

To evaluate the effect of these string similarity functions, we also compare them to constant function ($S(w, w') = 1$, where w and w' are words) used in the fourth step, denoting the corresponding embeddings by SWORDSS$_1$. Finally, we also compare the SWORDSS embeddings to SO2015 Soricut and Och (2015), which also applies morphological analysis to generate missing word embeddings quite similar to SWORDSS embeddings.

6.2.4 Results

Using SWORDSS embeddings definitely improves correlation rate in comparison to the original on the Gur65 task (shown in Table 6.3, though all the different string similarity functions except the constant function (SWORDSS$_1$) lie in a very close range. Henceforth, we treat all the string similarity functions except the constant function as one (SWORDSS$_{sim}$) and report the best correlation value after applying these functions.

Word Vectors	Luong2034	Gur350	Gur65	ZG222	Es353	Ro353	Ar353
BB2014 w/o morph	18	36	-	6	26	-	-
BB2014 w/ morph	30	56	-	**25**	28	-	-
SO2015 w/o morph	44.7	62.4	-	16.6	36.5	51.3	37.1
SO2015 w/ morph	**52**	**64.1**	-	21.5	**47.3**	**53.1**	**43.1**
Polyglot	9.7	25	28.5	21.3	14.0	28.1	8.4
Polyglot+SWORDSS$_1$	28.9	33.2	35.8	23	23.2	18	14.6
Polyglot+SWORDSS$_{sim}$	30.4	33.7	**37.6**	24.8	25.9	18.3	14.5

Table 6.4: Spearman's rank correlation (%) based evaluation of techniques with and without morphological features used to generate representations for word similarity task.

Word Vectors	Luong2034
SO2015 w/ Morph	**52**
Google News	45.3
GoogleNews+SWORDSS$_1$	51.3
GoogleNews+SWORDSS$_{sim}$	51.4

Table 6.5: Spearman's rank correlation (%) based evaluation of augmented Google News word2vec word vectors on the English Rare Word Similarity task

SWORDSS versions of *Polyglot* embeddings when compared to the original, in almost all word similarity tasks lead to a better correlation rate. Though, for each word similarity task the difference between SWORDSS$_1$ and SWORDSS$_{sim}$ remains small. These improvements, however, are still lower than the correlation rates reported by SO2015. This is due to the difference in initial quality of embeddings used by each method. As SWORDSS uses *Polyglot* embeddings trained on lesser amount of data than SO2015, it is easily outperformed.

In Table 6.5, we alleviate this lower performance issue by replicating our ex-

periment using Google News word2vec embeddings to jump start the SWORDSS versions for the Luong2034 task. Using these embeddings, trained on a larger dataset than used by *Polyglot*, leads to SWORDSS versions having on par results with SO2015 result for the Luong2034 task.

Overall SWORDSS technique is able to drastically improve *Polyglot* embeddings across almost all word similarity tasks. Even though SWORDSS embeddings' ability to compete with SO2015 is marred by the quality initial set of embeddings, it provides a simpler sub-word search based alternative to a search over the graph of morphological relationships performed by SO2015.

6.3 Word Embeddings in Language Models

Training language models (LMs) using an expanded vocabulary (having more word types than contained in the training corpus) leads to having words which are not present in the training set. These rare words are represented by a default value of probability conventional N-grams and long short term memory (LSTM) based reccurrent neural networks LM (Sundermeyer et al. (2012)). This is usually not beneficial for Keyword Search and Automatic Speech Recognition systems made for low resourced languages, since presence of rare words in speech queries is high (Logan et al. (1996, 2005)).

To avoid this misrepresentation of rare words, we apply SWORDSS embeddings in a language modelling framework. We investigate this further by incorporating the augmented pre-trained word embeddings in the log-bilinear language model (LBL) (Mnih and Hinton (2007)).

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

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

where $C_j \in \mathbf{R}^{d \times d}$ 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 next word representations (r_w) are initialised with the pre-trained word embeddings. Thereafter, we train 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 we still choose the original LBL because of the ease with which it incorporates pre-trained embeddings in its formulation.

6.3.1 Language Model Evaluation

Datasets

To evaluate the SWORDSS embeddings for language modelling, we use the European parliament dataset of German (de) language as processed by Botha and Blunsom (2014). We also perform the language modelling experiments with the SWORDSS embeddings on the Tagalog (tl), Turkish (tr) and Vietnamese (vi) datasets, 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 dataset is processed to have no OOVs, however, it still has a lot of low-frequency words (refer Table 6.2). Whereas the Babel datasets have OOVs

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 6.6: Statistical summary of datasets used for the language modelling experiments. Information corresponding to a language is presented in a column.

as well as other low-frequency words.

The Babel datasets are provided with training and development sets only. We divide the existing development set into two halves to use one as the test set and the other half as the new development set. Further statistics on the datasets are summarised in the Table 6.6.

Earlier in Tables 6.1 & 6.2, we had shown that even though a lot of rare-word embeddings are missing from the pre-trained set, SWORDSS is able to generate and obtain high coverage rates for such words, making it suitable to use in the context of rare words.

Experimental Setup

Before evaluating the SWORDSS embeddings for predicting rare words, we use all the OOVs to expand the corresponding vocabulary. SWORDSS embeddings for all the words in the expanded vocabulary are used to initialise LBL framework as described in Section 6.3. A bigram version of this LBL ($\text{LBL2}_{SWORDSS}$) is further trained on language datasets before being evaluated.

We compare the $\text{LBL2}_{SWORDSS}$ model with the conventional Modified-Kneser-Ney five-gram LM (MKN5) (Kneser and Ney (1995); Chen and Goodman (1996)) and also with the bigram (LBL2) based log-bilinear LM. As a more powerful base-

Language Model	German		Tagalog		Turkish		Vietnamese	
	PPL	RW1PPL	PPL	RW1PPL	PPL	RW1PPL	PPL	RW1PPL
MKN5	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[4]	323.1	596K	134.7	343K	489.8	110K	**102.1**	457K
Char-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
#PAR	4.7 M		2.9 M		3.2 M		0.8 M	

Table 6.7: Perplexities on test set (PPL), RW1 perplexities (RW1PPL) in thousands and number of parameters (#PAR) for LBL and LSTM based language models (LM) in millions, presented on four language datasets

line, we also train an LSTM based RNN LM to compare with LBL2$_{SWORDSS}$. Moreover, we compare the LBL2$_{SWORDSS}$, with a character aware language model (Kim et al. (2015)), denoted as Char-LSTM. The Char-LSTMs are 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, we use the expanded vocabulary as used by LBL2$_{SWORDSS}$. While training neural network based language models, we restrict the number of parameters to have a similar number of parameters as LBL2$_{SWORDSS}$.

