Count Information: Retrieving and Estimating Cardinality of Entity Sets from the Web

A dissertation submitted towards the degree
Doctor of Engineering (Dr.-Ing.)
of the Faculty of Mathematics and Computer Science
of Saarland University

by

Shrestha Ghosh

Saarbrücken, 2024

Day of Colloquium:	30.09.2024
Dean of the Faculty:	Univ.-Professor Dr. Roland Speicher
Chair of the Committee:	Prof. Dr. Anna Maria Feit
Reviewers:	Prof. Dr. Simon Razniewski
	Prof. Dr. Gerhard Weikum
	Prof. Dr. Katja Hose
Academic Assistant:	Dr. Soumi Das

Abstract

Extracting information from the Web remains a critical component in knowledge harvesting systems for building curated knowledge structures, such as Knowledge Bases (KBs), and satisfying evolving user needs, which require operations such as aggregation and reasoning. Estimating the cardinality of a set of entities on the Web to fulfill the information need of questions of the form ***"how many ..?"*** is a challenging task. While, intuitively, cardinality can be estimated by explicitly enumerating the constituent entities, this is usually not possible due to the low recall of entities on the Web.

We present our contributions towards retrieving and estimating cardinalities of entity sets on the Web:

- We propose a method, CounQER, for discovering count information in KBs. We identify interpretable classes of features to classify KB predicates that store counts and enumerations. Further, we devise heuristics to align semantically-related counts and enumerations to each other. CounQER is also accessible as a system demonstration.
- We propose a method, CoQEx, to infer count distribution from multiple text snippets. CoQEx is trained using distant supervision to identify relevant counts and predicts the final result via weighted median. CoQEx provides explanatory evidence by forming semantic groups of the contexts, by ranking exemplary instances and by provenance of the counts in the originating snippets. CoQEx is also available online as a system demonstration.
- We tackle the problem of predicting the larger of two sets of entities, when direct comparison of the counts may give incorrect results. We emulate a smart human's approach and introduce a variety of online signals that can be applied to solve the problem. We propose novel techniques for aggregating signals with partial coverage into more reliable estimates on which of the two given classes has more instances.
- We propose, CardiO, a lightweight and modular framework for estimating cardinalities on the Web. CardiO scores counts based on the relevance of their context to the expected answer type, the relevance of the parent sentence and snippet to the user query. CardiO leverages supporting facts to re-score the counts for the final prediction. Further, CardiO identifies relevant peer sets to predict the cardinality of the original entity set.

Kurzfassung

Das Extrahieren von Informationen aus dem Internet ist nach wie vor eine kritische Komponente in Knowledge-Harvesting-Systemen für den Aufbau Knowledge-Bases (KBs), und die Befriedigung sich weiterentwickelnder Nutzeranforderungen, die Operationen wie Aggregation und logisches Schliessen erfordern. Die Schätzung der Kardinalität einer Menge von Entitäten im Web, um den Informationsbedarf von Fragen der Form ***"wie viele ..?"*** zu erfüllen, ist eine anspruchsvolle Aufgabe. Im Prinzip, kann die Kardinalität zwar durch explizite Aufzählung der einzelnen Entitäten geschätzt werden. Aber ist dies aufgrund der geringen Auffindbarkeit von Entitäten im Web normalerweise nicht möglich.

Wir präsentieren unsere Beiträge zum Auffinden und Schätzen von Kardinalitäten von Entitätsmengen im Internet:

- Wir stellen eine Methode vor, CounQER, zum Auffinden von Mengenkardinalitäten in KBs vor. Wir identifizieren eine interpretierbare Klasse von Merkmalen zur Klassifizierung von KB-Prädikaten, die Kardinalitäten und Aufzählungen speichern. Außerdem entwickeln wir Heuristiken, um semantisch verwandte Kardinalitäten und Aufzählungen zueinander in Beziehung zu setzen. CounQER ist auch online als Systemdemonstration zugänglich.

- Wir stellen eine Methode vor, CoQEx, um aus mehreren Textfragmenten Verteilung von Kardinalitäten abzuleiten. CoQEx wird mit Hilfe von Distant Supervision trainiert, um relevante Kardinalitäten zu identifizieren, und sagt das Endergebnis über den gewichteten Median voraus. CoQEx liefert erklärende Evidenz, indem es semantische Gruppen der Kontexte bildet, beispielhafte Instanzen einordnet und die Herkunft der Kardinalitäten in den ursprünglichen Textfragmenten ermittelt. CoQEx ist auch online als Systemdemonstration verfügbar.

- Wir befassen uns mit dem Problem der Vorhersage der größeren von zwei Mengen von Entitäten, wenn der direkte Vergleich der Kardinalitäten falsche Ergebnisse liefern kann. Wir emulieren den Ansatz eines intelligenten Menschen, und stellen eine Vielzahl von Internetbasierten-Signalen vor, die zur Lösung des Problems verwendet werden können. Wir schlagen neuartige Techniken zur Aggregation von imperfekten Signalen zuverlässigeren Schätzungen darüber vor, welche der beiden gegebenen Klassen mehr Instanzen besitzt.

- Wir stellen CardiO vor, ein leichtgewichtiges und modulares Framework zur Schätzung von Kardinalitäten im Web. CardiO bewertet Kardinalitäten basierend auf der Relevanz ihres Kontexts für den erwarteten Antworttyp, der Relevanz des übergeordneten Satzes und den Textfragmenten für die Benutzeranfrage. CardiO nutzt unterstützende Fakten, um die Kar-

dinalitäten für die endgültige Vorhersage neu zu bewerten. Außerdem identifiziert CardiO relevante Peer-Sets, um die Kardinalität der eingegebenen Entitätsmengen vorherzusagen.

Acknowledgements

I would like to thank my advisors — Prof. Dr. Simon Razniewski and Prof. Dr. Gerhard Weikum — without whom this thesis would not be possible. Thank you, Simon, for your unwavering support, patience, and positivity. Thank you, Gerhard, for your abundant energy and curiosity. I would like to thank my collaborators Hiba, Fabian, Damien, and Jeff, my colleagues from D5 and my friends who have provided guidance and support in matters of the head and of the heart. I am indebted to MPII for the freedom to pursue research and the financial support for research visits, summer schools and conferences. I thank my family, especially my parents, for their faith in me. Thank you, Arpan, for always having my back.

Contents

Chapter 1

Introduction

Contents

1.1 Motivation

The internet is a massive storehouse of information which is available in the form of natural language documents and linked data. Information extraction and retrieval pipelines allow us to transform this information into machine-readable data which can be used in many downstream tasks, such as search and question answering (QA). The current information needs of users have evolved from simple factual knowledge to queries beyond lookup, which require additional operations, such as aggregation and reasoning. The application of these operations requires knowledge of different types of information, such as counts, quantities, events, and negations.

Example 1.1: Web Search for Songs by an Artist

A user wants to find the number of songs written by John Lennon for the Beatles. A lookup-based search on the Web might return a snippet from a page listing all songs written by Lennon, another snippet from a page listing a subset of songs Lennon wrote for the Beatles, and yet another snippet which mentions that Lennon wrote "almost 200 songs" under the Lennon-McCartney partnership. It is left to the user to reconcile all the information and form a complete and concise answer to their original question.

This dissertation focuses on count information on the Web. Count information models the cardinality of a set of entities, identified independently as a class, such as *physicists*, or in relation

to another entity, such as *winners of the Nobel Prize in Physics*. This cardinality can be expressed directly as an integer, such as *"there are 244 Nobel Prize winners in Physics"*[1], or as a list of winners, which should be enumerated. As a result, extracting and aggregating count information presents its own unique challenges.

Knowledge Bases (KBs), such as Wikidata [Vrandečić(2012)], store information in the form of ⟨*subject*, *predicate*, *object*⟩ triples[2]. The enumerations for a set of entities are accessible through SPARQL queries of the form ⟨*Marie Curie*, *authorOf*, *?work*⟩, where the variable, *?work*, returns all scientific articles authored by Curie. However, the completeness of such enumerations is usually unknown. Therefore, counting the enumerations for answering count questions is likely to result in inaccurate underestimations. KBs can also store direct assertions, such as ⟨*Marie Curie*, *totalWorks*, *50*⟩, which can be helpful in assessing completeness of enumerations and providing bounds when answering count questions. However, such count triples are not frequent in state-of-the-art KBs.

In Web text, cardinality is expressed in a variety of surface forms, including, but not restricted to numbers, numerals, approximations, and bounds. Additionally, entity sets in Web text, which co-occur with cardinality, are generally partial enumerations of notable entities, except when the set is very small.

Example 1.2: Cardinality Surface Forms.

- Numbers, for example, *there are 244 Nobel Prize winners in Physics*,
- Numerals, for example, *Marie Curie won two Nobel Prizes*,
- Approximations, for example, *there are around one million physicists*,
- Bounds, for example, *there are between 300,000 and 900,000 physicists.*

Example 1.3: Entity Sets Co-occurring with Cardinality.

- Complete enumeration: *Marie Curie won two Nobel Prizes, in Physics in 1903 and in Chemistry in 1911.*
- Partial enumeration of notable entities: *More than 200 spoken languages in Europe, including Russian, English, French, and German.*

Enumerations also appear in different semi-structured formats, such as lists and tables, and in specialized authoritative repositories, such as IMDb: a database of movies, TV shows and more[3], MusicBrainz: an open music encyclopaedia[4], and Ethnologue: a catalog of languages[5].

This dissertation aims to discover entity counts from KB and Web text and estimate cardinalities of classes of entities without having to explicitly enumerate individual entities, and to provide comprehensive answers to count questions of the following nature:

[1] as of 2023.
[2] KBs, for instance Wikidata, also store qualifiers for further contextualization
[3] `https://www.imdb.com/`. Last accessed on March 25, 2024.
[4] `https://musicbrainz.org/`. Last accessed on March 25, 2024.
[5] `https://www.ethnologue.com/`. Last accessed on March 25, 2024.

- specific and crisp, such as
  the number of Nobel Prize Winners in Physics and *the number of mammal species*,
- fuzzy and loosely-defined, such as
  the number of physicists in the world and *the number of languages spoken in Europe*,
- comparative, such as
  are there more physicists than lawyers and *are there more castles than rivers*.

State-of-the-art and its limitations. The significance of count information in KB curation has been established in recent surveys [Weikum et al.(2021), Razniewski et al.(2024)]. Prior work has used cardinality for assessing recall [Mirza et al.(2016)] and for evaluating the quality of learned rules [Tanon et al.(2017)] in KBs. In recall assessment, the general focus is on counting entities, both in KBs [Trushkowsky et al.(2013), Soulet et al.(2018), Luggen et al.(2019)] and in text [Razniewski et al.(2019)], and cardinality is treated as meta-data like a relation constraint [Giacometti et al.(2019), Galárraga et al.(2017)]. KBs are notorious for limited recall and popularity bias [Razniewski et al.(2024)], hence in practice, estimation techniques work well on popular classes, but grossly underestimate the cardinality of classes that form the long tail of the popularity distribution. In the downstream task of KB QA, count questions are identified as a special case of list answers. They are simplified by mapping the list of returned entities to the `COUNT` aggregate [Diefenbach et al.(2018)]. In the natural language domain, counts are obtained as a by-product of numerical information extraction [Roy et al.(2015), Saha et al.(2017)], though there has been work on building relation-specific cardinality extractors [Mirza et al.(2016), Mirza et al.(2017)].

Count questions form 5%-10% of QA benchmarks [Mirza et al.(2018)]. Popular QA benchmarks evaluate counts as span predictions [Rajpurkar et al.(2016), Rajpurkar et al.(2018), Kwiatkowski et al.(2019)], reducing the scope of the task to that of reading comprehension. This leads to two major limitations encountered when answering questions on the Web. *First*, existing benchmarks evaluate document-level boundary predictions. Hence, no connection is drawn between semantically related close count spans, such as *"exactly 73 songs for the Beatles"* and *"around 70 Beatles tracks"*. *Second*, systems cannot handle practical search scenarios, where a user encounters noisy counts, as illustrated in Example 1.1.

1.2 Challenges

We identify the following challenges associated with extracting, estimating and organizing count information from the Web.

1.2.1 Discovering Count Information in KBs

Count information in KBs occurs in complementary forms: as direct cardinality assertions $\langle$*Shakira*, *numberOfChildren*, *2*$\rangle$ and as enumerations $\langle$*Shakira*, *child*, $o_i\rangle$, where o_i is a named-entity, with $i \in \{1, 2\}$, defined for each child. Let us denote the predicates storing cardinality assertions as *counting predicates*, the predicates storing enumerations as *enumerating predicates* and the predicates belonging to either of the two classes as *set-values predicates* or *set predicates* for brevity. This particular instance, where the two predicates, *child* and *numberOfChildren* model the same

relation, is an example of an easily identifiable semantically related count information. However, most general-purpose KBs, like Wikidata [Vrandečić(2012)] and DBpedia [Auer et al.(2007)], do not recognize this semantic relatedness. Moreover, in practice, it is non-trivial to identify KB predicates as cardinality assertions or as enumerations. Cardinality assertions necessarily have integers as objects, but KB predicates can contain integers that represent a variety of other concepts, for instance identifiers or measures, like length and weight, or sequential numbers, like episode number or atomic number. Additionally, KBs do not explicitly specify the entities counted by the counting predicate, instead we see just the aggregate count. Conceptually, enumerating predicates take multiple values, as opposed to functional predicates like the date of birth or birthplace of a person. Yet actual KBs contain a considerable amount of noise, are incomplete, and blur functionality by redundancies, for instance by listing both the birth city and country of a person under *birthplace*. Hence, relying solely on functionality to identify enumerating predicates is imprecise. Similarly, relying only on an exact match between a counting predicate's value and the number of instantiations of an enumerating predicate can give imprecise results. Unlike our example where the cardinality assertion of Shakira's children is equal to the number of child entities, cardinalities do not perfectly match the count of enumerations. Making direct comparisons of *staffSize* or *memberCount* and the number of enumerations under *employerOf* or *hasMember* is unfeasible. Here, we need metrics that can capture correlation despite incompleteness.

1.2.2 Aggregating Count Information from Web Text

Count information on the Web presents itself in two complementary formats: as cardinality and as enumerations. The cardinality format comes from the numerous surface forms, which can be integers, numerals, cardinals, or ordinals [Mirza et al.(2016)]. This makes extraction and consolidation very challenging. The challenge with enumerations is that they are more often notable and incomplete. Thus, solely counting entities underestimates the true count. In addition, snippets in search results regularly contain varying, inconsistent and confusing counts. When searching for the number of songs John Lennon wrote for in his lifetime, we come across a distribution of counts in the search snippets, ranging from *"around 70 songs"* he wrote for the Beatles, to *"roughly 150 tracks"* he composed on his own, to *"almost 200 songs"* written under the Lennon-McCartney partnership. State-of-the-art QA methods simply return a single count without alerting the user to the underlying distribution of counts, be it confirming close counts or contradicting far counts. The emphasis is on predicting the precise span in a document, rather than consolidating evidence across documents. Additionally, predictions returned by retriever-reader systems [Zhu et al.(2021)], do not treat counts and their contexts separately and generate answers as a single string. Hence, close count contexts, such as *"exactly 73 songs"* and *"around 70 singles tracks"* cannot be easily compared.

1.2.3 Cardinality Prediction from Online Sources

Entity sets range from small clearly defined concepts, such as *children of a person*, to loosely-defined concepts, such as *castles*. Counting entities to estimate the cardinality of a class is feasible for sets of the former type. Sets that fall into the latter category cannot be fully enumerated, and estimation techniques are required. One way to simplify the estimation problem, so that precise cardinality prediction is not required, is to predict the bigger class of two given classes. Since Web documents and KBs rarely store direct answers to comparative questions, such as *"Are there more astronauts*

or Physics Nobel Laureates?", questions of this kind are often surprisingly hard to answer. For some comparisons, there are authoritative official sources that provide reliable (albeit not necessarily up-to-date) numbers. For example, the US Bureau of Labor Statistics provides the counts of nearly 830 occupations. In most cases, though, the natural resort is to tap into online sources like KBs, Web snippets from search engine (SE) results, or Large Language Models (LLMs). However, all of these come with biases in what they cover and what not, and often give treacherous signals that lead humans to wrong conclusions. The Web is probably complete on the winners of the Nobel Prize, but, predicting the cardinality of the physicists in the world is very challenging.

1.3 Contributions

Based on the above challenges, this dissertation aims to create a broad understanding of count knowledge in KBs and Web text and propose tractable methods for answering important count problems. In particular, this dissertation addresses the following research questions.

RQ 1. *How can we distinguish count information stored in KBs? Can we align semantically-related cardinality assertions and enumerations?*

In Chapter 3, we discover count predicates in KBs, such as *child* and *numberOfChildren*, capturing the count and the enumerated list. We propose the CounQER method to identify and connect semantically related count predicates in KBs, such as Wikidata or DBpedia. We identify interpretable features for classifying predicates that store counts or enumerations. We devise statistical and linguistics inferences as heuristics to align counts and enumerations that correspond to each other. Through case studies, we highlight how alignments can be used to identify anomalies and inconsistencies in the KB. This work was published in the *Journal of Web Semantics 2020* and a demonstration of the system was published in *The European Semantic Web Conference (ESWC) 2020*.

RQ 2. *How can we answer questions on entity counts from Web snippets? Do aggregation strategies assist in better prediction of entity counts?*

In Chapter 4, we show that counts in Web texts have a noisy distribution and propose CoQEx, a system that infers the count distribution from multiple text snippets and consolidates them into a comprehensive answer. CoQEx is trained with distant supervision to predict a count phrase from a web snippet given a question on entity counts. As a result, a user is presented with a count answer, the distribution used to predict the answer, and the explanatory context of the count. User studies corroborate the hypothesis that adding context and the provenance of the answer increases the user's confidence in the predictions of a system. This work was published as a short paper in the *ACM SIGIR Conference 2022* and as an extended version in the *Journal of Web Semantics 2023*. A demonstration of the CoQEx system, which answers real-time count questions from users, was published in the *ACM International Conference on Web Search and Data Mining, WSDM 2023*.

RQ 3. *Is it possible to predict the larger one of two sets of entities, such as scientists versus lawyers, from Web contexts? It is easier than estimating the cardinalities?*

In Chapter 5, we introduce the generalized problem of cardinality estimation and develop a method to predict the larger of the two classes when the precision of the class cardinality is unknown.

We investigate the counts obtained from different information sources, like KBs, Web snippets, and LLMs and different signal types, like the cardinality of the class itself, and cardinalities of certain subgroups. We devise heuristics to divide classes into subgroups that are applicable to all classes, for instance, physicists and lawyers by countries, or by years. We develop ensemble methods that effectively combine multiple cardinality signals thus obtained to predict the larger of two given classes. Furthermore, we find that information sources exhibit biases in covering different domains, such as classes of geographical entities, occupations, and man-made objects. For instance, KBs are more accurate in comparing two geographical classes than Web snippets or LLMs. Similarly, Web snippets are the most accurate source of information when comparing two occupations. This work was published in the *ACM Web Conference 2023*.

RQ 4. *How can we estimate the cardinality of a set of entities? Can LLMs help lightweight models in generating traceable predictions?*

In Chapter 6, we propose the CardiO method that answers count questions of different categories. We investigate the performance of LLMs and smaller unsupervised models on cardinality estimation. We evaluate the effect of using a single information source, only LLM or only Web snippets, versus using both. We evaluate the effect of different ground truth and question characteristics on the performance metrics. We find that our lightweight and modular prediction model beats small LLM models. Although larger models have higher precision, they lack traceability. Additionally, we find that when used to enhance CardiO components, larger models do not contribute to the final precision or recall. This work was published in the *ACM Web Conference 2024*.

1.4 Publications

The results of this dissertation have been published in the following publications. All publications, data, code, and demo systems are accessible from the main webpage of this project: `https://www.mpi-inf.mpg.de/count-knowledge`.

- **Uncovering Hidden Semantics of Set Information in Knowledge Bases**
  Shrestha Ghosh, Simon Razniewski, Gerhard Weikum
  Journal of Web Semantics Vol. 64, 2020.

- **CounQER: A System for Discovering and Linking Count Information in Knowledge Bases**
  Shrestha Ghosh, Simon Razniewski, Gerhard Weikum
  System demonstration at *European Semantic Web Conference (ESWC) 2020.*

- **Answering Count Queries with Explanatory Evidence**
  Shrestha Ghosh, Simon Razniewski, Gerhard Weikum
  ACM SIGIR Conference on Research and Development in Information Retrieval, 2022.

- **Answering Count Questions with Structured Answers from Text**
  Shrestha Ghosh, Simon Razniewski, Gerhard Weikum
  Journal of Web Semantics 76, 2023.

- **CoQEx: Entity Counts Explained**
  Shrestha Ghosh, Simon Razniewski, Gerhard Weikum
  System Demonstration at *ACM International Conference on Web Search and Data Mining (WSDM) 2023.*

- **Class Cardinality Comparison as a Fermi Problem**
  Shrestha Ghosh, Simon Razniewski, Gerhard Weikum
  Companion Proceedings of the ACM Web Conference 2023.

- **CardiO: Predicting Cardinality from Online Sources**
  Shrestha Ghosh, Simon Razniewski, Damien Graux, Gerhard Weikum
  Companion Proceedings of the ACM Web Conference 2024.

These are related works by the author that are not covered in this dissertation.

- **Completeness, Recall, and Negation in Open-World Knowledge Bases: A Survey**
  Simon Razniewski, Hiba Arnaout, Shrestha Ghosh, Fabian M Suchanek
  ACM Computing Surveys Vol. 56, Issue 6, 2024.

- **Limits of Zero-shot Probing on Object Prediction**
  Shrestha Ghosh
  Knowledge Base Construction from Pre-trained Language Models workshop at International Semantic Web Conference (ISWC) 2023.

- **Beyond Aggregations: Understanding Count Information for Question Answering.**
  Shrestha Ghosh
  Doctoral Consortium at International Semantic Web Conference (ISWC) 2020

1.5 Organization

The rest of this dissertation is organized as follows. In Chapter 2, we provide background on the definition, associated challenges, and scope of count information. Further, we discuss existing methods of extracting and organizing count knowledge. We describe our method for identifying semantically related count predicates in KBs in Chapter 3. Chapter 4 addresses the challenges of extracting and aggregating entity counts from Web snippets. The following two chapters extend the scope from entity counts to also include class cardinalities. Chapter 5 tackles the problem of class cardinality comparison to determine the larger one of two given classes. In Chapter 6 we describe CardiO, our framework for estimating cardinality from Web snippets using lightweight methods and LLMs. Finally, we conclude with a summary and an outlook in Chapter 7.