Perplexity Experiments

We compare the language models described in Section 6.3.1 using perplexity values calculated on test sets of different languages. The results of the evaluation are presented in the Table 7.1.

As shown in Table 7.1, LBL2$_{SWORDSS}$ is able to outperform the conventional LBL2 comfortably on all the datasets except on Vietnamese. For Vietnamese, LBL2$_{SWORDSS}$ shows an on par performance to LBL2. Due to SWORDSS' low coverage of Vietnamese vocabulary, initialising LBL2 with SWORDSS embedding leads to marginal performance gain.

Overall in terms of test set perplexity, Char-LSTM outperform LBL2$_{SWORDSS}$ comfortably on most language datasets. Though, on Vietnamese (which in written form is character sounds represented as words separated by spaces) Char-LSTM suffers and LSTM outperforms the other language model. In comparison to LSTM and Char-LSTM, LBL2$_{SWORDSS}$'s lower performance on test data is expected as the former are more powerful language models.

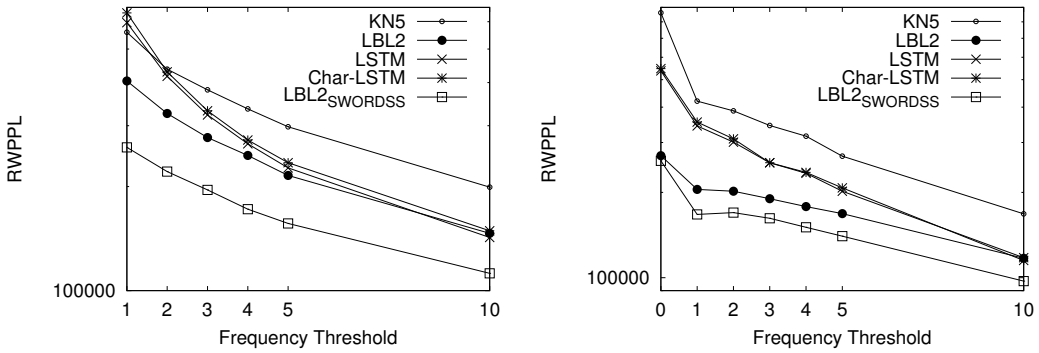
However, for tasks like Keyword Search, having low perplexities on most frequent set of words is not good enough and hence, we compare LMs on the perplexity of rare-word based test set. To perform this comparison, we compute perplexity only on rare words (RW1PPL), with training-set frequency of one, present in the test set. As shown in Table 7.1, we observe that LBL2$_{SWORDSS}$ performs better than the LSTM based LMs across various languages in terms of RW1PPL.

We also note that Char-LSTM, do not have a straightforward way of including SWORDSS embeddings. Hence, they are not directly comparable to LBL2$_{SWORDSS}$, as the latter has more information at its disposal. We also note, that when LSTM's input layer are initialised with SWORDSS augmented embeddings, it obtains the same perplexity values as the LSTM initialised with conventional random embeddings. This observation suggests that the LBL framework is better suited for this naïve way of initialising neural language models with SWORDSS embeddings and improving perplexity on rare words.

Performance on OOVs and Rare Words

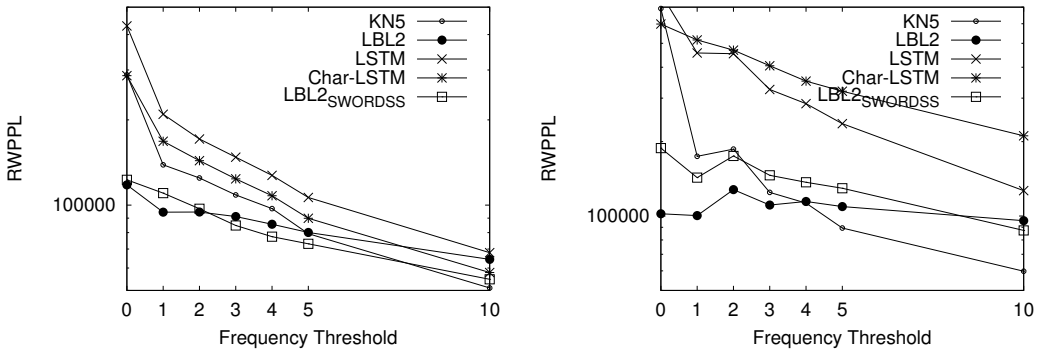
To further compare the performance of aforementioned language models on rare words, we analyze perplexities of such words (RWPPL) in the test set as a variation of the frequency classes of these words in the training set. This variation is displayed in Figures 6.1 & 6.2.

For OOV words (rare words with zero training-set frequency), LBL2$_{SWORDSS}$ outperforms the other language models built with a similar number of parameters,



(a) German **(b)** Tagalog

Figure 6.1: Variation of rare-word perplexity versus threshold on frequency of training-set words on German and Tagalog datasets



(a) Turkish **(b)** Vietnamese

Figure 6.2: Variation of rare-word perplexity versus threshold on frequency of training-set words on Turkish and Vietnamese datasets

on the Tagalog and Turkish datasets. In these cases, LBL2$_{SWORDSS}$ reduces rare-word perplexities by a factor of two over the character-feature rich Char-LSTM, whose design allows it to implicitly handle rare words.

Even for rare words with training set frequency up to one, LBL2$_{SWORDSS}$ reduces perplexity up to a factor of 2.5 times with respect to Char-LSTM, on the

[4]when initialised with SWORDSS embeddings it obtains the same perplexity values

German, Tagalog and Turkish datasets. Interestingly on these particular language datasets, Figures 6.1 & 6.2 show that LBL also performs better than both the LSTM based LMs in modelling OOV and rare words of frequency up to ten.

For Vietnamese, LBL alone is able to improve OOV and RW1 words over the other LMs. We allude this to low coverage of Vietnamese rare words by SWORDSS than for other languages, where adding SWORDSS embeddings leads to harm the prediction of OOV and RW1 words.

These perplexity improvements start to wane when higher frequency words are included into the rare word set, across the different languages. Nevertheless, in languages with morphological information, initialising LBL with SWORDSS embeddings reduces perplexities on rare words.

6.4 Conclusion

In this paper, we introduced SWORDSS, a novel sub-word similarity based search for generating rare word embeddings. It leveraged the sub-word similarity in morphologically rich languages to search for close matches of a rare word, and then used these close matches to estimate the embedding a rare word.

Even though SWORDSS is an unsupervised approach like Soricut and Och (2015), it differs from latter in the way it utilizes the morphological information. The latter automatically induces morphological rules and transformations to build morphological graphs of words. This graph is then tuned and used to induce embedding of a rare word. Whereas, SWORDSS replaces the overhead of induction of rules and creation of graph by searching a sub-word inverted index to find rare-word matches and combining their embeddings to estimate rare-word embedding.

To judge the SWORDSS technique, we use it to augment pre-trained embeddings and then evaluate the augmented embeddings on word similarity tasks. The augmented embeddings outperform the initial set of embeddings drastically. But,

it lags behind the state-of-the-art performance of Soricut and Och (2015), by using embeddings trained on larger datasets SWORDSS is able to perform on-par with it on a rare-word task.

We also investigate the effects of using SWORDSS augmented embeddings for modelling rare words. To perform this experiment, we train $\text{LBL}_{SWORDSS}$ LM and compare it with language models like character aware LM, LSTM based RNN LM restricted to similar size. Almost on all datasets, character aware LM outperforms the other LMs with respect to perplexity on complete test sets. But on rare words, it shows up to 50 % reduced perplexity values in comparison to other LMs. Hence, suggesting using SWORDSS embeddings prove useful in modelling rare-word tasks.

As part of future work, we plan to study SWORDSS embeddings to augment more complex LMs than LBL and further analyze the different string similarity functions used in SWORDSS's formulation.

Part IV

Combining long-term dependencies and rare word knowledge

Chapter 7

Combined evaluation of techniques for handling long-term dependencies and rare words

7.1 Introduction

In the previous chapters, we have introduced techniques to handle long-term dependencies and rare words to improve performance on low-resource tasks. We have evaluated these two language modelling challenges separately. In this chapter, we perform a combined evaluation of the proposed techniques on IARPA's KeyWord Search task (KWS).

Specifically, we create SWORDSS-based representations for Zulu and incorporate these rare-word representations in long-span models like LSTMs to perform first-pass decoding for the Keyword Search task. We also present intrinsic evaluations to verify the results we have observed earlier. Finally, we conduct a perfor-

mance evaluation based on KWS metrics like actual term weight value (ATWV) and word error rate (WER).

To apply LSTM-based models to the first-pass decoding, we apply the methods discussed in Chapter 4. As these are approximate methods, we also analyze the oracle performance to understand the limitations of these methods.

7.2 Related Work

7.2.1 Sub-word based techniques for Keyword Search

The Keyword Search system's performance on a low-resource language task is highly dependent on handling out-of-vocabulary (OOV) words in the list of keywords. IARPA's Babel project provides a data setup specifically for this low-resource setting and also has been studied earlier (Chen et al. (2013); Lee et al. (2014)). Both of these approaches have concentrated on leveraging acoustic similarity for OOV words to enable keyword retrieval. In both cases, this acoustic similarity information is not transferred to language model. Therefore, comparing and sorting during retrieval step suffers from an impoverished model. We alleviate this problem by enhancing language models with SWORDSS embeddings described in Chapter 6.

In contrast with above approaches, Hartmann et al. (2014) have also compared sub-word similarity lattice-based techniques for OOV handling in a Keyword Search task. The underlying principle of comparing sub-words to search for OOVs is similar to our SWORDSS' technique. However, Hartmann et al. (2014) use back-off smoothing based n-gram models, which suffer from data sparsity issues. To overcome such issues, we apply long-span model based n-gram models for first-pass decoding of our Keyword Search task.

7.2.2 Applying long-span models for Keyword Search

Previously, LSTM-based neural networks have been applied to Keyword Search tasks (Cai and Liu (2016); Wöllmer et al. (2013)). While Cai and Liu (2016) have used LSTMs for training well performing acoustic models, Wöllmer et al. (2013) have applied these models to decoding speech signal for Keyword Search. Our work is similar to the latter in applying LSTM models to the first-pass decoding, but we also improve these models OOV handling capacity by incorporating SWORDSS-based representations.

7.3 Language Models for Keyword Search

IARPA's Babel project focusses on developing a KWS system for low-resourced languages. For these languages, the OOV keywords constitute 50% of keyword lists. To be able to perform well under such restrictive conditions, the KWS pipeline needs to handle such words.

In this section, we concentrate on building advanced language models to cope with this issue in the KWS pipeline. We combine techniques discussed in earlier chapters (Chapters 4 & 6) to perform the first-pass decoding as part of the KWS pipeline.

To allow detection of OOV keywords, we augment language models with OOV keyword representations produced using the SWORDSS technique. The simplest way to perform such an augmentation is to initialise continuous-space models like Log-bilinear LMs or LSTMs with these representations and then carry out the model training.

To use these models to perform a first-pass decoding for the KWS system, we approximate the continuous-space models to n-gram based models allowing us to overcome expensive calculations required for using the full versions in first-pass decoding (For more details refer to Prior Work section in Chapter 4).

7.4 Keyword Search Pipeline

To build the KWS pipeline, we use the example Babel recipes available at the Kaldi repository[1]. Using these recipes we build a Deep Neural Network based acoustic model that is decoded using the language models described in Section 7.5.2.

Finally, a Keyword Search is performed using an instance of the system described in Chen et al. (2013). An important difference is that we use Kaldi-based recipes for ATWV evaluation instead of NIST provided evaluation tools.