Chapter 2

Background

Contents

This background chapter introduces the concepts associated with count information and positions the research problems addressed in this thesis in context with existing research. In particular, in Section 2.1, we discuss important count information concepts. In Section 2.2 we discuss how count information has been modeled. Next, Section 2.3 dives into the knowledge on the Web and how it is extracted. We discuss applications of count information in KB curation and QA in Section 2.4. Finally, we focus on bias in the online world in Section 2.5.

2.1 Count Information Concepts

Count information captures the cardinality of a set of entities, directly as ***a count***, or as ***an enumeration*** of individual entities. We keep a flexible notion of entity sets to allow count information to model ***crisp*** concepts, such as the set of Nobel Prize winners in Physics, as well as ***fuzzy*** concepts, such as the set of physicists in the world.

We illustrate two sets of entities in Example 2.1. Notice how the cardinality itself could be ***exact*** or ***an estimate***, depending on the properties of the set. In the example, we highlight two properties: crispness of definition and size. Fuzzy concepts are loosely defined, resulting in variations on how different sources report on the said set. Our example set of physicists in the world could include any or all of the scientists doing active research in the field, the professors teaching in a university, the people with a doctoral degree in Physics who have now turned YouTubers, and so on. Note that crisp definition does not imply small entity sets. The set of scientific articles and the set of patents

are crisply defined large sets. Similarly, fuzzy definition does not imply large sets — the set of plant species used in bonsai is probably a few hundreds[1], though any plant could be used for bonsai.

Example 2.1: Illustration of Count Information Concepts.

Concept	Example 1	Example 2
Set of Entities	Nobel Prize Winners in Physics	Physicists in the world
Properties	*crisp, small*	*fuzzy, large*
Count/Cardinality	224[a]	roughly 1 million [Day(2015)]
Property	*exact*	*estimate*
Enumeration	{Max Planck, Marie Curie, . . . , }	{Jocelyn Burnell, Peter Higgs, ⋯ , }
Property	*complete (224 entities)*	*incomplete (10,000 entities)*

[a] As of 2023 from `https://www.nobelprize.org/prizes/lists/all-nobel-prizes-in-physics`

2.2 Modeling Count Information

Knowledge representation is a well-known field in AI that deals with formal representation of world knowledge to enable formal reasoning [Brachman and Levesque(2004)]. Several works discuss the broad spectrum of knowledge representations [Staab and Studer(2013), Hogan et al.(2021)] and their inability to deal with web-scale data [Suchanek(2020)]. We cover knowledge extraction from web-scale data in the next section.

Count information in formal representations has been modeled as qualifying number restrictions in description logics [Hollunder and Baader(1991), Calvanese et al.(1998)] and as cardinality assertions in the OWL standard [McGuinness et al.(2004)]. In Example 2.2, we illustrate counts modeled in formal representations and in large web-scale KBs. While such formal representations provide the vocabulary to implement cardinality in KBs, such constraints are very sparsely applied. Cardinality constraints are more effective for functional predicates, such as the birthdate or the birthplace of a person, where the cardinality remains constant across subjects.

Take for example the *child* relation, a property instantiated for almost 15% of the human entities in Wikidata[2]. A simple query reveals that less than 8% of these human entities have a corresponding *numberOfChildren* predicate[3]. In the 7653 cases where both predicates are present, we can compare the number of *child* enumerations and the count value of *numberOfChildren* to predict completeness. Only 30% parent-child relations are complete, i.e., number of enumerations match the count value. More than 68% parent-child relations are incomplete, i.e., the number of enumerations are less than the count value and a little over 1% of the parent-child relations are inconsistent, i.e., the number of enumerations are more than the count value. The situation grows worse for larger sets, like members of a political party or songs by a band, where we still expect complete enumerations. Even larger sets like lakes, mountains, or castles are not expected to be fully enumerated and this corresponds to their representation in the real world where they are incomplete.

[1] From Wikipedia category page on Plants used in bonsai and a page on List of species used in bonsai. Last accessed April 2024.

[2] There are 6.3 million human entities in Wikidata as per `https://www.wikidata.org/wiki/Wikidata:Statistics`.

[3] `https://w.wiki/9fxP` As of March 2024.

Example 2.2: Count Information Modeling.

Restriction on the class humans to have one child in **Description Logics**

`human ⊓ (=1 child)`

Assertion of the number of child predicates for Marie Curie using **OWL Standard**

`ClassAssertion(ObjectExactCardinality(2 :child) :MarieCurie)`

Representing counts via a predicate in a **Web-scale Knowledge Base**

⟨*Marie Curie*, *numberOfChildren*, *2*⟩

Aligning Counts and Enumerations. Closest to our work is schema alignment, a classic problem in data integration [Rahm and Bernstein(2001)]. For ontologies and on the semantic web, added complexity comes from taxonomies and ontological constraints [Euzenat and Shvaiko(2007), Shvaiko and Euzenat(2013)]. The approaches to ontology alignment include BLOOMS [Jain et al.(2010)] and PARIS [Suchanek et al.(2011)], voting-based aggregation [Wang et al.(2013)], probabilistic frameworks [Niepert et al.(2010)], or methods for the alignment of multicultural data [Boldyrev et al.(2018)]. These methods typically rely on a combination of lexical, structural, constraint and instance based information. Our setting, where enumerations need to be aligned with counts, is atypical in ontology alignment and has not received prior attention. Despite several methods for automatically learning logical axioms and patterns [Lehmann and Hitzler(2010), Galárraga et al.(2013)], we are not aware of attempts to identify and align counts with corresponding enumerations.

2.3 Knowledge on the Web

In addition to the mainstream encyclopedic knowledge, which is harvested by general-purpose KBs and is focused on notable entities, there exists knowledge that focuses on specific information aspects, like quantities, negation and commonsense. We discuss some notable entity-centric KBs followed by different aspect-based knowledge, and parametric latent knowledge in pre-trained language models. Finally we discuss search results returned by the state-of-the-art commercial search engines.

Entity-centric Knowledge. Earliest pioneering works in creating universal KBs include the *Cyc* [Lenat(1995)] and the WordNet [Miller(1995)] projects. However, these were created from handcrafted rules and were thus limited in scope and scale. Progress in automatic information extraction methods [Sarawagi(2008)], led to the web-scale extraction of facts, also known as knowledge harvesting [Weikum and Theobald(2010), Suchanek and Weikum(2013)]. The largest general-purpose KBs, publicly accessible today, include YAGO [Suchanek et al.(2007)], DBpedia [Auer et al.(2007)], BabelNet [Navigli and Ponzetto(2010)], and Wikidata [Vrandečić(2012)]. Notable KBs used in the industry include the Google Knowledge Graph [Singhal(2012)], Microsoft Satori [Qian(2013)], Amazon Product Graph [Dong(2019)] and Baidu Knowledge Graph [Baidu(2020)]. Such comprehensive structured data is invaluable for information retrieval and NLP tasks like search and question answering [Weikum et al.(2021)].

Numeric Knowledge. Popular KBs contain considerable numeric information in the form of objects which are literals and can be parsed into a numeric datatype. Research has focused on extracting numeric data from web tables [Neumaier et al.(2016)] and detecting outliers in existing numeric data in KBs to improve parsers used in automatic KB curation [Wienand and Paulheim(2014)]. There has been work on extracting physical quantities, a subset of numeric information that represent measurement and their units, from Web documents and web tables for populating KBs [Sarawagi and Chakrabarti(2014), Subercaze(2017), Ho et al.(2021), Ho et al.(2022)]. Textual information extraction of numeric information targets the extraction of any quantity in text [Saha et al.(2017)]. Count information extraction from Wikipedia sentences like *"The LoTR series consists of three books"* for a given subject and predicate is used KB curation tasks such as increasing recall for missing objects [Mirza et al.(2017)]. Additionally, in natural language processing (NLP) tasks, counts are identified as a separate class of values [Weischedel et al.(2012)] by popular NLP pipelines, like SpaCy.

Negative Knowledge. Unlike factual statements that store information that hold, for example, *"Marie Curie won two Nobel Prizes"* and *"The European Union (EU) has 27 countries"*, negative statements store information that do not hold, such as *"Stephen Hawking* ***did not win*** *a Nobel Prize in Physics"* and *"Switzerland* ***is not part of*** *the EU"* [Arnaout(2023)]. KBs like Wikidata express empty objects using a special no-value statement [Darari et al.(2015)], but it is not widely used. Count information expressed specifically through zero-valued statements form a small subset of negations, which is more focused on discovering salient properties that do not hold [Arnaout et al.(2021)]

Commonsense Knowledge. Commonsense statements store information about general everyday concepts, such as *airplanes* and *food*, to be used in downstream reasoning tasks. Notable projects on commonsense knowledge are, ConceptNet [Liu and Singh(2004)], Webchild [Tandon et al.(2014)], Atomic [Sap et al.(2019)], Quasimodo [Romero et al.(2019)] and Ascent [Nguyen et al.(2021)]. Quantity distributions of common objects, such as mass of animals, can be mined from the Web to make informed relative comparisons, such as a lion is heavier than a jaguar [Elazar et al.(2019)]. There has been some work in evaluating numerical commonsense knowledge of LLMs, which shows that it is difficult to beat human performance [Lin et al.(2020)]. While commonsense knowledge does encode some count information, it mostly deals with non-entity objects, such as the number of wheels of a bicycle or the number of legs of an animal.

Temporal Knowledge. KB facts are often bound to a specific time-period or a point in time, for instance, people holding leadership positions and population of a country. This enables us to not only answer questions such as the total number of Indian prime ministers, but also the number of Indian prime ministers since the Emergency (which ended in 1977). While YAGO2 [Hoffart et al.(2013)] and temporal qualifiers in Wikidata [Patel-Schneider(2018)] enable curating temporal context, such knowledge is quite sparse. Extracting temporal knowledge [Ling and Weld(2010)] and curating existing KBs with temporal knowledge is under active research [Jain et al.(2020), Dhingra et al.(2022), Ji et al.(2021)]. Temporal knowledge is also researched in the context of event detection and extraction. We point the reader to a recent work that surveys temporal knowledge in the context of event detection and extraction [Guan et al.(2022)].

Count Knowledge. In the natural language domain, counts are obtained as a by-product of numerical information extraction [Roy et al.(2015), Saha et al.(2017)], which works well for quantities and numeral representations. Unlike physical quantities, which have explicit measurement units, counts do not have units. We can refer to the type of entities being counted as the unit of the count, though this has not been considered in previous works. There has been work on building relation-specific cardinality extractors [Mirza et al.(2016), Mirza et al.(2017)] and estimators for enriching KBs [Mirza et al.(2018)]. Previous work has also used mining techniques to determine cardinalities of predicates in KBs, though such techniques work well for predicates that take relatively constant number of objects across subjects [Giacometti et al.(2019)].

Parametric Latent Knowledge. Large pre-trained models like BERT [Devlin et al.(2019)], T5 [Raffel et al.(2020)], and GPT [Radford et al.(2018)] have fuelled a paradigm shift in NLP, where parameters learned by these models during self-supervised pre-training are used in several down-stream NLP tasks [Min et al.(2023)]. These models, also known as Large Language Models (LLMs) from the language modeling objective of predicting a word sequence for a given context, have been shown to be effective in recalling factual information [Petroni et al.(2019), Singhania et al.(2023)] and for dense representation [Reimers and Gurevych(2019), Karpukhin et al.(2020)]. LLMs rely heavily on high quality prompts [Jiang et al.(2020)] and post-hoc prompt-based learning [Brown et al.(2020)] leaving the main model a black-box function of its input [Hewitt et al.(2023)]. Overall, LLMs are known to struggle with tasks involving numeric and count information [Lin et al.(2020)]. Subsequent works have explored solving reasoning tasks by chain-of-thought prompting, a method of prompting LLMs to generate step-by-step logic [Kojima et al.(2022)], or to generate verifiable logic, such as a Python program or an algebraic equation in the case of solving math word problems [Imani et al.(2023)]. However, multiple research indicate that rather than building actual reasoning, LLMs learn frequent patterns to generate predictions and are easily misled by negations or distracting contexts [Helwe et al.(2021), Dziri et al.(2024)].

Search Results. Search engines (SEs) provide access to information on the open web. Current SEs by Google and Bing return relevant snippets from the source documents, highlight text spans in snippet, return direct answers and knowledge panels to enhance user experience, who would otherwise have to search through multiple links. These richly annotated results can be tapped using search APIs [Google(2010), Bing(2022)].

As shown in Figure 2.1, SEs can answer simple count queries from their underlying KBs, if present, a trait which we exploit to create our CoQuAD dataset (Chapter 4). But more often they return informative albeit inconsistent text snippets, similar to QA-over-text systems. This is due to two reasons: the *first* is due to counts which are easier to obtain from snippets than from KBs, where they are either incomplete and/or require an expert user to create the structured SPARQL queries and even then it might be difficult to recall all relevant entities. For instance, in Figure 2.2 (left), the 906 Nobel Prize winners does not include the prize winners in Economics because, at the point of querying, this award is not considered a subclass of the Nobel Prizes. While this is technically correct, the Nobel Prize website releases counts of the winners from all six categories (Figure 2.3 (left)). For the Lennon query, the structured KB result returns 168 songs (Figure 2.2 (right)), all attributed jointly to Lennon and his band member McCartney. However, SE snippets suggest that this number is close to 200 (Figure 2.3 (right)).

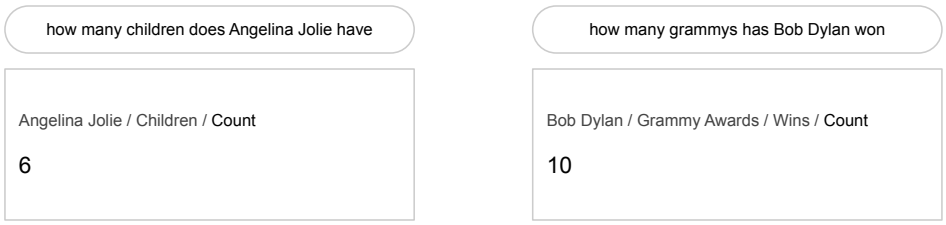


Figure 2.1: Result tapped easily from search engine's back-end KB.

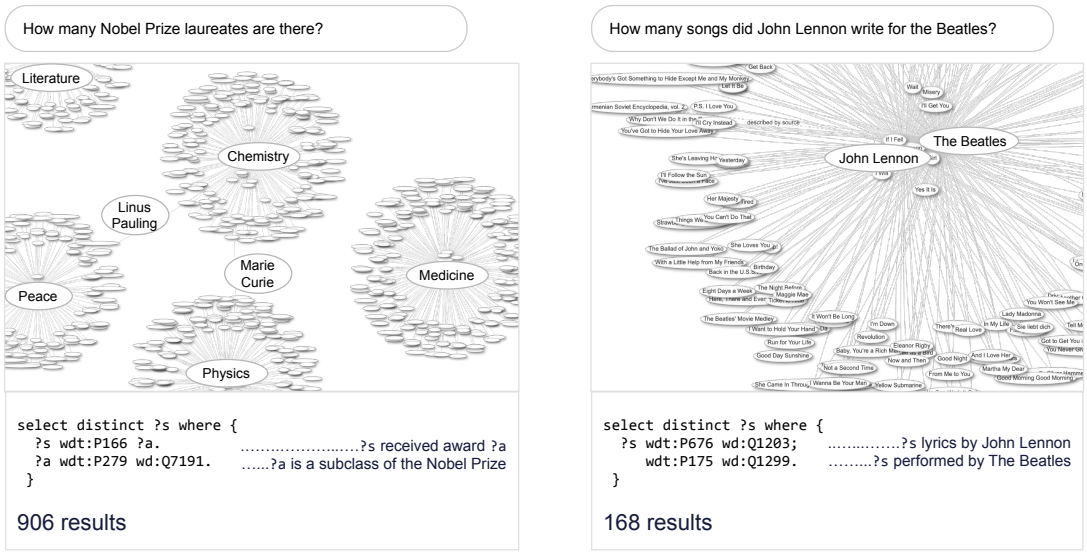


Figure 2.2: Wikidata results for structured queries as of April 2024.

The *second* reason SEs return inconsistent text snippets is due to variability in the count itself. For the number of Nobel Prize winners, we see some variance in the counts depending on how different sources report it — all *975 laureates* vs. *965 individuals* (Figure 2.3 (left)). The basic Lennon query has a highest-ranked Google snippet with *more than 150* when given the telegraphic input *number of songs by John Lennon* and *almost 200* when given the full-fledged question *how many songs did John Lennon write*. Refining this query by the qualifier *for the Beatles* makes this puzzling situation even more complex with counts referring to his joint contribution, such as 200 Beatles songs, and counts referring to his individual contribution, such as *61 songs* and *73 songs* (Figure 2.3 (right)). Because of the lack of consolidation, the user should now decide whether there are multiple correct answers across text segments, and how to combine them into a final count.

2.4 Applications of Count Information

2.4.1 KB Curation

The lifecycle of a KB extends beyond extraction to curation, which includes assessment, maintenance, and integration of new facts. Count information, especially cardinalities, has been used for KB recall assessment — how many entities missing in a KB — and for enriching KBs with missing enumerations. Recall is an important dimension of KB quality, with impact on downstream use cases

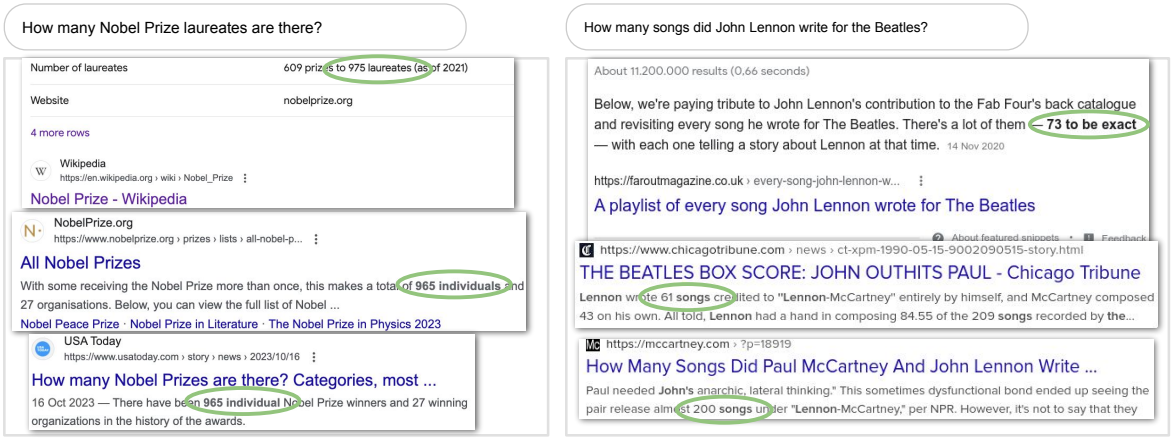


Figure 2.3: Result snippets from Web search with low variance (left) and high variance counts.

[Paulheim(2017)]. Unlike precision, it cannot be easily evaluated by sampling. Existing approaches to KB recall estimation can be grouped into three families. The first are approaches based on statistical patterns in the data, e.g., sample overlap [Luggen et al.(2019), Trushkowsky et al.(2013)], digit-distributions [Soulet et al.(2018)], or association rules [Galárraga et al.(2017)]. The second are relative approaches, i.e., where recall expectations are collected from related entities [Hopkinson et al.(2018), Soulet et al.(2018)]. The third are text-extraction based approaches [Mirza et al.(2017), Razniewski et al.(2019)].

2.4.2 Question Answering

The task of QA, in particular over the Web, employs techniques from information retrieval and NLP to answer questions posed in natural language from structured KBs, semi-structured tables and unstructured text collections. We direct the reader to [Roy and Anand(2022)] which comprehensively covers techniques to answer questions over the Web. These techniques depend on the source of information and the complexity of questions, i.e., whether they are factoid questions or conversational or multi-hop. It is reported that between 5% and 10% of questions in popular TREC QA datasets concern counts [Mirza et al.(2018)]. This information need, although acknowledged by QA systems, is so far not dealt with in a principled manner. We will discuss existing QA techniques over KBs and QA systems in open domain that typically employ a retriever-reader paradigm over text or KBs or both. Further, we will discuss benchmarks and evaluation techniques used by QA systems.

Question Answering over KBs. In KB QA, count questions are usually identified as a special case of list answers, and simplified by mapping the list of returned entities to the `COUNT` aggregate. This holds true for semantic parsers [Berant et al.(2013)], template-based systems like AQQU [Bast and Haussmann(2015)], as well as subgraph-based detection based systems like QAnswer [Diefenbach et al.(2018)], which map natural language queries to their SPARQL equivalent. Another prominent SPARQL-based system is QUINT [Abujabal et al.(2017)], though it does not handle aggregations. KBs are known to be incomplete and QA systems gravitate towards the Web for answering complex questions [Talmor and Berant(2018)]. This is especially true for count information. For example,

Wikidata contains 227 songs attributed to John Lennon[4], but is incomplete in indicating whether these are written for the Beatles, whether it was written under the Lennon-McCartney partnership or otherwise, without more sophisticated queries. Consequently, attempts have also been made to improve recall by hybrid QA over text and KB, yet without specific consideration of counts [Xu et al.(2016), Lu et al.(2019), Christmann et al.(2023)].

Open-domain Question Answering. Former state-of-the-art systems typically approached text QA via the reading comprehension paradigm [Chen et al.(2017), Dua et al.(2019), Sanh et al.(2019), Karpukhin et al.(2020), Joshi et al.(2020)], where the systems find the best answer in a given text segment. Current state-of-the-art uses the retriever-reader approach in open-domain QA. Extractive systems return either a single best text span from several supporting text segments [Chen et al.(2017), Wang et al.(2018)] or a ranked list of supporting texts with the best text span per document [Karpukhin et al.(2020)]. The DPR system by [Karpukhin et al.(2020)][5] returns *approximately 180* from its rank-1 text segment to both the query on the number of songs written by John Lennon and the refined variant with *. . . for the Beatles*. The other top-10 snippets include false results such as *five* and contradictory information such as *180 jointly credited* (as if Lennon had not written any songs alone). Generative systems encode multiple supporting texts to generate a single answer [Izacard and Grave(2021)]. Furthermore, this fusion-in-decoder method has been extended to infuse knowledge from KBs [Yu et al.(2022)]. Another line, [Krishna et al.(2021)], concerns long form question answering, where the QA model retrieves multiple relevant documents to generate a whole answer paragraph. The advent of transformer-based LLMs has revived the retriever-reader paradigm, with LLMs generating answers over retrieved evidences [Lewis et al.(2020)]. Nevertheless, LLMs are prone to be over-confident [Ji et al.(2023)] and mitigating this is an area of active research [Jiang et al.(2021), Asai et al.(2023)].

Benchmarks and Evaluation Techniques. QA systems are tested on reading comprehension datatsets, the most popular being SQuAD [Rajpurkar et al.(2016)], CoQA [Reddy et al.(2019)] and more recent being DROP [Dua et al.(2019)]. On open domain QA, datatsets such as Natural Questions [Kwiatkowski et al.(2019)], TriviaQA [Joshi et al.(2017)]. Notable KBs datasets are LC-QuAD 2.0 [Dubey et al.(2019)], WebQuestions [Berant et al.(2013)] and QALD [Usbeck et al.(2018)]. New benchmarks, such as GrailQA [Gu et al.(2021)], evaluate generalization ability of powerful KB-QA systems that use transformer models. Reading comprehension and open domain QA datasets are annotated with answer spans, while the datasets for KB-QA come with a single or a list of answers, and an optional equivalent SPARQL query. More recent KB-QA datasets, such as VQuAnDa [Kacupaj et al.(2020)], ParaQA [Kacupaj et al.(2021)] provide answer verbalizations, such that each KB answer comes with one or several paraphrased responses in natural language. Multi-document-multi-hop reasoning datasets, in turn, focus on chaining evidence [Dua et al.(2019), Bauer et al.(2018)]. The ELI5 [Fan et al.(2019)] and ASQA [Stelmakh et al.(2022)] datasets contains diverse open-ended queries with supporting information from relevant web sources. While the setting is related, long form QA is concerned with generating textual answers, and we focus on answering count questions in a more structured format.

Evaluation metrics for reading-comprehension style benchmarks typically employ strict matching

[4] `https://w.wiki/4XVq`, as of March 2024
[5] `http://qa.cs.washington.edu:2020`, last accessed July 2022.

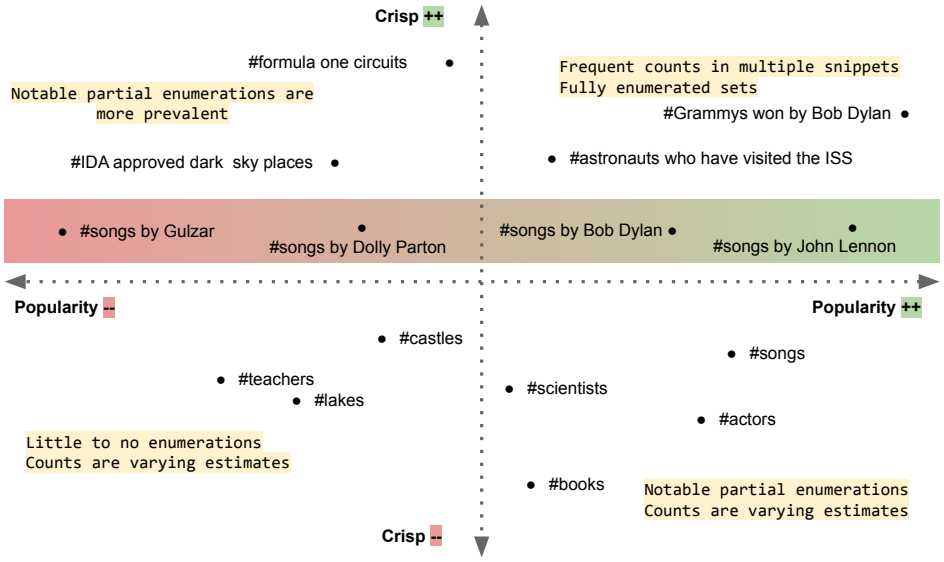


Figure 2.4: Illustrating availability of counts and enumerations of sets of entities based on their relative crisp definition and popularity.

requirements, like measuring accuracy, F1-score and exact match [Zeng et al.(2020)]. There exist popular n-gram matches as well, such as BLEU, ROUGE, METEOR and their variations, but these metrics are not well-adapted for automatic evaluation of freeform QA [Chen et al.(2019), Rogers et al.(2023)]. On a question level, these metrics measure the token-level overlap. This does not transfer well to count queries, especially counts which do not have one authoritative answer. We propose relaxed metrics for evaluation in Chapters 4 and 6.

2.5 Bias in the Online World

A related aspect when dealing with count information is data bias in online information. We look at the two main sources of Web content — the relatively small, but clean and comprehensive Wikipedia available since 2007 [Wikimedia(2007)], and massive crawls of the Web released regularly by the Common Crawl since 2008 [Crawl(2008)]. Wikipedia, for instance, is a major source of general knowledge used in construction of large KBs [Suchanek et al.(2007), Auer et al.(2007), Vrandečić(2012)] and popular QA benchmarks [Rajpurkar et al.(2016), Kwiatkowski et al.(2019)]. C4 [Raffel et al.(2020)] and the Pile [Gao et al.(2020)] are publicly available huge datasets used for training some well-known LLMs, such as T5 [Gao et al.(2020)], GPT3 [Brown et al.(2020)], LLaMA [Touvron et al.(2023)]. While C4 exclusively comes from pre-processing and filtering the Common Crawl data, the Pile dataset additionally scrapes data from 22 sources, including collections of academic, medical and legal sources.

Bias on the Web can have adverse effects on users unless taken into account in designing systems that use Web data [Baeza-Yates(2018)]. Wikipedia suffers from specific demographic biases such as gender bias [Sun and Peng(2021)]. As of 2023, two of the world's three most spoken languages

(Chinese and Hindi)[6] do not feature even in the top-10 languages in Wikipedia with the most articles[7]. Representation in KBs is unbalanced, where some entities have highly detailed facts than others, such as Hollywood actors versus actors from any other movie industry, and some classes are more complete than others, such as the class of Nobel Prize winners compared to the classes of professors at different universities. This lack of balance is due to multiple reasons, including, but not limited to, reporting bias [Gordon and Van Durme(2013)], data, schema, and inferential bias [Janowicz et al.(2018), Safavi et al.(2021)]. Both C4 and Pile datsets contain large number of duplicates, synthetic data as well as personal information [Dodge et al.(2021), Elazar et al.(2023)].

In our case, we focus on the inconsistencies between distribution in the real-world vs. the Web that arise due to unbalanced frequency distributions [Wei et al.(2021), Razeghi et al.(2022), Zevallos et al.(2023)]. Consider the availability of counts and enumerations of the entity sets illustrated in Figure 2.4. Take the example of actors and teachers. Even though the cardinality of both sets have the same order of magnitude in real life, actors are much better covered and more frequent in the online world than any other profession of the same size. One way to tackle big sets is to break them down into smaller subgroups, the data for which might be readily available. For instance, we could extrapolate the number of teachers in the world from official statistics on selected countries.

Even when we reduce the scale to smaller entity sets, certain demographic biases still prevail. We highlight a few song-writers who have written a comparable number of songs in Figure 2.4, and try to estimate the number of songs written by each of John Lennon, Bob Dylan, Dolly Parton and Gulzar. There exist multiple counts in search results referring to the number of songs written by Lennon (referring to 200 songs he co-wrote for the Beatles and 150 songs as a solo artist) and for Dylan (counts range between 600 and 1000 songs). Counts returned for songs written by Parton mix counts of songs written by her (roughly 700) with her compositions and recordings, which are in thousands. Contrary to these, almost no counts exist for the equally prolific Indian song-writer Gulzar. When we query Wikidata[8], we find 217 songs written by Lennon, 150 by Dylan, 112 for Gulzar and only 33 by Parton — a very skewed distribution. It is also odd that a more famous artist than Gulzar[9], Dolly Parton has only 33 enumerations of songs written by her. We refer to the works section of each artist on MusicBrainz, a music database website, for an authoritative ground-truth [MusicBrainz(2003)]. We estimate that Lennon wrote 400 songs, Gulzar wrote 500, Parton wrote 700 songs, and Dylan wrote 1000 songs.

[6] https://www.ethnologue.com/insights/ethnologue200/
[7] https://meta.wikimedia.org/wiki/List_of_Wikipedias
[8] `https://w.wiki/9iCe` Last accessed April 2024.
[9] When comparing the number of facts in Wikidata: Dolly Parton has 600 facts, while Gulzar has 120.

Chapter 3

Count Information in Knowledge Bases

Contents

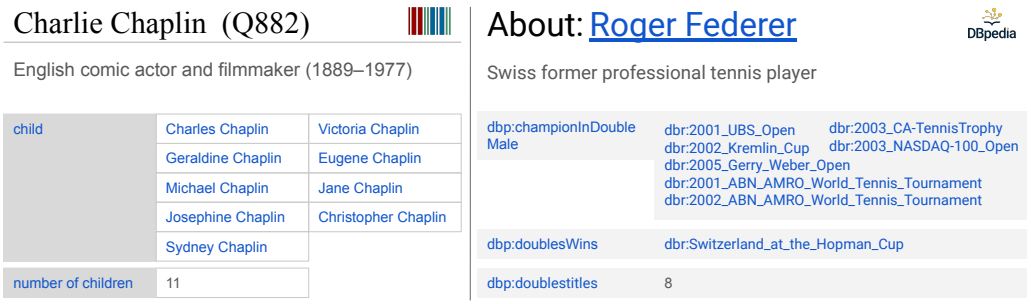


Figure 3.1: Illustrating set predicates in Wikidata and DBpedia (as of April 2024).

3.1 Introduction

This chapter focuses on count information in general-purpose Knowledge Bases (KBs). The goal is to identify *set-valued predicates*, i.e., predicates that represent the relationship between an entity and a set of entities by storing aggregated integers, such as `numberOfChildren` and `staffSize`, or by storing individual members, such as `parentOf` and `worksFor`. Both formats are typically complementary: unlike enumerating predicates, counting predicates do not give away individuals, but are more likely informative towards the true set size, thus this coexistence could enable interesting applications in question answering and KB curation.

Knowledge bases (KBs) like Wikidata [Vrandečić(2012)], DBpedia [Auer et al.(2007)], Freebase [Bollacker et al.(2008)] and YAGO [Suchanek et al.(2007)] are important backbones for intelligent applications such as structured search, QA and dialogue. Properly modelling and understanding the schema of such KBs, and the semantics of their predicates, is a crucial prerequisite for utilizing them. *Set-valued predicates*, *i.e.*, predicates which connect entities with sets of entities typically come in two variants: (i) as *enumerating predicates*, which list individual objects for a given subject, and (ii) as counting predicates, which present total object counts. Figure 3.1 illustrates set predicates in Wikidata and DBpedia. The predicate `child`, which lists individual children of Chaplin, is an enumerating predicate, while `numberOfChildren`, which gives a count of Chaplin's children, is a counting predicate, and both model the same phenomenon. Another example is about the number of doubles titles won by Roger Federer, where the enumerating predicates `doublesWins` and `championInDoubleMale` enumerate his wins, and the counting predicate `doublestitles` gives the count. Set predicates can also model merely related phenomena, for instance, for a given location, the sets described via `numberOfInhabitants` and `birthPlace` typically have a considerable overlap, but do not coincide.

Identifying set predicates and set alignments would be an important step towards a better understanding of KB semantics. In particular, set alignments would be beneficial for the following use cases:

1. *KB curation:* Identifying gaps and inconsistencies in the KB and getting directives for acquiring missing pieces of knowledge (e.g., adding the 3 absent children of US president Garfield to the KB) [Mirza et al.(2018)].

2. *Query formulation:* Aiding users to formulate comprehensive SPARQL queries by showing them related predicates (e.g., finding people with more than 2 children by computing the

union of matches for the counting predicate and results from aggregating the instances of the enumerating predicate[1]) [Calvanese et al.(2017)].

3. *Answer explanation:* Exemplifying query results by showing key instances of queries over counting predicates (e.g., showing a few individual Turing Award winners for a query about the number of award winners).

Note that we do not advocate that all gaps between related predicates hint at errors or incompleteness that require actions. Scope of a KB, (non-)notability of entities, or privacy considerations may well motivate that certain gaps should not be filled, and temporal semantics may add subtleties (same predicate stores the number of employees time-stamped by year). Properly identifying set predicates and set alignments in knowledge bases is also difficult for other reasons:

- Enumerating predicates are often incomplete (like Chaplin's children and Federer'd doubles titles in Figure 3.1) and counting predicates may be approximate estimates only (like number of inhabitants); so cardinalities do not match count values, yet the predicates should be linked in order to couple them for future KB completion, consistency assessment and other use cases listed above.

- KBs contain thousands of predicates, often with uninformative names and without coherent type signature, thus making the identification of set predicates and their alignments challenging.

Example 3.1: Challenges in Set Predicate Discovery

Predicate	Inferred Function	Challenge
`dbp:lost`	Stores the number of matches lost by a sports player	**Uninformative and generic** *In the case of Roger Federer it stores four records of matches lost, but no futher information about specific tournaments.* *Additionally, it records the number of satellites lost from a satellite group (Hot Bird).*
`dbp:frenchOpenResult`	Stores the result of a player at the French Open tennis tournament	**Incoherent type signature** *String values SF (semi-final) for Pete Sampras and W (winner) for Roger Federer.* *Integer values 1 through 4 for rounds leading up to the quarter-finals (QF).*

Approach and Contribution. We present CounQER (for "**Coun**ting **Q**uantifiers and **E**ntity-valued P**R**edicates"), the first comprehensive methodology towards identifying and linking set predicates in KBs. CounQER is judiciously designed to identify set predicates in noisy and incomplete web-scale KBs such as Wikidata, DBpedia and Freebase. It operates in two stages: In the first stage, supervised classification combining linguistic and statistical features is used to identify enumerating and counting predicates. In the second stage, a set of statistical co-occurrence and correlation measures is used in order to link the set predicates.

Our salient original contributions are:

[1]Example query for people with > 2 children: `https://w.wiki/9rtC`

1. We introduce the notion of set predicates, its variants, and highlight the benefits that can be derived from identifying their alignments.

2. We present a two-stage methodology for (i) predicting the counting and enumerating predicates via supervised classification and, (ii) ranking set predicates of one variant aligned to the other variant via statistical and lexical metrics.

3. We demonstrate the practical viability of our approach by extensive experiments on four KBs: Wikidata, Freebase, and two variants of DBpedia.

4. We publish results of our alignment methodology for these KBs, which contains 264 alignments from DBpedia mapping-based KB, 3703 alignments from the DBpedia raw KB, 25 alignments from the Wikidata-truthy KB, 274 alignments from the Freebase KB. These can be downloaded or interactively assessed through our online demonstration at `https://counqer.mpi-inf.mpg.de/spo`.

3.2 Related Work

While there is a rich body of research on ontology alignment and schema matching [Rahm and Bernstein(2001), Euzenat and Shvaiko(2007), Shvaiko and Euzenat(2013), Jain et al.(2010), Suchanek et al.(2011), Wang et al.(2013), Niepert et al.(2010), Boldyrev et al.(2018)], these works typically focus on identifying perfectly matching pairs of predicates with the same or largely overlapping values. This situation differs from our setting, where the integer values of counting predicates and the cardinalities of enumerating predicates modelling the same or related phenomenon rarely match perfectly.

Despite several methods for automatically learning logical axioms and patterns [Lehmann and Hitzler(2010), Galárraga et al.(2013)], we are not aware of attempts to learn set relatedness. Though formal ontologies provide the framework for modelling count information through role restrictions [Hollunder and Baader(1991), Calvanese et al.(1998)] and cardinality assertions [McGuinness et al.(2004)], such bounds are sparsely populated in KBs and cardinality assertions are not explicitly identified.

Count information is a subset of numeric information, i.e., information that can be parsed into a numeric datatype. In the context of KBs research has focused on extracting numeric information, particularly quantities, from Web tables [Sarawagi and Chakrabarti(2014), Neumaier et al.(2016), Ho et al.(2021)] and documents [Saha et al.(2017)], and extracting temporal information [Ling and Weld(2010), Hoffart et al.(2013)]. There has been some work on count information extraction from sentences like *"The LoTR series consists of three books"* [Mirza et al.(2017), Mirza et al.(2018)]. In the present work, we investigate the identification and alignment of set predicates in knowledge bases itself, i.e., without external text sources. Such information is important for assessing and improving KB completeness [Razniewski et al.(2016), Razniewski et al.(2019)]. Other approaches that utilize count information to predict KB recall include inferring from statistical patterns in the data, e.g., sample overlap [Luggen et al.(2019)], digit-distributions [Soulet et al.(2018)], or association rules [Galárraga et al.(2017)].

Set information is important for QA, where it is reported that between 5% and 10% of questions in popular TREC QA datasets concern counts [Mirza et al.(2018)]. SPARQL-based QA systems

acknowledge this need, but mostly rely on counting the set of entities returned. AQQU [Bast and Haussmann(2015)], for instance, aggregates over the set of entities for questions starting with *"How many?"*. QAnswer [Siciliani et al.(2022)] employs a more principled approach where a rule-based trigger identifies different modifiers (count, filter, order) to be applied on the returned set.

3.3 Design Space

3.3.1 Problem Statement

Let P be a set of predicates. A *knowledge base* (KB) is a set of triples (s, p, o), where $p \in P$, s is an entity, and o is either an entity or a literal. For the remainder of this chapter we assume that each triple (s, p, o) with an entity as object also exists in its inverse form (o, p^{-1}, s) in each KB, thus the following elaborations need to consider only one direction.

The foundational concept for this work is that of a set predicate.

Definition 1: Set Predicate

A set predicate is a predicate which *conceptually* models the relation between an entity and a set of entities.

Note. We emphasize that *set predicate* refers to the intended semantics of the modeller, not to be mixed with the capabilities of the modelling language. In particular, unlike SQL, the RDF data model does not know a *SET* datatype, but can capture sets via multiple triples sharing subject or object.

Set predicates can be expressed in KBs in two variants: Via binary predicates that enumerate individual set members, and via counting predicates that abstract away from individuals, and store aggregate counts only.

Definition 2: Enumerating Predicate

An Enumerating Predicate is a set predicate that models sets via binary membership statements.

Definition 3: Counting Predicate

A Counting Predicate is a set predicate whose values represent counts of entities.

Following these definitions, the predicates `child`, `numberOfChildren`, `doublesWins`, `championInDoubleMale` and `doublestitles` in Figure 3.1 are set predicates. Here, `numberOfChildren` and `doublestitles` are counting predicates and the rest are enumerating predicates. Other examples are `worksAt`$^{-1}$ and `authorOf`, which frequently enough take several values for a subject. This is in contrast to predicates of a functional or quasi-functional nature, such as `bornIn` and `mother`, which predominantly take a single object, and hence, where counts are uncommon and rarely informative. Yet the threshold for enumerating predicates is imprecise, for instance, the predicate `citizenOf` is predominantly functional, but some entities still have multiple citizenship which are conceivably countable.

Other examples of counting predicates are `population`, `numberOfStudents` and `airlineDestinations`. Entity counts necessarily are integers, yet KB predicates can contain integers that represent a variety of other concepts, for instance identifiers or measures like length and weight. The distinction between counting predicates and measurement predicates like `riverLength` and `revenue` is quite crisp, since measurements usually come with units (km, €, etc.) and can take fractional values (1.7 km) while entity counts cannot. Our definition is phrased to also exclude some predicates taking integer values, like `episodeNumber` (not a count, but a sequential number assigned to an episode of a TV series) and `floorCount` (a count, but not of something commonly considered as entities). Thus, integer values are a necessary but not a sufficient condition for being a count predicate.

Note that the above definitions are conceptual only. Functionalities computed over actual KBs are unreliable due to incompleteness, errors, and redundancies, and common KBs do not have an *entity-count* datatype. Thus, in later the sections, we will develop supervised classifiers for identifying both kinds of set predicates.

We summarize our first problem as follows.

Problem 3.1: Set Predicate Identification

Given a KB with a predicate set P, identify the set of enumerating predicates, E, and the set of counting predicates, C.

Let us now turn to the relation between set predicates. *Set-relatedness* refers to the (ideal) amount of overlap that two set predicates have on a per-subject basis. For instance, in a perfect KB, `child` and `numberOfChildren` describe exactly the same set of objects per subject, once via a listing of names, ones via the aggregate count.

In turn, the predicates `population` and $\texttt{bornIn}^{-1}$ do not describe the same sets, but would typically exhibit a significant overlap (many people live in the same place they are born in, though neither entails the other). On the other hand, `population` and $\texttt{headquarterLocation}^{-1}$ are not set related. Although population sizes and the number of company headquarters in a place are correlated numbers, the described entities do not overlap at all, instead, are even of distinct types (`person` and `company`).

Conceptually, set relatedness between two predicates can therefore be computed as an across-subjects aggregated set overlap measure, with perfect matches being the strongest relatedness. Note that this definition is strictly conceptual, since in actual KBs, counting predicates do not give away which actual entities they count.

Problem 3.2: Set Predicate Alignment

Given sets of enumerating predicates E and counting predicates C, for each set predicate $p \in E \cup C$, rank the predicates from the other set by their set-relatedness.

Note that the above definitions of set-relatedness are conceptual definitions. In practice, KBs do not give access to the entities counted by counting predicates, instead one only sees aggregate counts. To quantify and qualify set-relatedness, in the following sections, we will thus build a set of unsupervised alignment heuristics.

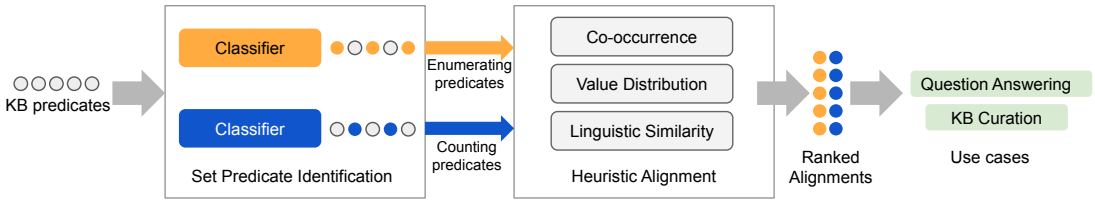


Figure 3.2: Illustration of the CounQER methodology.