7.5 Experiments

In this section, we compare the effectiveness of language modelling techniques combining long-term and rare-word information on IARPA's Keyword Search task, evaluating perplexity, WER and ATWV.

7.5.1 Experimental Setup

For our experiments, we use the IARPA Babel Program Zulu limited language collection release babel206b-v0.1d. This language collection contains 10 hours of transcribed phone conversations, with a lexicon size of 15K words. The collection also provides 1.5 hours of development corpus that we use as our test set.

This language collection contains 2K keywords, with 50% being not present in the training corpus. We choose a lexicon for both the in-vocabulary and out-of-vocabulary words. Choosing a lexicon with pronunciations for OOV words allows the acoustic model to detect these words. Then, we can focus on measuring the impact of training specialized language models for the Keyword Search task, as the decoding process does not suffer from non-detection of OOV words acoustically.

[1] `https://github.com/kaldi-asr/kaldi/tree/master/egs/babel/s5d`

7.5.2 Language Models

We describe the various language models used in our experiments. These include the language models studied in the earlier chapters and other simple baselines.

Trigram LM (KN3) is constructed by SRILM (Stolcke et al. (2011)) using Kneser-Ney smoothing. This forms a popular baseline (Chen et al. (2013); Hartmann et al. (2014)) used for KWS systems for Babel languages.

Keyword Restricted Trigram LM (KW-KN3) is a trigram LM trained similarly to KN3 but only on sentences containing at least one keyword. Since the task is geared towards detection of keywords, restricting training corpus to keyword-specific sentences can boost keyword probability estimates.

Log-Bilinear LM (LBL) as described in (Mnih and Hinton (2007)), is used as another simple baseline. An OOV handling variant ($\text{LBL}_{SWORDSS}$) is constructed in similar fashion to as described in Chapter 6 by initialising weight matrices with SWORDSS-based embeddings.

Long-Short-Term-Memory-based recurrent neural network models (LSTM) are trained on available the language collection. These models allow for capturing long-term dependencies, which have shown benefits in the context of Keyword Search task (Hartmann et al. (2014)). We develop two variants of this model, a regular LSTM and an LSTM initialised with SWORDSS-based embeddings (labeled: $\text{LSTM}_{SWORDSS}$).

Character-Aware LSTM models (Char-LSTM) introduced by Kim et al. (2015) are character-input-based variants of the LSTM models. These models capture long-term dependencies along with handling OOV inputs at the time of inference. Similarly to LSTM models, we compare to a regular variant of this model and also build SWORDSS-based version by initialising hidden-to-output weights matrix (Char-$\text{LSTM}_{SWORDSS}$) with SWORDSS-based embeddings.

To apply the continuous-space language models like the LBL, LSTM and Char-

Language Model	Regular LMs				Approximated LMs		
	PPL	KWPPL	OOV-KWPPL	#PAR	PPL	KWPPL	OOV-KWPPL
KN3	**201.5**	16.3K	100.7K	167K	-	-	-
KW-KN3	215.5	**11.7K**	88.3K	129K	-	-	-
LBL	248.9	20.0K	66.0K	4520K	404.3	57K	307K
$\text{LBL}_{SWORDSS}$	274.7	19.4K	**49.8K**	4520K	312.8	30K	81K
LSTM	216.5	15.5K	59.4K	9774K	288.0	**26K**	**67K**
$\text{LSTM}_{SWORDSS}$	224.5	15.5K	62.2K	9774K	522.0	83K	321K
Char-LSTM	220.5	17.5K	96.7K	6113K	530.8	33K	65K
$\text{Char-LSTM}_{SWORDSS}$	222.4	17.3K	106.3K	6113K	2344.8	216K	933K

Table 7.1: Perplexities and number of parameters (#PAR) of different language models (LM) on Zulu. The Keyword Perplexity (KWPPL) and Out-of-Vocabulary Keyword Perplexity (OOV-KWPPL) are reported in thousands (denoted by K).

LSTM variants, we approximate these models to an n-gram LM using the techniques described in Chapter 4. These models are then applied to first-pass decoding in the KWS pipeline.

7.5.3 Perplexity Experiments

Comparing these models on test set perplexity (Table 7.1), we observe that simple trigram LM (KN3) performs best. This is mostly due to the small amounts of training data available, which leads to overfitting in other models. However, comparing keyword restricted perplexity (KWPPL) and out-of-vocabulary keyword perplexity (OOV-KWPPL), we observe significant differences. Keyword-restricted trigram LM (KW-KN3) performs best in terms of keyword perplexity and $\text{LBL}_{SWORDSS}$ performs best on out-of-vocabulary keyword perplexity.

Additionally, initialising continuous-space long-span models with $SWORDSS$ embeddings does not help, as these models show no change in KWPPL and a degradation in OOV-based KWPPL. These effects we suspect are due to the small

Language Model	Approximated LMs		
	ATWV	OOV-ATWV	WER
KN3	59.0	7.3	**67.8**
KW-KN3	**61.6**	7.9	68.3
LBL	60.0	8.1	72.0
$\text{LBL}_{SWORDSS}$	59.7	**8.3**	71.6
LSTM	60.7	**8.3**	71.8
$\text{LSTM}_{SWORDSS}$	60.5	8.0	72.3
Char-LSTM	59.2	8.1	72.1
$\text{Char-LSTM}_{SWORDSS}$	57.7	7.8	77.6

Table 7.2: ATWV, OOV-ATWV and WER values of language models (LM) on Zulu. These metric values are reported as percentages.

amounts of training data available to these complex models.

Approximating the continuous-space models to n-gram LMs is similar to effects of pruning an n-gram LM. This pruning for continuous-space models initialised with SWORDSS leads to large losses in KWPPL as the relevant contexts are removed from the approximated n-grams. In contrast, regular continuous-space models show smaller degradation where this pruning affects only more frequent contexts. Generally, however, this approximation of models shows degradation in different types of perplexities.