3.3.2 Architecture

Our goal is to develop a robust set predicate identification and linking methodology, that, with limited supervision, can work across different KBs.

If knowledge bases were clean, set predicate identification could solely rely on relation functionality and datatypes. As this is not the case in practice [Wu et al.(2014), Wienand and Paulheim(2014), Zaveri et al.(2016)], we instead propose to approach set predicate identification via a supervised classification framework that combines a diverse set of textual, datatype, and statistical features. Predicate alignment can also in principle be approached via hand-crafted rules, heuristic alignment metrics, or supervised learning. Due to the particularities of individual predicates and KBs (most set predicates have only very few perfect alignments), to avoid overfitting, we opt here for a set of heuristic alignment metrics. We design the heuristics in order to capture various desiderata of meaningful alignments, and combine them in an ensemble metric.

Following the above considerations, we split our CounQER (short for "**Coun**ting **Q**uantifiers and **E**ntity-valued P**R**edicates") methodology in two steps: (i) supervised predicate classification, and, (ii) heuristic predicate alignment (see Figure 3.2).

3.3.3 KB Assumptions

Our approach is designed to work on a variety of knowledge bases, without requiring strong assumptions on their internal structure. Fulfillment of the following features is desirable, though not essential: (i) High-level categories/classes for entities, in particular *Person, Place, Organisation, Event* and *Work*. Where these are not available, we utilize links to Wikidata to extract them. (ii) High-level datatypes, in particular *float*, *int* and *date*. Where these are not available, we utilize standard parsers and simple heuristics, such as that numbers between 1900 and 2020 are likely dates. (iii) human-readable labels for properties, with spaces or in camel case notation. Where these are not available, we deactivate corresponding linguistic features.

3.4 The CounQER Methodology

The CounQER methodology is illustrated in Figure 3.2. In the first phase, *supervised predicate classification*, we use two classifiers to predict the two set variants, namely, enumerating and counting predicates. We rely on a family of features, most importantly, (i) set-related textual features, extracted from a background corpus, (ii) type information about the domain and range of the predicates, and (iii) statistical features about the number of objects per subject at different percentiles.

In the second phase, *heuristic predicate alignment*, we identify related counting and enumerating predicates using (i) set predicate co-occurrence information, (ii) set predicate value distribution, and (iii) linguistic relatedness. By assigning each pair of enumerating and counting a relatedness score, we can rank related predicates accordingly. While we evaluate the heuristics on labeled data, they are highly complementary, and thus, the choice of the heuristic to be used can be adapted to particular use cases.

3.4.1 Set Predicate Identification

3.4.1.1 Enumerating Predicates

As stated in Section. 3.3, if KBs were clean, functionality (#triples per subject) would be the criterion for identifying enumerating predicates.

Yet actual KBs contain a considerable amount of noise, are incomplete, and blur functionality by redundancies (e.g., listing both the birth city and country of a person under `birthPlace`). In CounQER, we thus rely on supervised classification, where functionality is only one among several features towards enumerating predicate identification.

Textual Features. Where KBs use human-readable predicate names, a basic sanity check for enumerating predicates is to verify whether in human language, the predicate name is used both in singular and plural.

1. *Plural-singular ratio:* For each predicate, we apply a heuristic to generate its plural/singular form. First we identify the last noun in the predicate label using the Python *nltk* package, and then we use the Python `inflect` library to identify its form (singular/plural) and convert it to the other (plural/singular). We then compute the text frequency ratio based on the Bing API, obtaining, for instance, for $\frac{|\texttt{children}|}{|\texttt{child}|}$ a ratio of $\frac{128.000.000}{87.000.000} = 1.47$, while for `birthplace`, the ratio is $\frac{|\texttt{birthplaces}|}{|\texttt{birthplace}|} = \frac{1.550.000}{21.000.000}$, a ratio of 0.08.

Type Information. Subject and object types may explain the applicability of other features, and certain types of objects may more naturally be counted than others, and certain subjects may more frequently come with set predicates than others. To avoid overfitting and ensure compatibility across KBs, in this work we only consider five frequent and general classes, {*Person*, *Place*, *Organisation*, *Event* and *Work*}, which we use to capture the domain and range type of a predicate. Further details follow in Sec. 3.5.2.

2. *Predicate domain:* We encode the most frequent class of a predicate domain via binary features per class, including a 6th class for *other*.

3. *Predicate range:* We encode the class of predicate range in binary variables (same as in predicate domain).

KB Statistics. KB statistics instantiate the observed functionality. As functionality may be blurred by outliers or a long tail of single-valued subjects, we input various datapoints in order to increase resilience of the measure. We also include basic information on datatypes.

4. *Mean, maximum, minimum, 10th and 90th percentile number of objects per subject (functionality):* These features describe the number of objects a predicate takes per subject, with mean and percentiles giving resilience against rare outliers. For example, `occupation` in Wikidata-truthy KB has a mean of 1.3, maximum of 30, minimum of 1, 10th percentile of 1 and 90th percentile of 2. The predicate `placeOfBirth` in Wikidata-truthy KB has a maximum of 6 objects per subjects and 1 object per subject for the other features — minimum, mean, 10th and 90th percentile.

5. *Datatype distribution:* The fraction of triples of a predicate whose objects are of datatype entity. For instance, both the predicates `occupation` and `placeOfBirth` take entities for 99% of the triples in Wikidata-truthy KB.

3.4.1.2 Counting Predicates

As per our conceptual definition, counting predicates are distinguished by having *entity-count* as their datatype. As none of the KBs investigated in this work records such a datatype, we have to use various heuristics towards identifying counting predicates. An important necessary condition are integer values, yet these alone are not sufficient. We utilize the following classes of features.

Textual Features.

1. *Plural-singular ratio:* This feature captures the plural/singular ratio of a predicate, obtained exactly as for enumerating predicates.

Type Information.

2. *Predicate domain:* We identify the domain of the predicates by tracing the class of the predicates to one of the most general classes in the type hierarchy, {*Place*, *Person*, *Organization*, *Event*, *Work*}. Each of the domain class is encoded as a binary variable in the classifier.

KB Statistics.

3. *Datatype distribution:* We calculate the fraction of triples of a predicate taking integer values over the total number of triples of that predicate. For instance, the predicate `numberOfEpisodes` in the DBpedia mapping-based KB takes only integer values, whereas `episodeNumber` in the DBpedia raw KB takes integer values for 96% of the triples.

4. *Mean, maximum, minimum, 10th and 90th percentile of count value:* These features describe the actual integer value of the predicate, e.g., the mean for `numberOfEpisodes` (DBpedia mapping-based KB) is 106, the maximum is 90015, the minimum is 0, the 10th percentile is 6 and the 90th percentile is 156.

5. *Mean, maximum, minimum, 10th and 90th percentile of the number of objects per subject (functionality):* These features describe the number of integer valued triples per subject.

   For example, the mean `numberOfEpisodes` (DBpedia-mapping-based KB) a subject takes is 1, the maximum is 8, the minimum, the 10th percentile and the 90th percentile all are 1, *i.e.,* most subjects have only one fact containing this predicate. In contrast, an ordinal integer predicate

like `episodeNumber` (DBpedia raw KB) has the following statistics - mean 32, maximum 975, minimum 1, 10th percentile 6 and 90th percentile 66. This odd behavior is exhibited because the article page lists all or a subset of the episode numbers in a series [2].

3.4.2 Heuristic Predicate Alignment

The output of the previous stage are the enumerating predicates E and the counting predicates C. The task of this stage is to find for each predicate in $E \cup C$ the most set-related predicates from the other set. As this task may to some extent be KB-specific, we approach it via a set of unsupervised ranking metrics. We introduce three families of metrics for predicate pairs. Here, we illustrate each heuristic using the enumerating predicate $\texttt{gold}^{-1}$, which links an entity (sportsperson) to the winning sports events, and the counting predicate `singlestitles`, which gives the count of singles titles of a tennis player.

Set Predicate Co-occurrence. Our first family of heuristics ranks predicates by their co-occurrence. Co-occurrence is an indication towards topical relatedness, and we propose various measures that capture absolute and relative co-occurrence frequencies.

1. *Absolute*(e, c): The number of subjects which have triples with both e and c set predicates. For instance,
$$\textit{Absolute}(\texttt{singlestitles},\texttt{gold}^{-1}) = 64. \tag{3.1}$$

2. *Jaccard*(e, c): The ratio of the absolute number of subjects for which e and c co-occur, *i.e.,* *Absolute*(e, c) divided by the union of subjects which take either e or c or both. For instance,
$$\textit{Jaccard}(\texttt{singlestitles},\texttt{gold}^{-1}) = 0.006. \tag{3.2}$$

3. *Conditional*(e, c): Co-occurrence can also be expressed as a conditional probability, i.e., the ratio of the absolute value, *Absolute*(e, c), to the number of subjects which take either c or e. For our given example,
$$\textit{Conditional}_E(\texttt{singlestitles},\texttt{gold}^{-1}) = 0.011. \tag{3.3}$$
with respect to subjects only taking the predicate $\texttt{gold}^{-1}$ and,
$$\textit{Conditional}_C(\texttt{singlestitles},\texttt{gold}^{-1}) = 0.015. \tag{3.4}$$
with respect to subjects only taking the predicate `singlestitles`. This implies that if a given subject has the predicate `singlestitles`, it is more likely that the subject also has the predicate $\texttt{gold}^{-1}$ than the other way around.

4. *PairwiseMutualInformation*(e, c) or (*P'wiseMI*): The log of the ratio of the joint distribution of e and c to their individual distributions.
$$PMI(e, c) = \log_2 \frac{|\{s \mid s \in \langle s, e, \cdot\rangle; s \in \langle s, c, \cdot\rangle\}| \times |\{s \mid s \in \langle s, \cdot, \cdot\rangle\}|}{|\{s \mid s \in \langle s, e, \cdot\rangle\}| \times |\{s \mid s \in \langle s, c, \cdot\rangle\}|} \tag{3.5}$$

[2] DBpedia subjects with count of `episodenumber` facts `https://tinyurl.com/dbpedia-raw-episodenumber`

$$PMI(\texttt{singlestitles}, \texttt{gold}^{-1}) = -5.2. \tag{3.6}$$

which implies that the two predicates are less likely to co-occur than expected from their individual occurrences. In general, this metric ranges between $-\infty$ and $\min(-\log p(e), -\log p(c))$. The lower bound is reached when the pair (e, c) does not co-occur for any subject, and the upper bound is reached when either e always co-occurs with c or vice versa.

Set Predicate Value Distribution. Co-occurrence is important but can nonetheless be spurious, e.g., when many sports teams have both the predicates `stadiumSize` and `coachOf`. A possibly even stronger indicator for set relatedness is a match or correlation in values, i.e., if across subjects, the number of values for the enumerating predicate, and the count stored in the counting predicate, coincide, or correlate. We propose three variants: (5) To count the number of exact matches, and in (6) and (7) two relaxed metrics that look for correlation and percentile similarity.

5. *PerfectMatchRatio*(e, c): The ratio of subjects where the number of objects in e exactly matches the value in c to the number of subjects, which takes both e and c predicates. For example,

$$P'fectMR(\texttt{singlestitles}, \texttt{gold}^{-1}) = 0.125. \tag{3.7}$$

6. *Correlation*(e, c): The Pearson correlation between the size of objects of e and the value of c for all subjects in which they co-occur. For the above predicate pair,

$$Correlation(\texttt{singlestitles}, \texttt{gold}^{-1}) = 0.724. \tag{3.8}$$

7. *PercentileValueMatch*(e, c): A softer score than a perfect match ratio (B 5), for matching the 90th percentile value of the number of objects that e takes per subject with the 90th percentile value of the c, such that the closer the value is to 1 the better the alignment. Let O_c and O_e denote the distribution of the values and the #objects per subject, respectively.

$$P'tileVM(e, c) = Min\left(\frac{p'tile(O_e, 90)}{p'tile(O_c, 90)}, \frac{p'tile(O_c, 90)}{p'tile(O_e, 90)}\right) \tag{3.9}$$

$$P'tileVM(\texttt{singlestitles}, \texttt{gold}^{-1}) = 0.333. \tag{3.10}$$

Linguistic Similarity. Besides co-occurence, correlations can also be spurious. For instance, `population` and `headquarterLocation`$^{-1}$ are well correlated (bigger cities host more companies), but nonetheless, they refer to completely different kinds of entities (persons vs. companies). Our third family of heuristics thus looks at topical relatedness.

8. *CosineSimilarity*(e, c) measures the cosine of the angles between the average of the sets of word vectors of the labels of e and c obtained from pre-trained Glove embeddings [Pennington et al.(2014)] using the Python *Gensim* library. Wikidata predicate labels are already individual

words, for DBpedia and Freebase we split the predicates at capitalization and punctuation, respectively. For example, `headquarterLocation`$^{-1}$ becomes {`headquarter, location`} and `race_count` becomes {`race`, `count`}.

$$CosineSim(\texttt{titles,gold}^{-1}) = 0.318 \tag{3.11}$$

Out of vocabulary words, such as `singlestitles`, lead to an empty word list. Similarity with an empty word list is assigned a score of zero as follows.

$$CosineSim(\texttt{singlestitles,gold}^{-1}) = 0. \tag{3.12}$$

Alignment Summary. In the following experiments we evaluate the alignment heuristics individually against ground truth annotations on the NDCG [Järvelin and Kekäläinen(2002)] score. In this way, we can discover the best performing heuristic, and with enough training data, could even perform ensemble learning. Yet as reliance on a single heuristic would be brittle, and ensemble learning requires larger evaluation data, as a robust best-effort, we propose here to retain from each of the three families of heuristics the best performing one, and merge their scores by averaging them.

3.5 Experiments

3.5.1 KBs Used

We use four popular general purpose KBs: DBpedia raw extraction [Auer et al.(2007)], DBpedia mapping-based extraction[3] [Lehmann et al.(2015)], Wikidata truthy[4] [Vrandečić(2012)] and Freebase[5] [Bollacker et al.(2008)]. We analyze each KB in terms of predicate coverage.

1. **DBpedia raw** *(52.6M triples)*: All predicate-value pairs present in the infoboxes of English Wikipedia article pages.
2. **DBpedia mapping-based** *(29M triples)*: A cleaner infobox dataset where predicates were manually mapped to a human-generated ontology. Unmapped predicates and type violating triples are discarded.
3. **Wikidata truthy** *(210.3M triples)*: Simple triple export of Wikidata that ignores some advanced features such as qualifiers and deprecated ranks.
4. **Freebase** *(1B triples)*: The tuple store available as an RDF dump at `https://developers.google.com/freebase`.

We also analyzed YAGO [Suchanek et al.(2007)] *(1.1B triples)*, a WordNet-aligned and sanitized harvest of Wikipedia infobox statements, containing only 76 distinct predicates. By manual inspection we found several enumerating predicates, such as `hasChild` and `isCitizenOf`, but only one counting predicate, `numberOfPeople` and therefore refrained from further processing of this KB.

On adding inverse triples, i.e., adding (o, p^{-1}, s) for every (s, p, o) where o is an entity, the size of DBpedia-raw increased by 7.6M, DBpedia-map by 18M, Wikidata by 101.1M and Freebase by 442.1M.

[3] We used DBpedia version 2016-10 for both extraction.
[4] We used the version as of the Oct 2018.
[5] We used the version as of 2019 July.

Table 3.1: Total number of KB predicates (direct + inverse) and most frequent ones.

KB	All	Frequent
DBP-raw	73,234	16,635
DBP-map	2,008	1,670
WD-truthy	6,111	4,067
Freebase	799,807	13,872
YAGO	*(79)*	*(79)*

To reduce noisy data, we use predicates which appear in at least 50 triples. In Table 3.1 we show the number of predicates that remain after filtering all infrequent predicates. It is evident that the cleaner KBs like Wikidata and DBpedia mapping-based KB have better predicate representation. Freebase and DBpedia raw KBs are noisier, with a very long tail of less frequently occurring predicates.

3.5.2 Preprocessing

Predicate Statistics Computation. Given a KB of SPO triples, we generate the descriptive statistics of the KB predicates including (i) the datatype distribution - fraction of the triples of a predicate which take *integer*, *float*, *date*, *entity* and comma-separated *string* values, (ii) the mean, maximum, minimum, 10th and 90th percentile of the integer values that a predicate takes, (iii) the mean, maximum, minimum, 10th and 90th percentile of the number of entities per subject of a predicate and, (iv) the mean, maximum, minimum, 10th and 90th percentile of the number of integer values per subject of a predicate. We identify comma-separated string values as a datatype in order to handle noisy representations, especially in DBPedia-raw, where object sets are often captured in a single string with comma separation (e.g., "children: Mary, John, Susan").

Type Information. We then proceed to find the predicate domain and range. To maintain uniformity across KBs we trace the type to one of the more general classes in the type hierarchy, {*Place, Person, Organization, Event, Work*}, with the default fallback class for entities being *Thing* and non-entities being *Literal*. The fallback classes capture long-tailed classes and string objects, which have no class information.

We sampled 100 subjects and 100 objects for each predicate and selected the majority class in each set as the domain and range of the predicate. Across all four KBs, the (micro-average) coverage of the predicate domain by the classes are {*Place:* 18%, *Person:* 23%, *Organization:* 14%, *Event:* 4.75%, *Work:* 17.25%, *Thing:* 23%} and, for predicate range, {*Place:* 18.25%, *Person:* 20%, *Organization:* 15.25%, *Event:* 1.25%, *Work:* 17%, *Thing:* 1.75%, *Literal:* 26.5%}.

Linguistic Features. The frequency of occurrence of a predicate on the web in singular and plural form is determined from the total estimated web search matches returned by the Bing custom search API[6]. For inverse predicates, we reuse the predicate labels of their forward form for getting the textual features for the classifiers and the linguistic similarity measure for the alignment heuristics.

[6]https://azure.microsoft.com/en-us/services/cognitive-services/bing-custom-search/

3.5.3 Training and Evaluation Data

We prepared the data for the *classification step* by employing crowd workers to annotate 400 randomly selected predicates for enumerating predicates and 400 for counting predicates from the four KBs — taking 100 from each KB. The annotation task comprised a predicate and five sample subject-object pairs, with options to select if the predicate was likely a set predicate (enumerating or counting). The format used for annotating counting and enumerating predicates is shown in Examples 3.2 and 3.3, respectively.

Example 3.2: Illustrating Counting Predicate Annotation Question

Q: *Based on the following facts, decide whether the relation gives a count of unique entities.*

The Herald (Sharon)	circulation	15715
H.O.W. Journal	circulation	4000
L'Officiel	circulation	101719
The Music Scene (magazine)	circulation	25000
Pipe Dream (newspaper)	circulation	7000

Options: ∘*Yes* ∘*Maybe yes* ∘*Maybe no* ∘*No* ∘*Do not know*

Example 3.3: Illustrating Enumerating Predicate Annotation Question

Q: *Based on the following facts, decide whether the relation enumerates entities.*

A Low Down Dirty Shame	producer	Mike Chapman
Bye Bye Brazil	producer	Luiz Carlos Barreto
Heaven Knows, Mr. Allison	producer	Eugene Frenke
Surviving Paradise	producer	Kamshad Kooshan
I'll Come Running Back to You	producer	Bumps Blackwell

Options: ∘*Yes* ∘*Maybe yes* ∘*Maybe no* ∘*No* ∘*Do not know*

We collected three judgements per predicate, *i.e.,* a total of 2400 annotations (2 variants of set predicates × 4 KBs × 100 predicates × 3 judgments). The options in the annotation task are graded. We translated the labels into numeric scores {*Yes*: 1, *Maybe yes*: 0.75, *Do not know*: 0.5, *Maybe no*: 0.25, *No*: 0}, with the final label being the average of all judgments. Concerning annotator agreement, we found the pooled standard deviation of the scores per predicate to be 0.41. We only keep rows with a clear polarity, *i.e.,* rows with average score outside the interval $(0.4, 0.6)$, effectively excluding rows averaging around the option *Do not know*. The labels of the remaining rows are then translated into binary 0-1-judgments. Of the counting predicate rows, 86.25% showed a clear polarity, of the enumerating predicate rows, 82%. Thus, we obtained our training data, with 39 positive and 306 negative data points for the counting classifier and, 133 positive and 195 negative data points for the enumerating classifier.

We can conclude from Table 3.2 that in general, KBs contain comparably few counting predicates, which also contributes to the low precision score of the counting classifier. From the numbers in Table 3.2, we observe that enumerating predicates have a comparably higher occurrence.

For the *alignment step*, evaluation data was prepared by collecting relevance judgements from crowd workers. We randomly chose 300 enumerating and 300 counting predicates as returned by

Table 3.2: Distribution of classifier training samples across KBs.

KB	Counting Predicates		Enumerating Predicates	
	Positive	Negative	Positive	Negative
DBP-raw	16	62	33	53
DBP-map	9	72	27	58
WD-truthy	7	87	27	55
Freebase	7	85	46	29
Total	39	306	133	195

our classifiers. As co-occurring predicate pairs have a long tail of infrequent pairs that due to small sample size might lead to spurious heuristics scores, we set a threshold on the absolute co-occurrences for the alignments. Of all co-occurring pairs we consider those which co-occur for at least 50 subjects for annotation, evaluation and final ranking purposes. We then created the set of top-3 counting predicates returned by all the alignment heuristics for each enumerating predicate, so that for each enumerating predicate we had up to 27 counting predicates as candidates. Note that the alignment is KB-specific, so we return top-3 predicates from the same KB to which the enumerating predicate belongs.

We repeated the step with the counting predicates, this time returning the top-3 enumerating predicates for each counting predicate. On an average, there were 5 candidates for each set predicate in the enumerating and counting case. The annotation task asked each worker to judge the topical relatedness of a pair of set predicates (an enumerating and a counting predicate) and the degree of completeness based on the integer value of the counting predicate and the entities covered by the enumerating predicate with respect to a subject. An example task where the system returns a counting predicate is illustrated in Example 3.4.

Example 3.4: Annotation Task for Aligned Predicates.

Subject	Predicate	Object
Univ. of California, L.A.	institution^{-1}	T. Sowell, H. Demsetz ..(5 in total)
Univ. of California, L.A.	faculty size	4016

Topical relatedness of ***institution***$^{-1}$ *to* ***faculty size*** is: ∘ *High* ∘ *Moderate* ∘ *Low* ∘ *None.*
Enumeration of the objects in the query is: ∘ *Complete* ∘ *Incomplete* ∘ *Unrelated.*

The task in the opposite direction is designed similarly with the query containing a counting fact and the result, an enumerating fact with the set of objects.

For this task as well, we collected three judgements per predicate pair in either direction. We again used a graded relevance system by calculating a mean score of the two responses, where the grades for topical relatedness are {*High*: 1, *Moderate*: 0.67, *Low*: 0.33, *None*: 0} and for the completeness of enumeration we have {*Complete*: 1, *Incomplete*: 0.5, *Unrelated*: 0}. Thus, the graded relevance score (1 being the highest and 0 being the lowest) is calculated by mapping the responses to their grades and averaging over all responses. Concerning agreement, the pooled standard deviation of responses across pairs was 0.3 for topical relatedness and 0.46 for completeness of enumeration.

Table 3.3: Performance (precision, recall and F1) of the set predicate classifiers.

Model	Counting Predicate			Enumerating Predicate		
	Recall	**Precision**	**F1**	**Recall**	**Precision**	**F1**
Random	12.8	12.8	12.8	40.6	40.6	40.6
Logistic	51.2	19.0	27.7	**55.6**	51.7	53.5
Prior	48.7	20.2	28.5	**55.6**	51.0	53.5
Lasso	**71.7**	**23.3**	**35.1**	51.1	**59.6**	**55.0**
Neural	35.8	20.8	26.3	53.0	49.6	51.2

3.5.4 Set Predicate Classifiers

We model our classifiers on logistic regression as well as neural networks. However, due to small dataset size, and our interest in interpretable insights, we focus on multiple logistic regression models. We consider a standard *logistic* regression model, a logistic regression model with a weakly informative default *prior* [Gelman et al.(2008)], a *Lasso* regularized logistic regression [Tibshirani(1996)] and a *neural* network composed of a hidden layer of size three and sigmoid activation function. Due to the small training set, we use Leave-One-Out cross validation to obtain our model performance scores. All models are compared against a random baseline modelled on the input distribution, i.e., predicting labels at random, with probabilities proportional to label frequency in the training data.

3.6 Analysis

3.6.1 Classifier Model Selection

The results of the classifier models are in Table 3.3. As one can see, the Lasso regularized model performs the best for counting predicates with an F1 score of 35.1, which is significantly better than the random model which has an F1 score of 12.8. We observe that the counting classifier models in general have lower precision scores, but higher recall. The scores of the random model are computed from the training data distribution of counting predicates, which contain 39 positive and 306 negative datapoints. Note that the number of datapoints is less than the initial selection of 400 datapoints since, as explained in the previous section, we remove datapoints with divided agreements. We use the Lasso regularized model to classify the counting predicates.

In the enumerating predicate scenario also, the Lasso regularized model has an overall highest performance with an F1 score of 55. Here too, the random classifier performance depends on the distribution of training data which has 133 positive and 195 negative datapoints, giving an F1 score of 40.6. We use the Lasso regularized model for predicting the enumerating predicates. The comparable recall and precision scores of the enumerating predicate classifier can be attributed to the almost equal class distribution in the training data, which is not the case for counting predicates.

Important Features. The most important features in the counting predicate classifier are the mean and 10th percentile of the count values of a predicate with negative weights of 0.006 and 0.031 suggesting that counting predicates usually take smaller integer values. The predicate domain of type *Organization* has a positive weight of 0.14.

Table 3.4: F1 scores for the enumerating and the counting classifiers.

Train	Test (Enumerating Predicates)				Test (Counting Predicates)			
	DBP-raw	DBP-map	WD-truthy	FB	DBP-raw	DBP-map	WD-truthy	FB
DBP-raw	57.1	36.0	17.6	-	56.5	5.2	33.3	-
DBP-map	58.5	52.5	50.0	-	15.1	43.7	11.7	12.1
WD-truthy	54.7	51.4	52.0	-	25.9	20.4	32.0	8.5
FB	41.4	41.8	8.5	80.0	34.6	13.7	7.4	40.0
Random	39.4	33.3	33.3	60.8	18.7	11.1	14.2	14.2

The determining features of the enumerating classifier are the type information on the predicate domain and range. For example, the weights for domain *Thing* and range *Organization* are positive values of 0.135 and 0.046, respectively. It is interesting to note that predicate ranges of type *Work* and *Place* have small negative weights of 0.008 and 0.097, respectively, suggesting that predicates with range type location are less likely to be enumerating predicates.

Transferability A crucial aspect for our framework is whether it can be utilized on new KBs without requiring too much adaptation. Our modular framework is aimed towards this purpose. The supervised predicate classification stage allows to transfer our approach by only creating new training instances.

We evaluate the transferability of set predicate identification models trained on one KB, evaluated on the others. We do this for all combinations of KBs and report the F1 scores for the enumerating predicates in and the counting predicates in Table 3.4. In each setting we also report the F1 scores of a random baseline.

We observe that in most cases, the classifiers significantly outperform the random baseline, although the performance is quite low when evaluating on the Wikidata-truthy and Freebase KBs. The task of predicting enumerating predicates in the two variants of DBpedia is better when trained on any of the remaining KBs, whereas no classifier can predict enumerating predicates in Freebase. In the counting predicate prediction, the classifier trained on DBpedia-map performs worse than random classifier for all test data. Here too, the performance of classifiers on Freebase is quite low. In contrast, the classifiers trained on Freebase perform well on DBpedia in both cases. We can conclude that the case of counting predicate prediction is more challenging, given that the F1 scores of the random classifier and the training data are lower. Additionally, training on a single KB does not fare well in the case of counting predicate prediction.

For KBs where textual predicates are unavailable, a sensible extension is the incorporation of latent representations of predicates [Wang et al.(2014), Lin et al.(2015)] as separate features in the classification stage and by considering cosine similarity of predicate embeddings as a heuristic in the predicate alignment.

3.6.2 Prediction Quality

The number of set predicates predicted by each classifier is shown in Table 3.5 in the *Output* columns. We have 10,396 predicted as counting predicates by the counting classifier out of 26,156. The percentage of predicted counting predicates is almost 40%, which is much higher than the class distribution in the training data (11%). One reason is the very low precision scores of the classifiers,

Table 3.5: Predicted set predicates across different KBs, where **Input** is the number of KB predicates to be labelled (direct + inverse), **Output** is the number of positively labelled classifier prediction and **Filtered** is the number (and percentage in brackets) of predicates remaining after removing predicates related to IDs and codes.

KB	Enumerating Predicates			Counting Predicates		
	Input	**Output**	**Filtered**	**Input**	**Output**	**Filtered**
DBP-raw	16,635	4,090	4,090 (24.5%)	13,394	5,853	5,853 (43.6%)
DBP-map	1,670	308	308 (18.4%)	1,127	898	898 (79.6%)
WD-truthy	4,067	216	203 (4.9%)	3,346	1,922	1,067 (31.8%)
Freebase	13,872	7,752	7,614 (54.8%)	8,289	1,723	1,687 (20.3%)
Total	36,244	12,366	12,215 (33.7%)	26,156	10,396	9,505 (36.3%)

which may lead to more false positives. The enumerating classifier predicts 12,366 (34%) of 36,244 predicates as enumerating predicates which is closer to the class distribution seen in the training data (40%).

We illustrate some correctly and incorrectly predicted set predicates in Table 3.6. The DBpedia raw KB predicate `voiceActor`$^{-1}$, for example, connects a voice actor to the associated shows[7] and `employees`[8] gives the number of employees in an organization.

The classifiers also misclassify as shown in previous tables, for example, the counting classifer wrongly predicts dates like `birthYear` and `foundingYear`, measurements such as `km`, `height` as counting predicates. The enumerating classifier makes errors by positively labelling functional and pseudo-functional predicates like `currentTeam`, `sourceOfIncome`.

Filtering Identifier Labels Our classifiers, especially the counting classifier, has lower precision than recall. One of the commonly occurring type of predicates are identifiers, which may be represented as a fact with a large number in integer or string format and, we can remove such predicates without losing any actual set predicate. The filtering is done by checking for the presence of the words *'id'* and *'code'* as substrings, but not part of a longer word, in the predicate label, irrespective of the source KB. In Table 3.7 we compare the number of identifier predicates that need to be filtered before classification versus the number of predicates filtered after classification. The enumerating classifier is good at filtering identifier predicates, since almost 90% of the identifier labels are predicted to be false. The counting classifier removes around 59% of the identifier predicates and could benefit from the identifier filter. Thus, we apply the identifier label filter only on the output of the classifiers, and the final numbers are shown in the *Filtered* columns of Table 3.5.

3.6.3 Predicate Alignment

The *NDCG* scores reported in Table 3.8 are an evaluation of the top three alignments from all nine alignment metrics based on relevance judgments collected from crowd workers. We report the *NDCG* at positions 1 and 3. The table is divided into the three alignment families, and we consider

[7] List of shows Mel Blanc voiced over `https://tinyurl.com/dbpedia-mel-blanc`
[8] Sample of subjects with the predicate `employees` `https://tinyurl.com/dbpedia-employees`

Table 3.6: Correct and incorrect set predicate prediction from the different KBs.

KB	Counting Predicates	Enumerating Predicates
Correct Predictions		
DBP-raw	employees, retiredNumbers, crewMembers, postgraduates, members	college, workInstitution, affiliations, members^{-1}, voiceActor^{-1}, nativeLangugae^{-1}, politicalParty^{-1}
DBP-map	numberOfStudents, facultySize, numberOfGoals, populationAsOf, capacity	recordLabel, developer, product, publisher, formerCoach^{-1} employer^{-1}, governor^{-1}
WD-truthy	employees, numberOfDeaths, numberOfConstituencies, numberOfSeats	participantOf^{-1}, airlineHub, developer, father^{-1}, sponsor
FB	children, numberOfMembers, population, numberOfStaff, injuries, passengers	actor, member, starring, publisher, airportsServed^{-1}, foundedLocation^{-1}
Incorrect Predictions		
DBP-raw	linecolor, km, birthyear	currentTeam, deathCause, weightClass
DBP-map	foundingYear, keyPerson	secondTeam, genre
WD-truthy	publicationDate, coordinateLocation	parentOrganization, hairColor
FB	maxLength, height	cameras, burstCapability, founder

two directions. The first is the direction from a counting predicate to its enumerating predicate alignments, and the second is the reverse.

Based on the scores presented in Table 3.8, we can conclude that the linguistic similarity metric of cosine similarity (defined in Sec. 3.4.2C) performs the best individually, except for *NDCG@3* for the counting to enumerating direction, where the Pearson correlation measure performs best. The *Correlation* metric in the counting to enumerating direction and the *P'fectMR* metric in the reverse direction are the best performing metrics of the set predicate value distribution family (defined in Sec. 3.4.2B). The strongest metrics in the set predicate co-occurrence family (defined in Sec. 3.4.2A) are $Conditional_E$ in the direction of counting to enumerating predicate alignment and *P'wiseMI* in the other direction.

The *Combined* metric takes the best performing metric from each family and computes the mean of the alignment scores to obtain a combined score which gives better results than any individual metric. We use this combined measure to rank our alignments.

Table 3.7: The number of identifier predicates present in the input to the classifiers **Pre-filter**. Number of identifiers successfully removed as a by-product of classification vs. the number present in the predicted predicates removed using a **Post-filter**.

Class	**#Identifiers in input**	**Successfully classified as non-set-predicates**	**Removed using a Post-filter**
Enumerating	2,167	2,016 (93%)	151
Counting	2,158	1,277 (59%)	881

Table 3.8: Average *NDCG* scores for the alignment stage.

Metric	**C → E**		**E → C**	
	@1	@3	@1	@3
Absolute	0.71	0.56	0.62	0.63
Jaccard	0.76	0.61	0.69	0.67
$Conditional_C$	0.71	0.56	0.68	0.67
$Conditional_E$	0.76	0.68	0.62	0.63
P'wiseMI	0.73	0.58	0.71	0.70
P'fectMR	0.70	0.57	0.73	0.72
Correlation	0.77	0.69	0.62	0.61
P'tileVM	0.72	0.57	0.65	0.65
CosineSim	0.79	0.61	0.74	0.73
Combined	**0.84**	**0.67**	**0.75**	**0.75**

3.6.4 Discussion

While the experimental results are encouraging, F1-scores of $35 - 55\%$ for the identification stage indicate that there is still much room for improvement. Further feature engineering, e.g., regarding IDs and codes, and further distribution measures, may still yield moderate improvements, but we see three principled challenges that limit any fully-automated approach: (i) statistical cues allow ruling out well some clear negatives, but for many infrequent predicates, provide only weak signals. (ii) textual cues for predicates are mostly short and thus of limited informativeness. (iii) input KBs come with a considerable level of noise.

Where possible, we would therefore recommend executing set predicate identification and alignment not in a fully automated manner, but employ a human-in-the-loop process, where statistical procedures narrow the search space for human annotators, or human annotators focus on the "fat head" (e.g., the 100 most used predicates), and automated methods focus on the long tail. In industrial deployment of the question answering use case, human input could also come from user feedback, e.g., query-click-logs or query reformulations and follow-up questions.

Inverse Predicates. For generalizability, our method currently does not incorporate existing definitions of inverse predicates. Freebase contains $12K$ such definitions (via `owl:inverseOf` [Chah(2018)]), Wikidata has 136 (identified via the meta-property *inverse property*), DBpedia has three mentions in comment fields. Incorporating such KB-specific constraints could further boost the accuracy of alignments.

Indirect Alignments. The alignments are also helpful in identifying redundancies in schema, where two or more set predicates (enumerating/counting) describing the same concept exist. For example, the enumerating predicate `affiliation` in the DBpedia mapping-based KB aligns with the counting predicates {`facultySize`, `staff`, `numberOfStaff`}, though the counting predicates seldom co-occur. Hence, we could also infer semantic groups, such as, {`employer`, `affiliation`, `workInstitution`}, which are used interchangeably in the KB.

Multi-hop Alignments. Counting predicates may well align with multi-hop paths of enumerating predicates. For instance, an interesting near-subset of `population(x,y)` is `worksAt(y,z), basedIn(z,x)`. The search space for such alignments would grow quadratically, but clever pruning may keep it manageable.

Crowd Annotation Costs. The cost of the annotating classifier training data is approximately 0.13$ per task per judgment. For the alignment evaluation task the cost was almost 0.40$ per task per judgment. Thus the average cost per task is $0.5(0.13*3+0.4*3) \approx 0.80$$ if we collect 3 judgments. Given the average time spent per task, we arrive at an hourly pay of $14, which corresponds to the salary of student assistants in Germany in 2019.

Open information extraction So far we have only considered the alignment of canonicalized KB predicates. An interesting direction would be to extend this alignment towards open information extraction and open knowledge bases in the style of Reverb [Fader et al.(2011)], i.e., to align textual phrases like *"X has Y employees"* with phrases like *"Z works at Y"*, *"Z recently joined X"*, etc. Numeric open information extraction traditionally focuses on temporal information [Ling and Weld(2010)] and measures [Saha et al.(2017)], though there are also some recent works on counting information extraction [Mirza et al.(2017), Mirza et al.(2018)], which one might build upon.

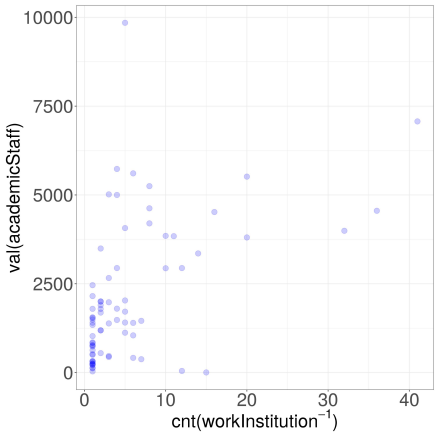
3.7 Use Cases

Question Answering. Question answering benefits from set predicate alignment at three stages: natural language parsing, query result debugging, and query result enrichment.

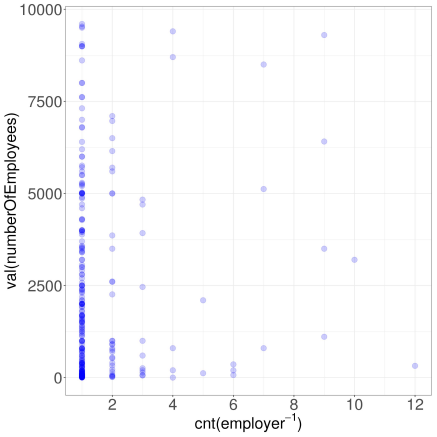
In *natural language parsing*, state-of-the-art KBQA systems typically generate a set of candidate parses, which subsequently are ranked, and the top one executed (with research prototypes often allowing the user to inspect and choose among them). Set predicate alignments could be used to generate further candidates, or used as ranking feature.

Structured queries over KBs often produce empty or otherwise unexpected results that require *query debugging*. Set predicate alignments could help to understand whether an ambiguous predicate name refers to the intended predicate, and related counts could help to understand that a predicate is an intended one, but that just the KB is incomplete.

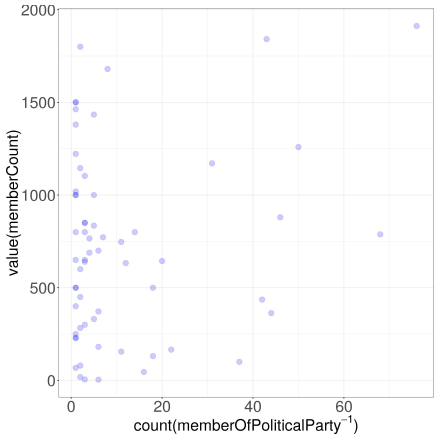
KB Curation. In this section, we look into a few alignments from different KBs and the distribution of their values. The first alignment in Figure 3.3a is the pair ($\texttt{workInstitution}^{-1}$, `academicStaff`) from the DBpedia raw KB, which co-occurs across 76 subjects (institutions). Each point (x, y) in the plot represents an institution, which is connected to x entities by the predicate $\texttt{workInstitution}^{-1}$, and which takes the value y for the predicate `academicStaff`. In an ideal



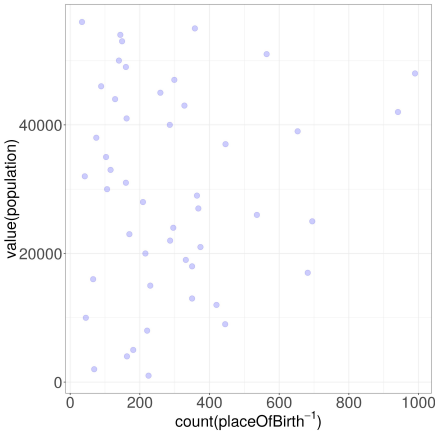
(a) `academicStaff` and `workInstitution`$^{-1}$ across 76 subjects in DBpedia raw.



(b) `numberOfEmployees` and `employer`$^{-1}$ across 278 subjects in DBpedia mapping-based KB.



(c) `memberCount` and `memberOfPoliticalParty`$^{-1}$ across 62 subjects in Wikidata-truthy KB.



(d) `populationState` and `placeOfBirth`$^{-1}$ across 48 subjects in Freebase KB.

Figure 3.3: Distribution of counts and enumerations per subject for different counting and enumerating predicate pairs.

condition the count of instances should match the value and all points should lie along the line $y = x$. Points lying above this line suggest incompleteness. Such is the case in Figure 3.3a where the predicate `workInstitution`$^{-1}$ is often only connected to the popular or important staff members.

Next we look into an alignment from DBpedia mapping based KB, (`employer`$^{-1}$, `numberOfEmployees`) in Figure 3.3b. In this alignment we also observe that the enumerated facts is much smaller than the number of employees, typically because such facts only exist for the most important employees.

In Figure 3.3c, we show an alignment from the Wikidata KB which is regarding the members of a political party (`memberOfPoliticalParty`$^{-1}$, `memberCount`). Similar to the previous trends, here also the number of enumerated facts about the members in a political party is less than the actual value. The final alignment we show is of the pair (`placeOfBirth`$^{-1}$, `populationState`) in the Freebase KB as shown in Figure 3.3d. From the numbers it seems that the predicate covers

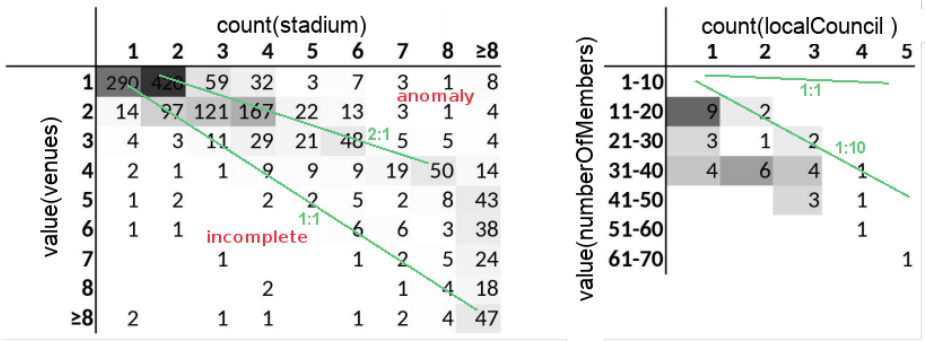


Figure 3.4: Value distribution of the counting predicate `venues` and count of enumerating predicate `stadium` for 2179 sports events (left), and `numberOfMembers` and count of $\texttt{localCouncil}^{-1}$ for 35 political assemblies (right) from the DBpedia raw KB.

small geographical locations where the number of enumerated facts is more complete than in the previous cases.

In each of the alignment figures there are at most two instances where the value of the counting predicate is less than the count of enumerated instances, there exist no such instances in the Figure 3.3d. Such a low number of points indicate anomaly or inconsistencies such as a backdated counting predicate value or, for instance, in Figure 3.3b where the value of counting predicate `numberofEmployees` is 0 even though there exist enumerated facts.