7.5.4 ATWV Experiments

In Table 7.2, we report the ATWV and Out-of-Vocabulary ATWV (OOV-ATWV) results when applying these models in KWS system for decoding. In most cases, continuous-space models perform better than regular trigram on different versions of ATWVs. On word-error-rates (WER), however, we observe that any keyword

Language Model	Approximated LMs		
	OOVPPL	OOV-ATWV	WER
LBL	308K	8.1	72
+ 1000-best	309K	7.6	**71.1**
$\text{LBL}_{SWORDSS}$	**81K**	**8.3**	71.6
+ 1000-best	85K	7.8	71.5

Table 7.3: OOVPPL, OOV-ATWV and WER values of language models (LM) for Zulu. Even rows present the metric values when 1000-best list-based n-grams are used for approximation. The OOV-ATWV and WER values are reported as percentages.

restriction during training or usage of continuous-space models further degrades the Speech Recognition results.

Also, long-span models generally perform on-par with smaller context models. The *3*-gram-based LBL models perform on par with *5*-gram-based LSTM models. This is mostly due to the approximation process of continuous-space models, where smaller contexts are not predicted well by these long-span models. Overall, a simple keyword-restricted trigram LM (KW-KN3) performs best in terms of ATWV and SWORDSS-based LBL ($\text{LBL}_{SWORDSS}$) performs best on OOV-ATWV. Additionally, continuous-space models usually improve the OOV-ATWV over the trigram models KN3 and KW-KN3, but at the cost of degrading the overall AWTV performance.

7.6 Discussion

After reporting the results in the previous section, we discuss and justify these observations in the following section.

7.6.1 KWS on Zulu

The low Speech Recognition performance on Zulu makes KWS a hard task and this effect is mostly due to the small amount of training data. Also, as we develop more advanced models geared towards keyword detection, the Speech Recognition performance degrades while the keyword detection improves.

7.6.2 1000-best n-grams for Approximating Continuous-Space Models

Previously in Chapter 4, we had observed improved Speech Recognition performance by utilizing 1000-best lists-based n-grams for approximation of continuous-space models. Similar experiments show improvements in WER performance, as shown in Table 7.3. But, using these 1000-best n-grams show degradation in the ATWV performance. We suspect this is because the model assigns lower scores to keyword-specific hypotheses, which degrades ATWV performance.

7.6.3 Using Approximated LMs for First-Pass

During our KWS experiments, we created approximated versions of continuous-space models. As pointed out in the earlier sections, this approximation is a pruning step of probabilities present in the original language model. The pruning choices are guided by the frequency of n-grams in the training set and hence, rare-word contexts not present in the training set do not appear in the approximated language models. This lack of such contexts is reflected in low performance of these approximated language models on keyword-restricted perplexity and ATWV results.

Especially for the SWORDSS-based model, such effects are pronounced as these model's approximated version lose rare-word contexts , which the SWORDSS-

based models are better at predicting that regular models.

7.7 Summary

In this chapter, we present a combined evaluation of models described in this thesis. We applied SWORDSS-based continuous-space models to first-pass decoding in the Keyword Search Task.

Combining techniques discussed in Chapter 4 and 6, we evaluated ATWV on keywords and out-of-vocabulary keywords. Our smaller-context SWORDSS-based models performed on par with Approximated long-span models. We believe that this effect was mostly due to losses incurred during approximation. Nevertheless, this chapter outlines ia novel attempt to apply neural-network-based long-span models for first-pass decoding in the KWS task.

For future work, we hope to introduce keyword-restrictions to long-span models. Such restrictions helped regular trigram language model to perform competitively with long-span models and will be interesting to investigate further.

Part V

Conclusion and outlook

Chapter 8

Conclusion & Future Work

8.1 Conclusion

For low-resource tasks like the Babel project and language model adaptation for Speech Recognition, handling the low amount of language resources becomes a key challenge. Among various challenges posed by such tasks, we specifically looked at handling long-span information and handling low-frequency words (rare words).

8.1.1 Handling Long-Term Information

Recent work (Deoras et al. (2011b,a); Cho et al. (2014)) has shown that long-span information can help improve performance in language modelling tasks like Speech Recognition and Machine Translation. In these tasks, language models are used to reduce the search space of hypotheses. By leveraging the long-span information, these models can further reduce the search space of hypotheses by scoring the correct hypothesis higher than it counterparts.

To reap these benefits in low-resource tasks, we re-evaluated existing techniques

on language modelling tasks. Among the existing techniques, we enhanced the conventional n-gram styled skip-gram models using a Unified Smoothing technique. This smoothing technique enabled better capturing of information over longer context sizes. We compared these enhanced skip models with contemporary recurrent neural network language models (RNNLM), known for implicitly infinite history sizes, on the basis of perplexity on limited-resource Babel datasets. On these datasets, enhanced skip models outperformed the RNNLM comfortably. On the Keyword Search task, however, these enhanced skip models performed comparably with regular trigram models, which are known to be outperformed by neural network methods. Next, we concentrated on using such neural network based methods for capturing long-span information than enhancing skip-grams for this purpose.

Language model adaptation for Speech Recognition is another low-resource task, where there is not enough in-domain data to train a Speech Recognition system on the data. Hence, large amounts of out-domain data are used to train background acoustic and language models. The language model is then adapted using the in-domain data and used to perform decoding in the Speech Recognition system.

In this context, using long-span neural network based methods like LSTM-based LM can be beneficial, however, applying these methods can result in prohibitive amounts of computations. To alleviate this issue, we proposed an approximation to LSTMs by scoring sampled n-grams with this model. Faster than the existing techniques, this approximation technique made first-pass decoding with LSTMs possible. To apply these approximated LSTMs to language modelling adaptation for the Metalogue Speech Recognition task, we extended the underlying architectures to produce adapted-approximated LSTMs. Performing first-pass decoding with these LSTMs, we were able to perform reasonably better than other language modelling adaptation techniques.

Aforementioned techniques like skip models and RNNLMs, both capture long-

term dependencies differently. Skip models enumerate these dependencies and RNNLMs form an abstract representation of these dependencies. Though RNNLMs outperform the skip models, the neural network based model do not form a good framework to study the long-term dependencies due to their abstract nature. To counter this effect, we introduced the custom decay language models (CDLM). These models not only enumerate the long-term dependencies but leverage vector representations for each such dependency to help study them better. Analyzing the long-term dependencies in the CDLM framework, we contrast its power with RNNLMs. We observed that a six-gram CDLM is able to perform comparably with a similar sized RNNLM. Further analysis revealed that the longer-term dependencies have denser vector representations storing more global than local information.

8.1.2 Handling Rare Words

Training good word embeddings for a language requires large amounts of data. Even then the language can come up with words not present in the training set leaving the words without embeddings. This lack is detrimental to building systems for low-resource languages. To overcome this lack of embeddings for such rare words, existing methods leverage their morphological features to generate their embeddings. While the existing methods use computationally-intensive rule-based (Soricut and Och (2015)) or tool-based (Botha and Blunsom (2014)) morphological analysis to generate embeddings.

In contrast to such techniques, we performed a computationally simpler sub-word search on words that have existing embeddings. Embeddings of the sub-word search results were then combined using string similarity functions to generate rare-word embeddings. Using these rare-word embeddings to augment the pre-existing embeddings, we evaluated them on various word similarity tasks. On most of the word similarity tasks, the augmented set of embeddings were able to outperform the existing set of embeddings by up to three times improvement in correlation

rates.

Further to evaluate these rare-word embeddings in the framework of language modelling, we inserted these embeddings into existing language modelling techniques. With these language models, we observed up to 50% reduction in rare-word perplexity in comparison to other more complex language models.

8.1.3 Combined Evaluation

Working towards handling long-term dependencies and rare-word information better in language models, we performed a combined evaluation on the Babel program's Keyword Search task. Previous such evaluations have used sub-word based regular n-grams, however, in our work we evaluated sub-word based recurrent neural network language models for first-pass decoding in the KWS task.

In our experiments, smaller context-based sub-word language models performed on par with long-span neural network models. This was mostly due to the approximation of long-span models which led to degradation in these model's performance. Overall SWORDSS-based augmentation improved results for smaller context models but degraded the system's performance for long-span models.

8.2 Outlook

This section briefly outlines possible extensions of methods presented in this thesis and other possible scenarios related to these methods which may be worthwhile examining in future work:

- **Capturing sentential-based dependencies in continuous-space language models:** As part of our efforts to handle long-range dependencies, we developed models to capture long-range dependencies within a sentence. However, these models are still shorted-sighted as long-range dependencies

occur much beyond a sentence boundary, spanning multiple sentences. Earlier, Momtazi and Klakow (2011) have shown the benefits of using such sentential context in an n-gram language model and applying them to question answering task. It would be interesting to incorporate such dependencies in continuous-space language models like neural network and evaluating them on tasks like question answering and summarization.

- **Analysing usage of various sub-word models:** For handling rare words, in Chapter 6 we used an ad hoc sized unit to perform a search for rare words without embeddings. A deeper analysis of using differently sized sub-word units and also leveraging morpheme-based sub-word units could be performed. This will help us understand the impact of choosing the sub-word units on the quality of the rare-word embeddings.

- **Analysing effects of different similarity functions:** Also for handling rare words as part of our work (Chapter 6), we found that using language-dependent orthographic similarity functions did not affect the rare-word embeddings quality much in comparison to language-agnostic ones. This aberration in our experiments could be a topic of analysis, which explores the reasons of such functions not working.

- **Developing keyword-accurate approximations to LSTMs:** A major hurdle while approximating LSTMs, trained with rare-word information, is accurately capturing the rare-word knowledge in the approximated language model. In our combined evaluations (Chapter 7), we observed that rare-word information augmented LSTMs degrading in performance after approximation. Further work is required to create approximated LSTMs keep the rare-word knowledge intact while also not losing much on the overall performance.

List of Figures

List of Tables

Bibliography

Heike Adel, Katrin Kirchhoff, Ngoc Thang Vu, Dominic Telaar, and Tanja Schultz. Comparing approaches to convert recurrent neural networks into backoff language models for efficient decoding. In *INTERSPEECH*, pages 651–655, 2014.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-3520`.

Ebru Arisoy, Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. Deep 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 20–28. Association for Computational Linguistics, 2012.

Ebru Arisoy, Stanley F Chen, Bhuvana Ramabhadran, and Abhinav Sethy. Converting neural network language models into back-off language models for effi-

cient decoding in automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):184–192, 2014.

Jerome R Bellegarda. Statistical language model adaptation: review and perspectives. *Speech communication*, 42(1):93–108, 2004.

Samy Bengio and Georg Heigold. Word embeddings for speech recognition. 2014.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155, 2003.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553380. URL `http://doi.acm.org/10.1145/1553374.1553380`.

Jan A. Botha and Phil Blunsom. Compositional morphology for word representations and language modelling. *CoRR*, abs/1405.4273, 2014. URL `http://arxiv.org/abs/1405.4273`.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

Meng Cai and Jia Liu. Maxout neurons for deep convolutional and LSTM neural networks in speech recognition. *Speech Communication*, 77:53 – 64, 2016. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2015.12.003. URL `http://www.sciencedirect.com/science/article/pii/S0167639315001375`.

Eugene Charniak. Immediate-head parsing for language models. In *Proceedings*

of 39th Annual Meeting of the Association for Computational Linguistics, pages 124–131, Toulouse, France, July 2001.

Ciprian Chelba and Noam Shazeer. Sparse non-negative matrix language modeling for geo-annotated query session data. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 8–14. IEEE, 2015.

Guoguo Chen, Oguz Yilmaz, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. Using proxies for oov keywords in the keyword search task. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 416–421. IEEE, 2013.

Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics, 1996.

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

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Christopher Cieri, Mike Maxwell, Stephanie Strassel, and Jennifer Tracey. Selection criteria for low resource language programs. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference*

on Language Resources and Evaluation (LREC 2016), Paris, France, may 2016. European Language Resources Association (ELRA). ISBN 978-2-9517408-9-1.

Ronan Collobert. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*, 2011.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390177. URL `http://doi.acm.org/10.1145/1390156.1390177`.

M. Creutz and K. Lagus. Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0. Technical report, Helsinki University of Technology, 2005.

Salil Deena, Madina Hasan, M Doulaty, Oscar Saz, and Thomas Hain. Combining feature and model-based adaptation of rnnlms for multi-genre broadcast speech recognition. *Proc. of the 17th Interspeech, San Francisco, CA*, 2016.