Figure 3.4 shows the value distribution of an alignment where each cell in position (x, y) gives the number of subjects which takes x as the value of the counting predicate and y number of instances with the enumerating predicate. The first analysis concerns the places where a sports event took place. A notable anomaly in DBpedia raw KB is that regularly, for each venue, both the stadium and the city were recorded. Thus, we plot two green lines showing 1:1 matches, and 2:1 matches. Instances below both lines likely point to incompleteness (some stadiums are missing), instances below both lines likely point to some errors in the data (i.e., too many stadiums added). As one can see, the completeness appears to be relatively high, while there are several cases that deserve closer inspection with respect to possible incorrectness.

The second analysis concerns the number of members of local councils compared with individual members listed. Here, incompleteness is prevalent, with typically only 1 in 30 to 1 in 10 members listed.

3.8 The CounQER System

We develop the CounQER system to demonstrate how set predicate alignments highlight redundancies in the KB schema, enhance question answering by providing supporting counts and/or enumerations and help in KB curation. Figure 3.5 shows the interface with results on an example query on the DBpedia-raw KB. The query is on the events where the entity, *Leander Paes*, wins gold ($\texttt{dbp: gold}^{-1}$). The main result (set predicate in blue) is succeeded by related results on ranked and aligned set predicates (in orange). Enumerations expand on hovering, and we show up to 1000 enumerations. A user can check the actual query fired for each row by following the link to the SPARQL query endpoint. Also, KB-specific predicate statistics show up while hovering over the

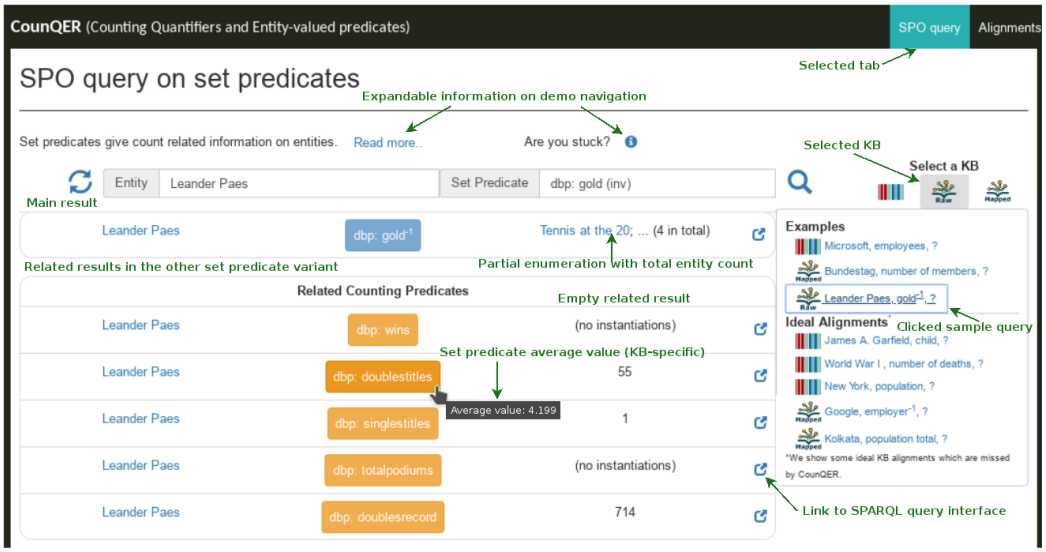


Figure 3.5: The interface for SPO queries, showing results on an example query.

predicate buttons. On clicking a set predicate from the related results, a new query is fired on the same subject and the new clicked predicate. The complete ranked list of set predicate alignments for the three KBs as well as Freebase can be viewed as in Figure 3.6. Here too, we provide links to the SPARQL endpoint showing the subjects that have populated facts for the alignments.

3.8.1 System Description

SPO query. The SPO query function provides two input fields, *Entity* and *Set Predicate*, and a KB selection button. The first field provides real-time entity suggestions from the selected KB, based on the input prefix, to the user to choose from. Next, the user selects a set predicate from the set predicate input field. The predicate choices are KB-specific and ordered by i) whether they are populated and have alignments, ii) they are populated but without alignments, and iii) they are unpopulated.

Upon execution, the input parameters are sent to our server, where we determine the variant of the user-selected set predicate — counting or enumerating. Then, from the KB-specific alignments containing the queried predicate, we shortlist the top-five highest scoring pairs to obtain related set predicate facts. If there are no alignments, we do not generate any related query. The server then fires the main query to the SPARQL endpoint of the corresponding KB followed by the SPARQL queries for the aligned set predicates, if present. Once these results are obtained, the server returns the results along with KB-specific predicate statistics, *i.e.,* the average value that the counting predicates take and the average number of entities per subject that the enumerating predicates take.
Alignments. CounQER provides an option of viewing all alignments across the four KBs along with their alignment scores. A user can go through the list ordered by the alignment score or use the search bar to filter matching set predicates and view their corresponding alignments. Each alignment has a link to SPARQL query API, where the user can view the list of subjects for which the predicate pair co-occur.
The main features of the interface are as follows.

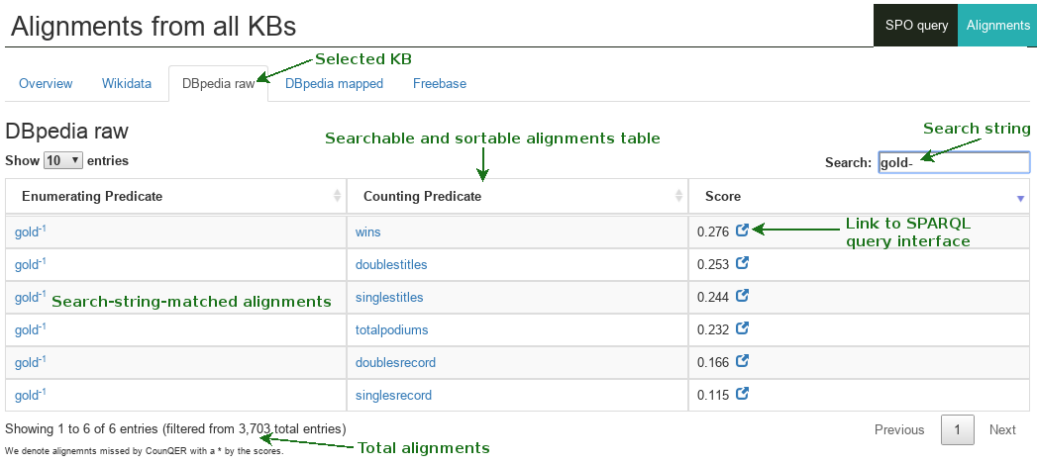
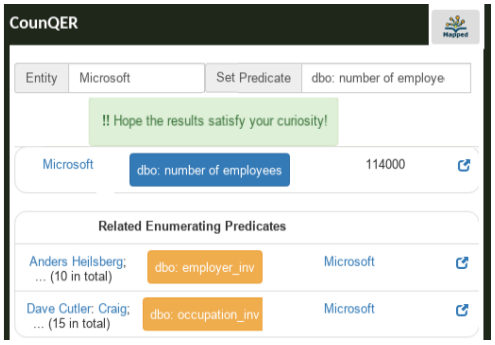
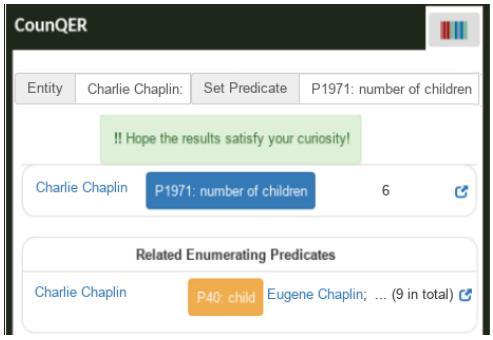


Figure 3.6: Interface for viewing KB-specific alignments.



(a) Employees in Microsoft (DBpedia-mapped).



(b) Children of Charlie Chaplin (Wikidata).

Figure 3.7: Results on alignment queries using CounQER (as of March 2020).

1. *Predicate suggestions* — Set predicates are ordered based on whether they are populated for the selected entity and whether alignments exist for them.
2. *Empty results* — If the main query returns an empty result, but, the predicate has populated alignments, CounQER shows the related results. Conversely, if the set predicate in the main query is populated and alignments exist for this predicate, we show the related results regardless of them being empty, thus highlighting potential incompleteness in the KB, w.r.t the queried entity.
3. *Links to SPARQL queries* — Every row in the results contains a link to the SPARQL endpoint, which a user can follow to check the actual query that was fired and also view enumerations of size more than 1000. Alignment tables also link to the SPARQL endpoint with queries which list subjects for which the set predicate pair co-occur.

We also show some manually added ideal alignments, *i.e.,* the alignments which are present in the investigated KBs but missed by the automated CounQER methodology. These alignments are also present in the table with a fictitious score between [0.9-1].

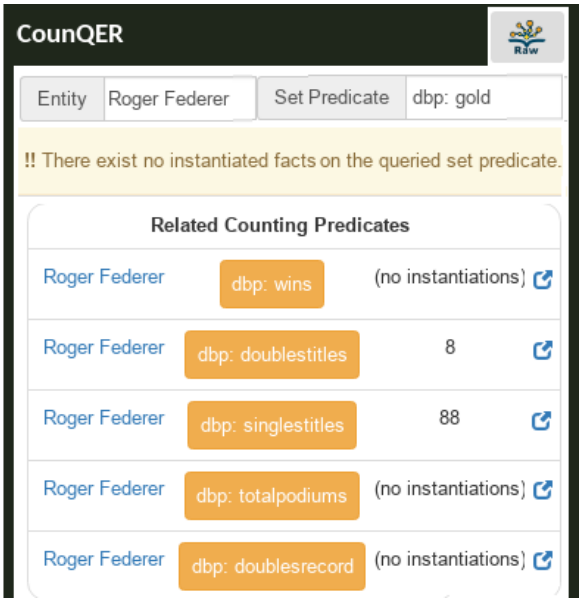


Figure 3.8: Empty enumerations on the original query about golds won by Roger Federer (DBpedia-raw), still return information from aligned counting predicates (as of March 2020)

3.8.2 Demonstration Experience

Scenario 1 - QA. In a query about the number of employees at Microsoft, CounQER finds the main result from the queried KB, DBpedia-mapped, to be 114,000 employees. In addition, CounQER returns instantiated facts on interesting enumerating predicates, such as, `employer`$^{-1}$ and `occupation`$^{-1}$ (see Figure 3.7a).

Scenario 2 - KB curation. Consider the example in Figure 3.7b, where the user searches for the number of children of the British comic actor, Charlie Chaplin. The alignment results reveal inconsistent information in Wikidata-truthy. While the value for `number of children` is 6, there are 9 statements for the enumerating predicate `child`.

Next, we investigate the winning titles of Roger Federer in DBpedia-raw (Figure 3.8). Even though a query on the golds won by Federer returns no main results, unlike the query on the golds won by Leader Paes in Figure 3.5, the counting predicates `doublestitles` (2^{nd}) and `singlestitles` (3^{rd}) give the number of doubles and singles titles won by Federer.

3.9 Conclusion

In this chapter we introduced the problem of set predicate alignment, and presented CounQER, a methodology for identifying and aligning counting predicates with enumerating predicates that combines co-occurrence, correlational and linguistic features. Our system demonstration highlights the utility of aligned set predicates to enhance count information in KBs. We have shown that automated methods can identify and align set predicates on four diverse knowledge bases, and that these alignments are useful for use cases in knowledge base curation and count question answering.

We believe that understanding set predicate semantics in today's KBs is an important step

towards a better interaction with structured world knowledge repositories. In this direction, we identify relevant open challenges in tackling count information in KBs. *Firstly*, the sparsity in KB leads to weak global signals as we observe with our alignment scores. Future work could look into a small focused set of entities to identify set predicates alignments with stronger signals, such as number of wins/losses and matches played for the set of tennis players, or number of seasons and has part(s) for TV series. *Secondly*, set predicate alignments can be extended to identify semantic groups, redundant predicates and multi-hop alignments in KBs. In the next chapter, we focus on counts available in text on the Web.

Chapter 4

Count Information in Web Text

Contents

4.1 Introduction

In this chapter we address the challenging case of answering count queries, such as *number of songs by John Lennon*, from web text where we have many relevant documents returned for a single query

and the challenge is to then consolidate this into a structured and comprehensive answer for the end user. We propose a methodology for answering count queries from web snippets returned by a search engine result page. Our method infers final answers from multiple observations. It supports semantic qualifiers for the counts and contextualizes the counts into semantic groups. The method also provides evidence by enumerating representative instances obtained from these snippets. We present our dataset, which we use for the training and evaluation of our method. Finally, we present our demonstration system, which is available online for making count queries on the Web.

Count queries can have multiple correct answers due to variance in semantic qualifiers, variance in the count values due to estimations, close counts and time dependencies, and alternative representations through instances. Such queries remain underexplored and pose open challenges.

Example 4.1: Count Queries

Here, we observe text spans containing relevant counts present in different web search snippets for a given count question.

- *How many songs did John Lennon write for the Beatles?*
  The text spans — *almost 200 Beatles songs*, *180 songs under the Lennon-McCartney partnership* — have variations in the count value and the qualifiers.
- *How many languages are spoken in Indonesia?*
  The text span — *around 700 languages including Bahasa, Indonesian and Javanese* — has incomplete, but exemplary instances.
- *How many unicorn companies are there?*
  The text spans — *over 1,200 unicorn companies*, *580 new unicorns* and *1000+ unicorn startups* has a high distribution variance.

Count queries are frequent in search engine logs, as well as QA benchmarks [Voorhees(2001), Rajpurkar et al.(2016), Kwiatkowski et al.(2019), Dubey et al.(2019)]. If the required data is in a structured KB, such as Wikidata [Vrandečić and Krötzsch(2014)], then answering is relatively straightforward. However, as we saw in Chapter 3, KBs are limited not only by their sparsity but also by the lack of direct links between instances and explicit counts when both are present. In Figure 4.1, *160* is the output to a SPARQL query, which counts the number of songs by Lennon performed by The Beatles[1]. The query translation, here performed by an expert user, is a challenge in itself and is beyond the scope of this work. Nevertheless, for a user who is unaware of the composer duo Lennon-McCartney, would not know that the output of 160 songs to the SPARQL query contains songs jointly written by Lennon with his co-band member and are not separated from his individual contributions to the band.

Search engines are the commercial state-of-the-art that are exposed to real-life user queries. They handle popular cases reasonably well, but also fail on semantically refined requests (e.g., *for the Beatles*), merely returning either a number without explanatory evidence or multiple candidate answers with high variance (Figure 4.1). We also see incorrect spans being highlighted *22 songs* in the second snippet, which is actually a count of songs written by another Beatles band member, George Harrison.

[1] `https://w.wiki/4XVq`: last queried on June 2022.

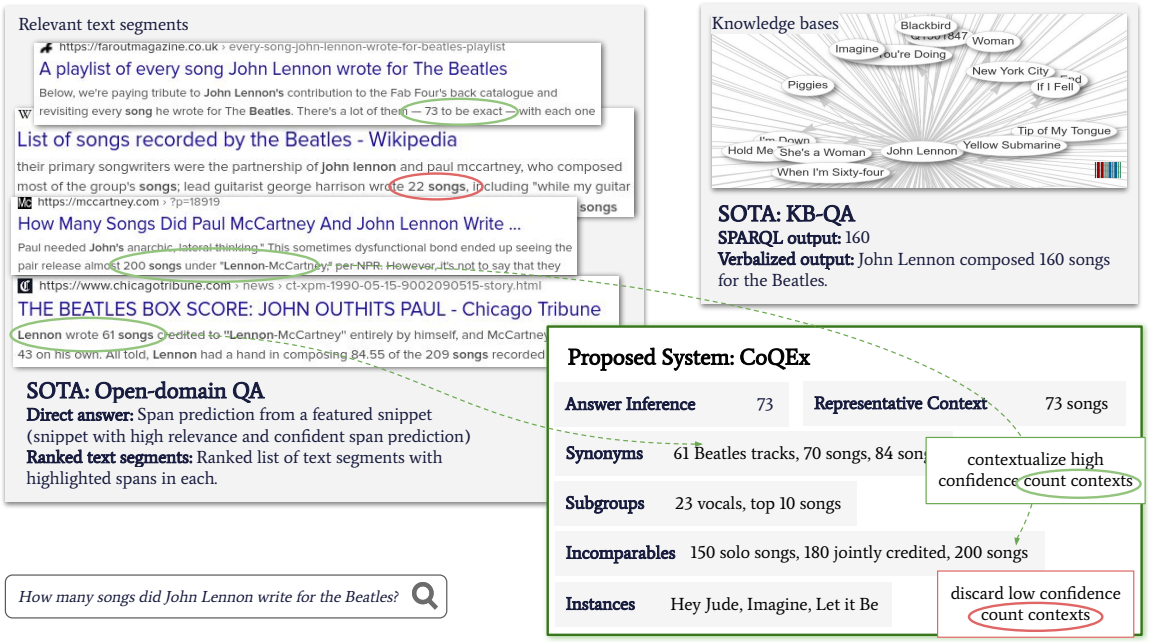


Figure 4.1: User experience with state-of-the-art QA systems against our proposed methodology for count question answering. In the existing setting, a user would often encounter varying counts in returned text snippets (open-domain QA), or limited context (KB-QA). CoQEx provides a more comprehensive answer, by aggregating evidence across text snippets, contextualizing contexts and providing instances as count explanations.

Answering count queries from web text thus poses several challenges:

1. *Aggregation and inference:* Returning just a single number from the highest-ranked page can easily go wrong. Instead, joint inference over a set of candidates, with an awareness of the distribution and other signals, is necessary for a high-confidence answer.

2. *Contextualization:* Counts in texts often come with contexts on the relevant instance set. For example, John Lennon co-wrote about 180 songs for the Beatles, 150 as a solo artist, etc. For correct answers, it is crucial to capture context from the underlying web pages and properly evaluate these kinds of semantic qualifiers.

3. *Explanatory Evidence:* A numeric answer alone, such as *180* for the Beatles songs by Lennon, is often unsatisfactory. The user may even perceive this as non-credible, and think that it is too high, as they may have only popular songs in mind. It is, therefore, crucial to provide users with explanatory evidence.

Contribution We present CoQEx, Count Question answering with Explanatory evidence, which answers count queries via three components: i) *answer inference* ii) *answer contextualization*, iii) *answer explanation.*

Given a full-fledged question or a telegraphic query and relevant text segments, CoQEx applies joint inference to compute a high-confidence answer for the count itself. It provides contextualization

of the returned count answer, through semantic qualifiers into equivalent or subclass categories, and extracts a set of representative instances as explanatory evidence, exemplifying the returned number for enhanced credibility and user comprehension.
Contributions of this work are:

1. introducing the problem of count query answering with explanatory evidence;
2. developing a method for inferring high-confidence counts from noisy candidate sets;
3. developing techniques to provide answer contextualization and explanations;
4. evaluating CoQEx against state-of-the-art baselines on a variety of test queries;
5. creating an annotated data resource with 5k count queries and 200k text segments.

4.2 Related Work

In the KB-QA domain, SPARQL-based systems like AQQU [Bast and Haussmann(2015)] and QAnswer [Diefenbach et al.(2019)] tackle count queries by aggregating instances using the SPARQL `count` modifier. This is liable to incorrect answers when instance relations are incomplete, which is often the case for KBs. QALD [Usbeck et al.(2018)], LC-QuAD 2.0 [Dubey et al.(2019)] and WebQuestions [Berant et al.(2013)] are popular KB-QA benchmarks. In the text domain, current QA systems operate in the retriever-reader paradigm, where a retriever model returns relevant text snippets (such as paragraphs) from a source (usually Wikipedia) and a reader model returns a final answer. Datasets like SQuAD [Rajpurkar et al.(2016)] and DROP [Dua et al.(2019)] focus on the latter part, i.e., machine reading and comprehension. Transformer-based encoder language models like BERT [Devlin et al.(2019)] and its descendants [Sanh et al.(2019), Joshi et al.(2020)] perform quite well on this task. Open-domain QA systems, have both retriever and reader components, where the reader is extracts relevant text span or generates the answer [Chen et al.(2017), Karpukhin et al.(2020), Lewis et al.(2020), Izacard and Grave(2021)]. TriviaQA [Joshi et al.(2017)], WebQuestions [Berant et al.(2013)], NaturalQuestions [Kwiatkowski et al.(2019)] are popular benchmarks for open-domain QA. Attempts have also been made to improve recall by hybrid QA over text and KB, yet without specific consideration of counts [Xu et al.(2016), Lu et al.(2019), Saha Roy and Anand(2020), Yu et al.(2022)]. The latest development in this direction is using retrieval-augmented generation with LLMs, which we discuss in detail in Chapter 6.

Most QA benchmarks focus on reasoning capabilities of a model based on chaining evidences, which does not directly apply to counts about entity sets. Even so, count questions form 5%-10% of popular QA datasets [Mirza et al.(2018)]. Evaluation metrics for reading-comprehension style benchmarks typically employ strict matching requirements, like exact match and bag-of-words overlap to compute an F1 score, as introduced by the SQuAD benchmark [Rajpurkar et al.(2016)]. On a question level, these metrics measure the token-level overlap of the prediction with ground truth answers. This does not transfer well to numeric queries, especially counts where the variance is numeric rather than lexical. We propose relaxed metrics for evaluation in Section 4.6.

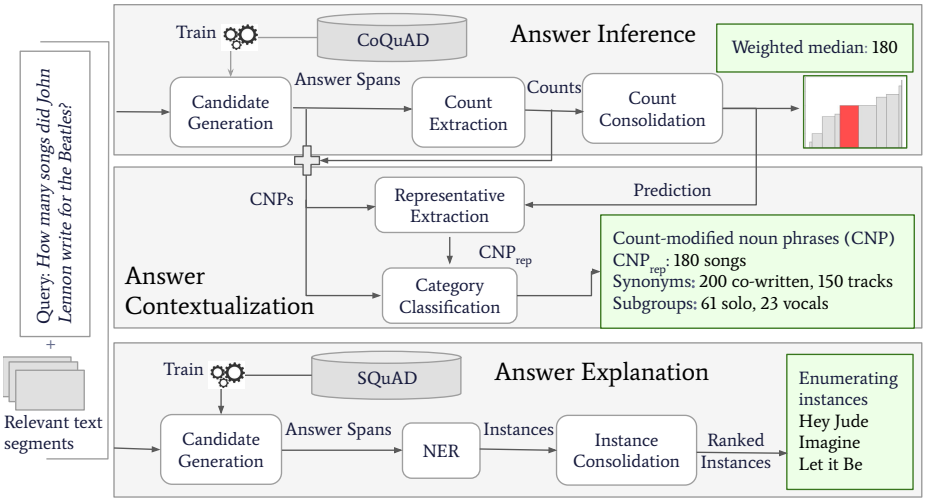


Figure 4.2: Overview of the CoQEx methodology.

4.3 Design Rationale

Traditional open domain QA architectures involve query analysis, document retrieval and answer extraction, where named-entity recognition (NER) is an important component for recognizing named entities of the answer type [Zhu et al.(2021)], with the Text REtrieval Conference (TREC) QA tracks leading the research in fact-based question answering [Voorhees(2001)]. Current research employs deep learning with the benefit of achieving end-to-end trainable systems. In this direction we discussed open-domain QA systems in the reader-retriever paradigm, KB-QA which translates natural language questions to structured queries finally executed over a KB to return the answer and the hybrid QA setting which uses a mix of structured KBs and texts to answer natural language questions.

We approach count query answering by a combination of per-document answer span prediction, context extraction, and consolidation of counts and instances across documents. Figure 4.2 gives the overview of CoQEx. We consider as input a query that asks for the count of named entities that stand in relation with a subject, for instance, full queries like *How many songs did John Lennon write for the Beatles*, or a keyword query like *songs by lennon*.

We further assume that relevant documents or passages are given. This could be the result of a standard keyword/neural embedding-based IR procedure over a larger (locally indexed) background corpus, like Wikipedia or the Web. We utilize snippets returned by search-engine results, since this allows up to capture the count distribution across sources, unlike Wikipedia which has low variability. We explain the methodology of CoQEx in the next section.

4.4 The CoQEx Methodology

CoQEx extracts counts and instances (entity-mentions) from the text segments to subsequently i) consolidate the counts to present the best answer, ii) present contextualization as a means to semantically qualifying the predicted count, and iii) ground the count in instances. We denote the set of relevant text segments for a query by D.

4.4.1 Answer Inference

In order to generate count candidates, we use the popular SpanBERT model [Joshi et al.(2020)], trained on the CoQuAD train split. Span prediction models return general text spans, which may contain worded answers (*five children*, $Conf = 0.8$), modifier words and other context (*17 regional languages*, $Conf = 0.75$), where *Conf* is the confidence score of the model. These answer spans have two components — the count itself and qualifiers, which we separate with the help of fixed rules and the CogComp Quantifier by [Roy et al.(2015)].

Algorithm 1 shows the outline for answer inference. We run all relevant documents, D for a given query through the span prediction model to get the candidate spans comprising the answer span, $c.Span$, and model confidence, $c.Conf$, (`Line 3-4`). If the span is non-empty and confidence of the model is higher than a threshold, θ, then we extract integer count from the span using the ExtractCount function (`Line 5-6`). If the integer extraction in non-empty, we save the span, the extracted integer and the model confidence (`Lines 7-9`).

The ExtractCount() sequentially applies rule-specific conversions before applying CogComp Quantifier to achieve maximum recall. The function first applies type conversion (`int`(*17*)→*17*, `float`(*17.0*)→*17.0*), followed by a dictionary based look-up for worded to integer conversion (*seventeen*→*17*) and lastly the CogComp Quantifier, proceeding only when previous conversions yield empty results. The counts are further cleaned by removing fractions $\in (0, 1)$, since counts are whole numbers.

If there is at least one count extracted from the relevant document set, we consolidate the counts using either of the proposed consolidation methods defined in the Consolidate function (`Lines 10-13`), else the answer inference is empty (`Line 15`).

To consolidate the resulting candidate counts into a prediction C_{pred}, we compare four methods:

1. *Most confident:* The candidate given the highest confidence by the neural model. This is commonly used in textual QA [Chen et al.(2017), Wang et al.(2018)].

2. *Most frequent:* A natural alternative is to rank answers by frequency, and prefer the ones returned most often.

While *most confident* may be susceptible to single outliers, *most frequent* breaks down in cases where there are few answer candidates. But unlike textual answers, numbers allow further statistical aggregation:

3. *Median:* The midpoint in the ordered list of candidates.

4. *Weighted Median:* The median can be further adapted by weighing each candidate with the model's score.

Example 4.2: Count Aggregation.

Candidate Set $\{150_{0.9}, 160_{0.8}, 180_{0.4}, 180_{0.4}, 210_{0.3}\}$ (confidences as subscripts)

Most confident:	150
Most frequent:	180
Median:	180
Weighted median:	160

Algorithm 1 Answer inference

Input: Count query, q,
set of relevant text segments, D,
span prediction model, SpanPrediction,
span selection threshold, θ,
count extraction function, ExtractCount,
consolidation function, Consolidate.

Output: Answer Inference, C_{pred},
List of count span and extracted integer tuples, C.

1: $C \leftarrow \{\}$ ▷ *Passed to Algorithm 2*
2: $WeightedC \leftarrow \{\}$ ▷ *Counts with confidence*
3: **for** $d \in D$ **do**
4: $c \leftarrow$ SpanPrediction(d, q)
5: **if** $c.Span \neq None$ *and* $c.Conf > \theta$ **then**
6: $i \leftarrow$ ExtractCount$(c.Span)$
7: **if** $i \neq None$ **then**
8: $C \leftarrow C \cup (c, i)$
9: $WeightedC \leftarrow WeightedC \cup (i, c.Span)$
10: **if** $WeightedC \neq \{\}$ **then**
11: $WeightedC \leftarrow$ SortAscending$(WeightedC)$
12: ▷ *Return the weighted median of the counts.* ◁
13: $C_{pred} \leftarrow$ Consolidate$(WeightedC)$
14: **else**
15: $C_{pred} \leftarrow Null$
16: **return** C_{pred}, C

4.4.2 Answer Contextualization

The answer candidates from the previous module often contain nouns with phrasal modifiers, such as ***17 regional languages***. We call these ***count-modified noun phrases*** (CNPs). These CNPs stand in some relation with the predicted count from the answer inference module as explained in Algorithm 2. The representative CNP, CNP_{rep}, which best accompanies the predicted count is first chosen and then compared with the remaining CNPs. Since answer inference uses a consolidation strategy, we select the CNP with count equal to C_{pred} having the highest confidence as CNP_{rep} (`Line 2`).

The remaining CNPs are categorized as follows:

1. *Synonyms*: CNPs, whose meaning is highly similar to CNP_{rep} and accompanying count is within a specified threshold, α, of the predicted count (`Lines 6-7`), where α is between 0% and 100%, 0 being most restrictive.

2. *Subgroups*: CNPs which are semantically more specific than CNP_{rep}, and are expected to count only a subset of the instances counted by CNP_{rep}, such that the accompanying count is lower than the synonyms set (`Lines 8-9`).

3. *Incomparables*: CNPs which count instances of a completely different type indicated by negative cosine similarity (`Lines 4-5`) or an accompanying count higher than the synonyms (`Line 11`).

Algorithm 2 CNP Category Classifier

Input: Answer Inference, C_{pred},
synonym threshold, α,
list of count spans and extracted integer tuples, C (from Algorithm 1)

Output: Representative CNP, CNP_{rep},
List of CNP categories, *Categories*

1: $Synonyms,\ Subgroups,\ Incomparables \leftarrow \{\}, \{\}, \{\}$
2: $CNP_{rep} \leftarrow \underset{c}{\text{argmax}}\{c.Conf \mid i = C_{pred}, (c, i) \in C\}$
3: **for** $(c, i) \in C \setminus (CNP_{rep}, C_{pred})$ **do**
4: **if** CosineSim($c.Span$, $CNP_{Rep}.Span$) <= 0 **then**
5: $Incomparables \leftarrow Incomparables \cup c$
6: **else if** $i \in C_{pred} \pm \alpha$ **then**
7: $Synonyms \leftarrow Synonym \cup c$
8: **else if** $i < C_{pred} - \alpha C_{pred}$ **then**
9: $Subgroups \leftarrow Subgroups \cup c$
10: **else**
11: $Incomparables \leftarrow Incomparables \cup c$
12: **return** $CNP_{rep}, Synonyms, Subgroups, Incomparables$

We assign these categories based on (textual) semantic relatedness of the phrasal modifier, and numeric proximity of the count. For example, ***regional languages*** is likely a subgroup of ***700 languages***, especially if it occurs with counts $\langle 23, 17, 42\rangle$. ***tongue*** is likely a synonym, especially if it occurs with counts $\langle 530, 810, 600\rangle$. ***Speakers*** is most likely incomparable, especially if it co-occurs with counts in the millions. CNPs with embedding-cosine similarity [Reimers and Gurevych(2019)] less than zero are categorized as incomparable, while from the remainder, those with a count within $\pm\alpha$ are considered synonyms, lower count CNPs are categorized as subgroups, and higher count CNPs as incomparable.

For instance, for the query on the *languages spoken in Indonesia* in Example 4.3 with an inference of 700, the reference CNP is the highest scoring CNP with an equal count. Here, CNP_{rep} is *estimated 700 languages*. {***700 languages***, ***750 dialects***} would be classified as synonyms, {***27 major regional languages***, ***5 official languages***} as subgroups and {***2000 ethnic groups***, ***85 million native speakers***} as incomparables.

Example 4.3: CNP Categories

Query	*How many languages are spoken in Indonesia?*
Candidate set	$\{5_{(0.8)}, 27_{(0.6)}, 700_{(0.7)}, 700_{(0.8)}, 750_{(0.7)}, 2{,}000_{(0.4)}, 85{,}000{,}000_{(0.5)}\}$
Answer Inference	700
CNP_{Rep}	estimated 700 languages$_{(0.8)}$
Synonyms	700 languages$_{(0.7)}$, about 750 dialects$_{(0.7)}$
Subgroups	27 major regional languages$_{(0.6)}$, 5 official languages$_{(0.8)}$
Incomparables	2000 ethnic groups$_{(0.4)}$, 85 million native speakers$_{(0.5)}$

4.4.3 Answer Explanation

Beyond classifying count answer contexts, showing relevant sample instances is an important step towards explainability. To this end, we aim to identify entities that are among the ones counted in the query using Algorithm 3.

Let I denote the inverted dictionary of instances where $I[i]$ contains the text IDs and confidence scores of the instance i. We collect the answers from a QA model to create a more precision-oriented candidate space. We again use the SpanBERT model (fine-tuned on SQuAD 2.0 dataset) to obtain candidates (tuples comprising answer span, *c.Span*, and the model confidence, *c.Conf*) from every document (`Lines 4-5`), this time with a modified query, replacing *"how many"* in the query with *"which"* (or adding it), to not confuse the model on the answer type (`Line 1`). If the span is non-empty and has a confidence higher than threshold, θ, we extract named entities from the span (`Lines 7-8`) using an off-the-shelf NER. We create an inverted index of these instances, keeping track of the text segment it belongs to and the span prediction (`Lines 9-10`). The instances are then scored globally using either of the following alternative consolidation scoring approaches defined in the ConScore function in `Lines 11-12`. The instances are then ranked in decreasing order of their consolidated scores (`Line 13`).

The alternatives for instance consolidation are as follows. We normalize the consolidation scores for comparison across instances and strategies. All consolidation strategies lie between $[0, 1]$.

1. *QA w/o Consolidation.* In the spirit of conventional QA, where results come from a single document, we return instances from the document with the most confident answer span.
2. *QA + Context Frequency.* The instances are ranked by their frequency, $S[i] = \frac{|I[i]|}{|D|}$.
3. *QA + Summed Confidence.* We rank the instances based on the summed confidence of all answer spans that contain them, $S[i] = \frac{\sum_{(\cdot,c)\in I[i]} c.\text{Conf}}{|I[i]|}$.
4. *QA + Type Compatibility.* Here, instances are ranked by their compatibility with the query's answer type, extracted via the dependency parse tree. We obtain the answer type by extracting the first `noun` token and any of its preceding `adjectives` from the dependency parse tree of the query. We form a hypothesis *"(instance) is a (answer type)"* and use the probability of its entailment from the parent sentence in the context from which the instance was extracted to measure type compatibility. We use a transformer-based textual entailment model by [Liu et al.(2019)] to obtain entailment scores, which are again summed over all containing answer

spans, such that, $S[i] = \frac{\sum_{(d,c)\in I[i]} \text{ENT}(i,d,c,q)}{|I[i]|}$. Here, the function ENT takes the instance i, the answer spans c to determine the parent sentence in the text segment d, and query q to determine the answer type for the hypothesis.

Algorithm 3 Extracting answer explanations

Input: Count query, q,
set of relevant text segments, D,
span predictor model, SPANPREDICTION,
candidate selection threshold, θ,
named-entity recognizer, NER,
instance consolidation function, CONSCORE

Output: A ranked list of instances I_{Ranked}

```
q' ← q.replace("how many", "which")
I ← {}                                   ▷ Inverted dictionary of instances.
S ← {}                                   ▷ Consolidated score for instances.
for d ∈ D do
    c ← SPANPREDICTION(d, q')
    ▷ Get all instances from the span. ◁
    if c.Span ≠ None and c.Conf > θ then
        I_d ← NER(c.Span)
        for i ∈ I_d do
            I[i] ← I[i] ∪ (d, c)
for i ∈ I do
    S[i] ← CONSCORE(I[i], D)
I_Ranked ← SORTDESCENDING(I, key = lambda i : S[i])
return I_Ranked
```

4.5 The CoQuAD Dataset

4.5.1 Dataset construction

Query Collection Existing QA datasets only incidentally contain count queries; we leverage search engine autocomplete suggestions to automatically compile count queries that reflect real user queries [Sullivan(2020)]. We provide the Google search engine with iterative query prefixes of the form *"How many X"*, where $X \in \{a, b, \ldots, z, aa, ab, \ldots, zz, aaa, aab, \ldots, \ldots, zzz\}$. Similar to the candidate generation from patterns used in [Romero et al.(2019)], we generate prefixes where X varies from a single character up to three characters in the English alphabet. We use the SERP API[2] to collect the query autocomplete suggestions for the above query prefixes and keep those with at least one named-entity (to avoid too general queries). Figure 4.3 illustrates autocomplete suggestions for two query prefixes. Queries with at least one named-entity are selected (green underline) and the others with are discarded (red strike-through). Of the $69k$ autocomplete suggestions collected, around $11.9k$ queries have at last one named-entity.

[2] `https://serpapi.com`

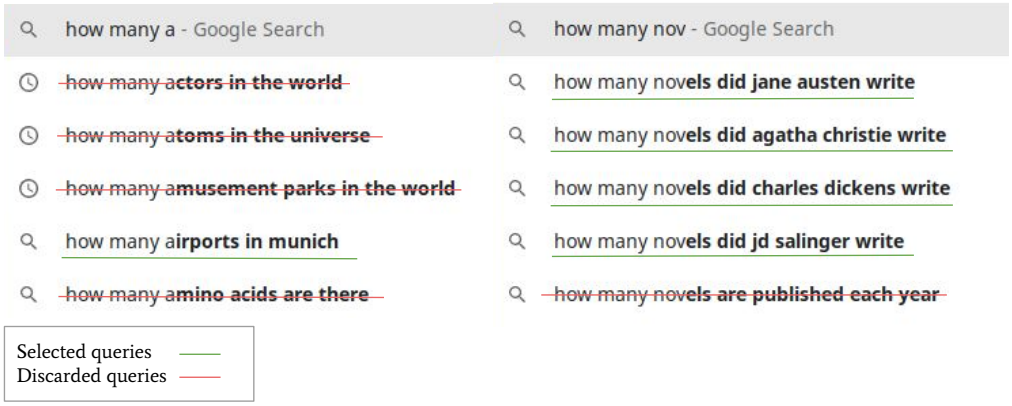


Figure 4.3: Query autocomplete suggestions for the query prefixes: *how many a* and *how many nov*. Queries with no named-entity mentions are discarded (here, red strike-through).

Ground Truth Counts We automatically obtain *count ground truth* by collecting structured answers from the same search engine. Executing each query on Google, we scrape knowledge graph (KG) answers and featured snippets, using an off-the-shelf QA extraction model [Sanh et al.(2019)] to obtain best answers from the latter.

We further clean the dataset by removing queries where no counts could be extracted from the text answers and applying simple heuristics to remove queries dealing with measurements. We achieve this by using the CogComp Quantifier which serves a dual purpose. We use it to normalize the text answers to integer counts and identify empty extractions or non-entity answer types when any word representing measurement is returned as *units* of the identified quantity.

This gives us the ground truth for 5k queries obtained from either KG or from featured snippets. There also exists 4k count queries with no directly available ground truth which we retain in the dataset for evaluation purposes. We manually annotate a sample of 100 queries from those without automated ground truth.

Text Segment Annotation Next, we scrape the top-50 snippets per query from Bing, and obtain *text segment ground truth* by labelling answer spans returned by the CogComp Quantifier [Roy et al.(2015)] as positive when the count lies within ±10% from the ground truth. There are around 800 queries with no positive snippets, which we do not discard, so the system is not forced to generate an answer. In the end we have 5162 count queries with automated ground truth, and an average of 40 annotated text segments per query.

Evaluation Data We use 80% of the count queries with automated ground truth for training and 20% for test and development. We report our evaluations on the hand annotated subset of 322 CoQuAD queries which consists of test data with the automated ground truths (69%), and queries without any direct answers (31%). We manually annotate the queries with categories, counts, and count contexts. We also track the effect of time, availability of instances and count contexts on these queries. 142 queries that would benefit from instance explanations have at least top-5 prominent manually annotated instances for evaluating answer explanations.

Figure 4.4: Example view of the search-engine result pages for two queries in JSON format. **(a)** *how many kubrick movies* with answer from a knowledge graph and **(b)** *how many satellites does earth have* with an answer from a featured snippet.

Quality Assessment We supplement the 69% automated ground truth evaluation data with manually searched ground truths, to obtain insights into the quality of the automated ground truths. We gave ourselves access to the search-engine result page (see Figure 4.4) used by the automatic extractor, and could hence refer to the same snippet and source links, from which the automated ground truth was extracted. We also allowed ourselves to inspect the provided links, in case the snippets alone did not give a conclusive answer. In a few cases, we could only provide estimates, or counts of enumerated instances, if no source gave a definitive count answer.

We evaluate the deviation of the automated answer from the human annotation by calculating the ratio of the smaller of the two annotations to the larger of the two. A ratio of 1 is a perfect match, while lower ratios suggest deviations. We find that in 81% of cases, automated and manual ground truth matched perfectly, and for 84%, the deviation was no more than 10% upward or downward. This is encouraging quality, at a level well above what is typically required in supervised machine learning (e.g., distant supervision for relation extraction often works with training data with accuracy around 50%).

Nonetheless, it is insightful to see where the automated ground truth is incorrect. A majority of errors (16 of 25 queries) stem from counts of subsets, followed by 8 annotations that count a superset. For instance, the automated answer for the number of *bruce lee movies* is ***at least 18***, a lower bound, while the human annotation is 37. The automated answer for *tgi friday restaurants in the United States* is ***900***, extracted incorrectly from a snippet which says ***"There are over 900 T.G.I. Friday's branded restaurants in 60 countries worldwide*** ... ***There are 566 restaurants in the U.S. ..."***.

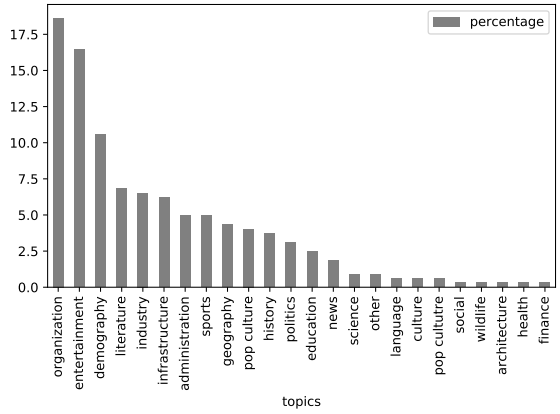


Figure 4.5: Distribution of topics in CoQuAD.

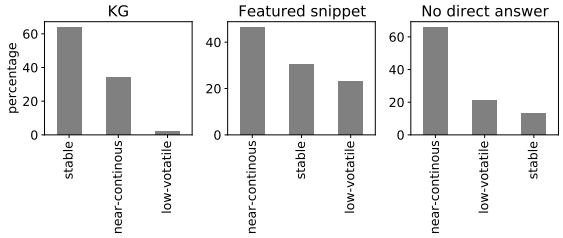


Figure 4.6: Time variance of CoQuAD queries by answer source.

4.5.2 Query Analysis

Dedicated count question answering is a novel topic for question answering, and as such, we first aim to gain insights into the nature of typical count queries. Our analysis is divided into four questions.

1. What ground truths are available for these queries?
2. What are the modes of count answers?
3. What domains do these queries cover, and how topically stable are they?
4. What are their syntactic characteristics?

We look into the evaluation data of 322 CoQuAD queries to answer these questions unless specified otherwise.

Nature of Ground Truth When we automate the ground-truth extraction, we realize that there exists structure to the results provided by the search engines as illustrated in Figure 4.7. Answers to a small minority of roughly 2% of the queries come from the internal KG. These *KG-answerable* queries have well-structured outputs. In the case of count queries, the answers returned are counts and the path to the KG answer is also displayed to the user.