Anoop Deoras, Tomáš Mikolov, and Kenneth Church. A fast re-scoring strategy to capture long-distance dependencies. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1116–1127. Association for Computational Linguistics, 2011a.

Anoop Deoras, Tomáš Mikolov, Stefan Kombrink, Martin Karafiát, and Sanjeev Khudanpur. Variational approximation of long-span language models for lvcsr. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5532–5535. IEEE, 2011b.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, July 2011. ISSN 1532-4435.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. Sentence compression by deletion with lstms. In *EMNLP*, pages 360–368, 2015.

Ankur Gandhe, Florian Metze, and Ian Lane. Neural network language models for low resource languages. In *INTERSPEECH*, pages 2616–2619, 2014.

Joshua T. Goodman. Classes for fast maximum entropy training. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 561–564, Salt Lake City, Utah, USA, May 2001. IEEE.

David Graff and Christopher Cieri. English gigaword ldc2003t05. Web Download. Philadelphia: Linguistic Data Consortium, 2003.

David Graff, Zhibiao Wu, Robert MacIntyre, and Mark Liberman. The 1996 broadcast news speech and language-model corpus. In *Proceedings of the DARPA Workshop on Spoken Language technology*, pages 11–14, 1997.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

Iryna Gurevych. *Natural Language Processing – IJCNLP 2005: Second International Joint Conference, Jeju Island, Korea, October 11-13, 2005. Proceedings*, chapter Using the Structure of a Conceptual Network in Computing Semantic Relatedness, pages 767–778. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-31724-1. doi: 10.1007/11562214_67. URL `http://dx.doi.org/10.1007/11562214_67`.

David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC)*, pages 1222–1225, 2006.

D. Hakkani-Tur and G. Riccardi. A general algorithm for word graph matrix decomposition. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 1, pages I–596–I–599 vol.1, 2003. doi: 10.1109/ICASSP.2003.1198851.

William Hartmann, Viet-Bac Le, Abdel Messaoudi, Lori Lamel, and Jean-Luc Gauvain. Comparing decoding strategies for subword-based keyword spotting in low-resourced languages. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Karl Moritz Hermann and Phil Blunsom. Multilingual models for compositional distributed semantics. *CoRR*, abs/1404.4641, 2014. URL `http://arxiv.org/abs/1404.4641`.

Zhiheng Huang, Geoffrey Zweig, and Benoit Dumoulin. Cache based recurrent neural network language model inference for first pass speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6354–6358. IEEE, 2014.

Infinte monkey theorem. Infinite monkey theorem— Wikipedia, the free encyclopedia, 2002. URL `https://en.wikipedia.org/wiki/Infinite_monkey_theorem`. [Accessed 20-February-2017].

Paul Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, February 1912. URL `http://www.jstor.org/stable/2427226?seq=3`.

Matthew A. Jaro. 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, 1989. ISSN 01621459. URL `http://www.jstor.org/stable/2289924`.

Fred Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In Edzard S. Gelsema and Laveen N. Kanal, editors, *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397. North Holland, Amsterdam, 1980.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. *CoRR*, abs/1508.06615, 2015. URL `http://arxiv.org/abs/1508.06615`.

Dietrich Klakow. Log-linear interpolation of language models. In *Fifth International Conference on Spoken Language Processing (ICSLP)*, pages 1695–1698, 1998.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE, 1995.

Reinhard Kneser, Jochen Peters, and Dietrich Klakow. Language model adaptation using dynamic marginals. In *Eurospeech*, 1997.

Francis Kubala. Design of the 1994 csr benchmark tests. In *Proc. Spoken Language Sys. Technology Workshop*, pages 41–46, 1995.

Roland Kuhn and Renato De Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583, 1990.

Gwénolé Lecorvé and Petr Motlicek. Conversion of recurrent neural network lan-

guage models to weighted finite state transducers for automatic speech recognition. Technical report, Idiap, 2012.

Hung-yi Lee, Yu Zhang, Ekapol Chuangsuwanich, and James R Glass. Graph-based re-ranking using acoustic feature similarity between search results for spoken term detection on low-resource languages. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

George James Lidstone. Note on the general case of the bayes-laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, 8:182–192, 1920.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

B. Logan, J. M. Van Thong, and P. J. Moreno. Approaches to reduce the effects of oov queries on indexed spoken audio. *IEEE Transactions on Multimedia*, 7 (5):899–906, Oct 2005. ISSN 1520-9210. doi: 10.1109/TMM.2005.854429.

Beth Logan, Pedro Moreno, Jean-Manuel Van Thong, et al. An experimental study of an audio indexing system for the web. In *Proceedings of the 4th International Conference of Spoken Language Processing*. Citeseer, 1996.

Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113, 2013.

David J.C. MacKay and Linda C. Bauman Peto. A hierarchical dirichlet language model. *Natural Language Engineering*, 1:1–19, 1994.

Jonathan Mamou, Bhuvana Ramabhadran, and Olivier Siohan. Vocabulary independent spoken term detection. In *Proceedings of the 30th Annual Interna-*

tional ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07, pages 615–622, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-597-7. doi: 10.1145/1277741.1277847. URL `http://doi.acm.org/10.1145/1277741.1277847`.

Michael McCandless, Erik Hatcher, and Otis Gospodnetic. *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Manning Publications Co., Greenwich, CT, USA, 2010. ISBN 1933988177, 9781933988177.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5528–5531, Prague, Czech Republic, May 2011. doi: 10.1109/ICASSP.2011.5947611. URL `http://dx.doi.org/10.1109/ICASSP.2011.5947611`.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 641–648, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273577. URL `http://doi.acm.org/10.1145/1273496.1273577`.

Saeedeh Momtazi and Dietrich Klakow. Trained trigger language model for sentence retrieval in qa: Bridging the vocabulary gap. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*,

CIKM '11, pages 2005–2008, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0717-8. doi: 10.1145/2063576.2063876. URL http://doi.acm.org/10.1145/2063576.2063876.

Saeedeh Momtazi, Friedrich Faubel, and Dietrich Klakow. Within and across sentence boundary language model. In *INTERSPEECH*, pages 1800–1803, 2010a.

Saeedeh Momtazi, Friedrich Faubel, and Dietrich Klakow. Within and across sentence boundary language model. In *11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1800–1803, Makuhari, Japan, September 2010b.

Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1 – 38, 1994. ISSN 0885-2308. doi: 10.1006/csla.1994.1001. URL http://www.sciencedirect.com/science/article/pii/S0885230884710011.

Youssef Oualil, Mittul Singh, Clayton Greenberg, and Dietrich Klakow. Long-short range context neural networks for language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,*, pages 1473–1481. IEEE, 2016.

David S Pallett, Jonathan G Fiscus, William M Fisher, John S Garofolo, Bruce A Lund, and Mark A Przybocki. 1993 benchmark tests for the arpa spoken language program. In *Proceedings of the workshop on Human Language Technology*, pages 49–74. Association for Computational Linguistics, 1994.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

Jochen Peters and Dietrich Klakow. Capturing long range correlations using log-linear language models. In *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 79–94. Springer, 2000.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Nagendra Goel, Mirko Hannemann, Yanmin Qian, Petr Schwarz, and Georg Stemmer. The kaldi speech recognition toolkit. In *In IEEE 2011 workshop*, 2011.

Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

Lawrence Saul and Fernando Pereira. Aggregate and mixed-order markov models for statistical language processing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 81–89. Association for Computational Linguistics, 1997.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. Large, pruned or continuous space language models on a gpu for statistical machine translation. 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 11–19. Association for Computational Linguistics, 2012.

Sadi Evren Seker, Oguz Altun, Ugur Ayan, and Cihan Mert. A novel string distance function based on most frequent K characters. *CoRR*, abs/1401.6596, 2014. URL `http://arxiv.org/abs/1401.6596`.

Noam M Shazeer, Joris Pelemans, and Ciprian Chelba. Sparse non-negative matrix language modeling for skip-grams. 2015.

M. Singh and D. Klakow. Comparing RNNs and log-linear interpolation of improved skip-model on four babel languages: Cantonese, pashto, tagalog, turkish. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8416–8420, May 2013. doi: 10.1109/ICASSP.2013.6639307.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D. Manning, and Andrew Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1201–1211, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=2390948.2391084`.

H. Soltau, F. Metze, C. Fugen, and A. Waibel. A one-pass decoder based on polymorphic linguistic context assignment. In *Automatic Speech Recognition and Understanding, 2001. ASRU '01. IEEE Workshop on*, pages 214–217, 2001. doi: 10.1109/ASRU.2001.1034625.

Radu Soricut and Franz Och. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1627–1637, Denver, Colorado, May–June 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N15-1186`.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. Srilm at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, volume 5, 2011.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *INTERSPEECH*, pages 194–197, 2012.

Martin Sundermeyer, Zoltán Tüske, Ralf Schlüter, and Hermann Ney. Lattice

decoding and rescoring with long-span neural network language models. In *INTERSPEECH*, pages 661–665, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Amos Tversky. Features of similarity. *Psychological review*, 84(4):327, 1977.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. Decoding with large-scale neural language models improves translation. In *EMNLP*, pages 1387–1392. Citeseer, 2013.

William E Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. 1990.

Martin Wöllmer, Björn Schuller, and Gerhard Rigoll. Keyword spotting exploiting long short-term memory. *Speech Communication*, 55(2):252 – 265, 2013. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2012.08.006. URL `http://www.sciencedirect.com/science/article/pii/S0167639312000982`.

Yiming Yang, Hanxiao Liu, Jaime Carbonell, and Wanli Ma. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 159–168, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3317-7. doi: 10.1145/2684822.2685292. URL `http://doi.acm.org/10.1145/2684822.2685292`.

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, April 2004. ISSN 1046-8188. doi: 10.1145/984321.984322. URL `http://doi.acm.org/10.1145/984321.984322`.

Appendices

A Language Model Adaptation

Language model adaptation techniques can largely be classified into interpolation-based techniques and techniques applying constraint specification. In both cases, a combination of language model trained on a large background corpus is combined with a language model trained on the small adaptation corpus. However, arguably the latter set of techniques is considered more powerful. In this section, we discuss one such widely used technique which uses dynamic marginals to perform language model adaptation.

A.1 Fast Marginal Adaptation

Most adaptation techniques utilize unigram language models trained on adaptation text to dynamically adapt predictions as the text is transcribed in Speech Recognition. An important part of adapting predictions using such unigrams ($p_{adap}(w)$, where w is the present word) is to combine these models with language models with background information ($p_{back}(w|h)$, where h is the history for the word w). Language model adaptation with dynamic marginals is fast way of combining such unigram information in a background language models (Kneser et al. (1997)) and arguably more powerful than interpolation-based techniques (Bellegarda (2004)).

Treating the adaptation unigram language model ($p_{adap}(w)$) as a dynamic marginal, Kneser et al. (1997) applies probabilistic constraints and *Minimum Discriminant Estimation* to estimate to model adaptative parameters ($\alpha(w)$) . These parameters are then estimated ($\alpha(w) = \frac{p_{adap}(w)}{p_{back}(w)}$) using a *Generalized Iterative Scaling* scheme, where stopping after the first iteration of the algorithm and substituting adaptation language models with background corpus-based approximations produces the final adapted language model ($p_{adap}(w|h)$). This language model is

defined over T — the set of all n-grams — as follows:

$$p_{adap}(w|h) = \begin{cases} \dfrac{\alpha(w)}{z_0(h)} \cdot p_{back}(w|h) & \text{if } (h,w) \in T \\ \\ \dfrac{1}{z_1(h)} \cdot p_{adap}(w|\hat{h}) & \text{else} \end{cases}$$

with the normalization factors $z_0(h)$ and $z_1(h)$:

$$z_0(h) = \frac{\sum_{w:(h,w)\in T} \alpha(w) \cdot p_{back}(w|h)}{\sum_{w:(h,w)\in T} p_{back}(w|h)}$$

and

$$z_1(h) = \frac{1 - \sum_{w:(h,w)\in T} p_{adap}(w|h)}{1 - \sum_{w:(h,w)\in T} p_{back}(w|h)}$$