The majority of the ground truth labels, (54% of all CoQuAD queries), are extracted from the top snippet. These snippets rank at the top of the search results, identified as featured snippets and

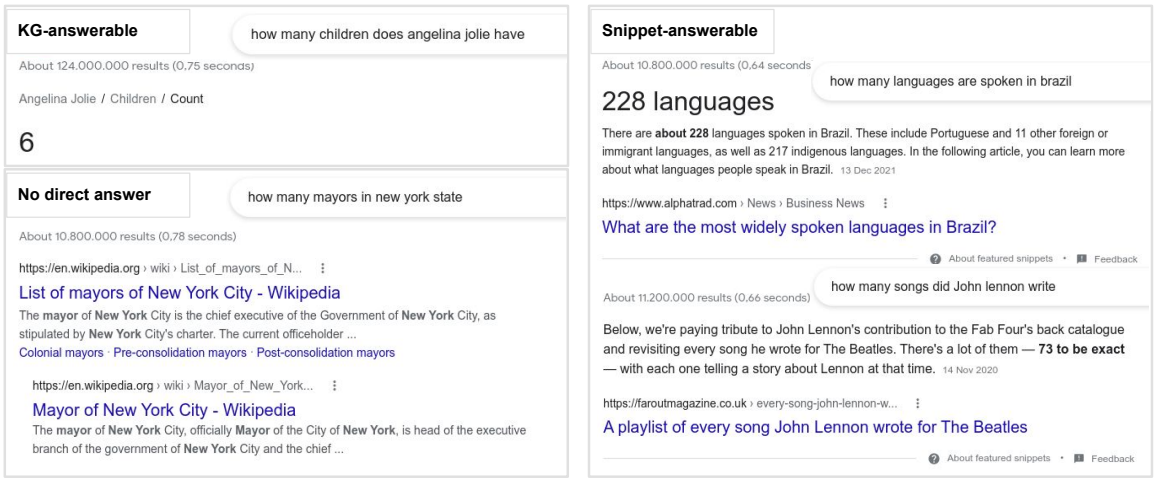


Figure 4.7: Nature of ground truths extracted. **KG-answerable** queries show the path (entity and relation) and aggregate used (Count). **Snippet-answerable** queries have a featured snippet with a highlighted answer displayed at the top of the snippet (228 languages) or within the snippet which then can be extracted by any off-the-shelf extractive QA models. **No direct answer** type of queries do not have any automated ground truth as the results returned only ranked.

are accompanied by an answer span, highlighted within the text or as a heading of the snippet. We refer to such queries as *snippet-answerable*, and provide two examples in Figure 4.7.

The queries which yield no automated ground truths (44% of all CoQuAD queries) come under the *No direct answer* category. As illustrated in Figure 4.7, these queries only return ranked page snippets, due to the lack of any KG answer or featured snippet.

Of the automated ground truth labels, the vast majority come from featured Google snippets, while only 2% come directly from the Google KG. These labels are thus complemented by a more balanced manual annotation, where we manually labelled 50 questions that were KG answerable, 172 that were snippet-answerable, and 100 without any automated ground truth.

Answer Modes We have identified three modes how QA systems can answer count queries — via counts, CNPs and instances. We analyze the occurrence of these modes in the automated ground truths. If we analyze the paths returned by KG answerable queries and the featured snippets, we find that the KG-answerable queries are usually simply related and have no occurrences of semantic qualifiers (equivalent to CNPs) in CoQuAD. Featured snippets on the other hand contain CNPs in 61% of the cases. Instances come up in 90% of the KG-answerable count queries and in only 20% of the snippet-answerable count queries.

We then proceed by annotating the CoQuAD queries with binary variables indicating whether semantic qualifiers and instance explanations are necessary. For instance, the query *how many novels did jane austen complete* would benefit from instances and CNPs which differentiate her finished and unfinished works or at least hint at the fact.

We found that, indeed, helpful semantic qualifiers for KG-answerable queries are necessary in around 8% of the queries. In snippet-answerable queries and queries with no direct answers, semantic qualifiers are desirable in more than 81.3% of the cases. As far as instance explanations are concerned,

they are desirable in all KG-answerable queries and in 80.8% of the snippet-answerable queries. We already see a gap between the desired explanatory evidence and what is available when no consolidation is performed. In Section 4.7.4, we further report on these answer modes in light of the predictions made by CoQEx to see whether this gap can be reduced.

Domain and Stability We assigned high-level topics to the queries. We went through each of the 322 queries, introducing a new topic label if none of the previous ones make a good match. We found that queries in CoQuAD cover a range of topics, notably organizations (18.6%), entertainment (16.4%), demography (10.5%), literature (6.8%), industry and infrastructure (12.7%) (see Figure 4.5). A second important dimension concerns their temporal stability. Queries whose result continuously changes are naturally much harder to deal with, especially if fluctuations are big. We find that 30% of query results are fully stable (a company's founders, casts in produced movies), 20% are low-volatile (lakes in a region, band members of an established but active band), 50% are near-continuous (employment numbers, COVID cases). In Figure 4.6 we break it down by answer source (KG, featured snippet, and no direct answers), and we see that the majority of the KG answerable queries is stable (64%) and near-continuous for the rest. Featured snippet can be used to answer time-variant queries, though near-continuous queries are a majority (46.5%). Near-continuous queries form an overwhelming majority (66%) where search-engines do not provide any direct answers. Only about 13% of queries with no direct answers are stable, which means that search-engines can handle such queries with little to no-variance.

Syntactic Properties We identify different query components, namely the named entities, the relation, the type of the entities being counted (conventionally referred to as the type of the answer entity), and remaining context as a bag of words. We perform basic natural language processing[3] to obtain the query components as follows.

- *Named entity*: tokens were extracted if NER returned a label or if the tokens were tagged `proper noun` by the POS tagger.
- *Relation*: tokens were identified when a token had a POS tag = `verb` and the token was the root of the dependency parse tree.
- *Answer type*: tokens identify the type of the counted entities. The first `noun` token from the dependency parse tree with any of it's preceding `adjective` tokens formed the answer type. This is used by CoQEx to determine the type compatibility of the instance candidates (discussed in the methodology Section 4.4.3 on the *QA + Type Compatibility* consolidation strategy). For example, if the query counts restaurants, cities, or cartoon characters, we use text entailment to check whether candidate named entities are of the respective type.
- *Context*: tokens were all remaining keyword tokens, *i.e.,* excluding conjunctions, determiners, auxiliary verbs, pronouns, punctuation.

The average query length is 6.40 words, with an average of 1.08 named entities per query implying that most queries count entities in a simple relation to one named entity. We found that 95% of the 322 queries returned non-empty answer types, with more than 200 unique phrases, for instance,

[3]We use SpaCy's `en_core_web_sm` model.

sibling, movie, employee, nfl stadium, real estate agent, czech player. The queries spanned over 49 different relations.
Our research data is made publicly available[4].

4.6 Experiments

While we can use regular IR metrics of precision and recall for evaluating answer explanations (Mean Average Precision@k, Recall@k, Hit@k and MRR) and accuracy of classification for evaluating count context categories, we need a new metric for counts. This is because counts come with a natural order and distance function (e.g., 507 may be a good answer when the ground truth is 503, but not when it is 234), for which exact string match or embedding distance is not a suitable metric.

4.6.1 Evaluation Metrics

We report the following evaluation metrics for measuring count inference.

1. *Relaxed Precision* (RP) is the fraction of answered queries where the prediction lies within $\pm 10\%$ of the ground truth and is reported as a percentage.

2. *Coverage* (Cov) measures the fraction of queries that a system returns an answer for and is reported as a percentage.

3. *Relaxed Precision-Coverage trade-off* (P/C) is the harmonic mean of the relaxed precision and coverage values, reported as a percentage.

4. *Proximity* $\in [0, 1]$, which is the ratio of the minimum of the predicted and the gold answer to the maximum of the two, averaged over all queries.

Since we deal with non-canonicalized surface forms of instances, we maintained a list of aliases, obtained from Wikidata, for the annotated prominent instances. An instance is relevant if its length-normalized Levenshtein distance [Navarro(2001)] from any of the aliases is less than 0.1. We evaluate answer explanations on the following metrics:

1. *Mean Average Precision* (MAP) is the fraction of retrieved entities that are relevant, averaged over the queries.

2. *Average Recall* (AR) is the fraction of ground truth entities retrieved, averaged over the queries.

3. *Hit@k* is the percentage of queries with at least 1 relevant answer in the top-k.

4. *Mean Reciprocal Rank* (MRR) is the inverse of the rank of the first relevant result, averaged across all queries.

The CNPs are evaluated based on the accuracy of the classified labels of *Synonyms*, *Subgroups* and *Incomparables*, measured as the ratio of correct predictions to the total predictions in each class.

[4] `https://github.com/ghoshs/CoQEx`

4.6.2 Baselines

We compare our proposed system with two complementary paradigms.

1. Knowledge-base question answering: QAnswer [Diefenbach et al.(2019)].
2. Commercial search engine QA: Google Search Direct Answers (GoogleSDA). In other words, we scrape the structured results from the result page of the Google Search engine.

For fairness to QAnswer, which specifically deals with count queries by aggregating on top of the SPARQL query, we queried the system[5] twice — the original count query (for the count answer) and a modified variant as in Section 4.4.3, i.e., replacing *"how many"* with *"which"*. We then post-processed the results to extract count and instances. For evaluating instances by GoogleSDA, we post-processed knowledge graph and featured snippet of the search engine result page, keeping items from list-like structures as instances ranked in their order of appearance.

4.6.3 Datasets

In order to test the generalizability of CoQEx we present the results on count queries from multiple datasets in addition to CoQuAD:

1. 100 count queries from an existing dataset LCQuAD [Dubey et al.(2019)],
2. A manually curated dataset of 100 challenging count queries called *Stresstest*, and
3. 84 count queries found in the Natural Questions [Kwiatkowski et al.(2019)] dataset. These queries are similar in nature to our CoQuAD queries (sample of real user queries from Google), but not subject to our own scraping and filtering, and thus provide a corroboration signal for our larger CoQuAD dataset.

4.6.4 Implementation Details

The candidate generation steps for answer inference and explanation uses two instances of SpanBERT. The model for answer inference is trained on CoQuAD for 2 epochs, at a learning rate of $3e^{-5}$ using an Adam optimizer. An input datapoint for training consists of a query, a text segment and a text span containing the count answer (empty if no answer). We train over 3 seeds and report the average score on the test data. For getting the instances from the answer spans, we use the pre-trained SpaCy NER model[6]. The model for answer explanation is trained on SQuAD 2.0.

4.7 Analysis

4.7.1 Extrinsic Evaluation

Baseline Comparison on Answer Inference Table 4.1 shows the answer inference performance of CoQEx against the baselines on different datasets. The RP-Coverage trade-off metric highlights the advantages provided by CoQEx. In both CoQuAD and Natural Questions datasets GoogleSDA

[5] QAnswer API at `https://qanswer-core1.univ-st-etienne.fr/swagger-ui.html`. Last accessed in Nov 2022.
[6] `https://spacy.io` on the `en_core_web_sm` model.

Table 4.1: Comparing baselines on answer inference results (in percentages), where **RP**= relaxed precision, **Cov**=coverage and **P/C**=relaxed precision-coverage trade-off.

System	CoQuAD			LCQuAD$_{count}$			Stresstest			NaturalQuestions		
	RP	**Cov**	**P/C**	**RP**	**Cov**	**P/C**	**RP**	**Cov**	**P/C**	**RP**	**Cov**	**P/C**
QAnswer [Diefenbach et al.(2019)]	6.6	96.2	12.4	45.0	96.1	**61.3**	9.0	100	16.5	12.5	98.8	22.1
GoogleSDA	93.2	18.3	30.6	44.4	8.6	14.4	79.3	29.0	42.4	94.4	22.6	36.4
CoQEx	37.7	84.7	**52.2**	13.6	49.3	21.3	43.6	91.6	**59.1**	43.0	91.6	**58.5**

Table 4.2: Comparing answer inference results (in percentages) by GT source of CoQuAD queries: KG-answerable, snippet-answerable and no direct answers (NDA). The number of queries in each type in mentioned in brackets in the column header.

System	**KG** (50)			**Snippet** (172)			**NDA** (100)		
	RP	**Cov**	**P/C**	**RP**	**Cov**	**P/C**	**RP**	**Cov**	**P/C**
QAnswer [Diefenbach et al.(2019)]	12.2	98.0	21.7	4.1	97.0	8.0	8.2	94.0	15.1
GoogleSDA	100	100	**100**	75.0	2.3	4.5	40.0	5.0	8.8
CoQEx	23.1	98.0	37.4	45.3	85.8	**59.3**	31.9	76.3	**45.0**

has an RP above 90%, albeit for very low coverage, whereas QAnswer has a high coverage, more than 96% in all datasets with poor RP. CoQEx not only provides a high coverage, but also a decent RP, with the improved version increasing RP by 10% on CoQuAD. Except on the LCQuAD dataset, CoQEx provides a better trade-off than the baselines.

On the LCQuAD dataset, designed specifically for KG queries, it can be argued that as the LCQuAD queries are created from question templates which in turn are generated from SPARQL templates, the semantic gap between the natural language query and its KG counterpart is much lower. This aids QAnswer and hinders natural language document retrievers used in the other baselines. The fact that CoQEx has the lowest coverage in LCQuAD among all datasets also backs this hypothesis.

The manually created Stresstest dataset shows the potential of CoQEx in terms of coverage and RP, even though RP of GoogleSDA is higher. Here, results indicate that reliance on structured KBs (QAnswer) is not sufficient for general queries, and robust consolidation from crisper text segments is necessary.

Effect of Query Types on Answer Inference In Table 4.2, we analyze how QA systems perform on the answer inference when a query is KG-answerable, snippet-answerable and when a query is not directly answerable. The difficulty in answering the queries increases with each type.

We observe that the baselines achieve their best performance on KG-answerable queries. While QAnswer has a high coverage, the RP metric is quite low, even for KG-answerable queries. GoogleSDA has by design 100% precision in KG-answerable queries. While its coverage goes down drastically with increasing difficulty levels of the queries, barely above 5%, the RP remains respectable. CoQEx maintains a decent balance between coverage and RP values in all three scenarios.

Since, CoQEx considers only text, it loses to GoogleSDA in KG-answerable queries by a margin, but is still better than QAnswer. In snippet-answerable queries and queries with no direct answers, CoQEx provides a much better P/C trade-off than the baselines.

Table 4.3: MAP@k, AR@k (R@k), Hit10 and MRR of CoQEx and baselines for the answer explanations.

System	MAP@1	MAP@5	MAP@10	AR@1	AR@5	AR@10	Hit@10	MRR
CoQuAD (142 queries)								
QAnswer [Diefenbach et al.(2019)]	8.5	9.3	9.6	2.9	6.5	8.4	19.7	0.118
GoogleSDA	**14.8**	**12.8**	**10.6**	**4.8**	**13.7**	**14.3**	23.2	0.185
CoQEx	12.0	11.7	11.0	2.3	9.3	12.7	**37.3**	**0.200**
Natural Questions (84 queries)								
QAnswer [Diefenbach et al.(2019)]	14.3	15.7	16.3	5.0	11.1	14.3	33.3	0.199
GoogleSDA	**25.0**	**21.7**	**17.9**	**8.0**	**23.2**	**24.2**	**39.3**	**0.313**
CoQEx	9.5	8.0	7.2	3.6	8.0	11.2	25.0	0.143

Evaluating Answer Contexualizations For evaluating count contextualizations, we cannot directly compare CNPs acquired through CoQEx with the other baselines, especially in the KB-QA setting since they return answers with little to no context.

Semantic qualifiers are still common in GoogleSDA featured snippets, coming up in 61% of queries. While semantic qualifiers can be expressed in KG answers,

$$\textbf{\textit{volcanic islands in Hawaii}} \Rightarrow \textbf{\textit{islands}} \rightarrow \textbf{\textit{Hawaii}} \rightarrow \textbf{\textit{volcanoes}},$$

this rarely shows up in GoogleSDA for two reasons, i) KG answers are provided for short (single entity) and simple queries (relations with no semantic qualifiers) and ii) KG answers are provided for very popular queried entity and qualifier.

Unlike the hybrid mode in GoogleSDA which returns results from both a KB and texts, QAnswer is fully KB-based, and SPARQL query understanding is challenging. If we only consider the top-1 SPARQL query, we get a detailed interpretation of the natural language query, but the result is homogeneous. For example, in the query *how many territories does canada have*, ***territories*** is interpreted as ***located in the administrative territorial entity*** and in the query *how many poems did emily dickinson write*, ***write*** is interpreted as ***author***.

The relations and answer types used in the top-k SPARQL queries can provide insights into existing contextualization in KB-QA as follows. If a subsequent query returns the same set of answers and has similar interpretation, then we have *Synonyms* and when a subsequent query returns an overlapping set of answers such that one query returns a subset of the other, we have *Subgroups*.

When we look into the top-2 SPARQL queries, we find that only about 3% of the queries provide equivalent answers. These, however, cannot be considered *Synonyms* by definition, since they are always equivalent query reformulations. Typical relations are ***spouse*** and ***sibling*** such that the reformulations $\langle ?x, spouse, Entity\rangle$ and $\langle Entity, spouse, ?x\rangle$ are symmetric, and the answer sets are identical but no additional semantic context is obtained.

In only 5% of the queries, where one SPARQL query returns the subset of the other, we find distinct relations indicating subgroups. For example, ***albums*** in the query, *how many elton john albums are there* is interpreted as ***album*** in the first query and ***studio album*** in the second and ***mvps*** in *how many mvps does kobe bryant have* is interpreted as ***NBA Most Valuable Player Award*** in the first query and ***most valuable player award*** in the second.

Table 4.4: MAP@k, AR@k (R@k), Hit@10 and MRR for the answer explanations of CoQEx and baselines on CoQuAD queries by their GT source.

System	MAP@1	MAP@5	MAP@10	AR@1	AR@5	AR@10	Hit@10	MRR
KG (50 queries)								
QAnswer [Diefenbach et al.(2019)]	20.0	21.3	22.0	6.1	14.2	18.5	38.0	0.250
GoogleSDA	**42.0**	**36.4**	**30.0**	**13.5**	**38.9**	**40.7**	**66.0**	**0.526**
CoQEx	14.0	13.7	12.9	3.8	13.0	18.4	42.0	0.233
Snippet (92 queries)								
QAnswer [Diefenbach et al.(2019)]	2.2	2.8	2.9	1.2	2.4	3.0	9.8	0.046
GoogleSDA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
CoQEx	**10.9**	**10.6**	**10.0**	**1.5**	**7.3**	**9.7**	**34.8**	**0.182**

Baseline Comparison on Answer Explanation The results on instance-annotated CoQuAD and Natural Questions dataset are in Table 4.3. We see that GoogleSDA is the best across datasets in terms of MAP and AR. CoQEx comes close in the CoQuAD dataset but performs worse than both baselines in the Natural Questions dataset. Instance explanations when readily available in KGs, can be extracted with a single query. Texts prove useful when KG is incomplete or the SPARQL translation does not capture the user intent. We test this hypothesis in the next subsection.

We identify some challenges which need to be tackled to improve instance explanations from text. The low precision scores of CoQEx can be attributed to:

1. noise due to non-entity terms recognized as entities,
2. alternate human-readable surface forms like the ***European Union*** as the group with which ***South Korea*** has a foreign trade agreement with instead of the specific group name ***European Free Trade Association***,
3. entities satisfying a more general criterion, for instance returning other airports from Vietnam when asked for airports in Ho Chi Minh City,
4. local or generalized surface forms, for instance ***Himalayan rivers*** referring to the group of rivers in India originating from the Himalayan mountain range instead of specifically naming the rivers.

These errors are specific to texts and are difficult to overcome without human annotation.

Answer Explanation by Query Type We analyze the system performances on answer explanation by the query answerability: KG-answerable, snippet-answerable in Table 4.4. Our hypothesis is that the baselines perform very well on answer explanations when the queries are KG-answerable. The performance values support this since we observe that GoogleSDA provides the best precision, recall and MRR sores, followed by QAnswer for the KG-answerable queries. Given that CoQEx only uses text information, it still finds relevant instances, achieving a 14% MAP at rank 1.

In the case of snippet-answerable queries, the dependence of the baselines on KGs becomes clear. CoQEx performs the best, followed by QAnswer and GoogleSDA. It should be noted that GoogleSDA might return list pages in the search result, such that if we were to scrape the contents of the list page, we would likely find correct instances. However, we limit ourselves to instances found on the result page itself, either as an answer from its KG or in the form of direct answers (featured snippets).

Table 4.5: User preference for different explanation modes (in percentages).

Explanation Type	Bare Count	Explanation	Both	None
CNPs	13.3	50.0	33.3	3.4
Instances	3.3	73.3	23.4	0.0
Snippet	0.0	80.0	20.0	0.0
All	10.0	63.3	23.4	3.3

Table 4.6: Extrinsic user study on annotator precision (in percentage).

Class	Only Count	+Instances	+CNPs	+Snippet	All
Correct	73	63	78	75	**88**
Incorrect	28	45	40	**53**	45
Both	55	56	63	66	**71**

4.7.2 Intrinsic Analysis

We evaluate the CoQEx components to determine the best configurations for answer inference, consolidation and explanation.

Span Prediction Model for Answer Inference We test the candidate generator for count spans on SpanBERT finetuned on i) CoQuAD and, ii) the popular general QA dataset SQuAD [Rajpurkar et al.(2016)] on different span selection thresholds in Figure 4.8. Span selection works such that counts coming from spans with a model confidence above the threshold is used for aggregation. As is expected, the precision goes up while the coverage decreases as the thresholds are set higher, since the model becomes more conservative on high confidence predictions. A threshold of 0.5 gives the best precision-coverage trade-off.

Fine-tuning on SQuAD gives higher precision scores at thresholds greater than 0.4. However, this difference, which goes up to 3% max, is a trade-off to the higher coverage of CoQuAD gives, between 5%-8% higher, resulting in overall more correctly answered queries. Here, we average the metrics over all consolidation strategies (*Most Frequent*. *Median* and *Weighted Median*) and compare the consolidation schemes next.

Best Consolidation for Answer Inference When selecting a consolidation strategy, we compare the *Relaxed Precision* and *Proximity* metrics, since coverage is same for all strategies (see Table 4.7). The *confident* strategy, which in essence performs no consolidation, has the lowest *Proximity* beating only the *median* consolidation strategy in *Relaxed precision* by 1.6%.

The *weighted median* is the winning strategy, indicating that using model confidence as weights boosts performance. The naive *frequent* strategy comes very close to the *weighted median* scheme, both in terms of RP, where it is equal, and *Proximity* (behind by 0.09). Thus, for queries backed by less variant data, *frequent* is good enough, but to have an edge in more variant data *weighted median* is the way to go.

Accuracy of Answer Contextualization The accuracy of the CNP categories is directly dependent on the quality of the prediction. Since in this experiment we only want to test the accuracy

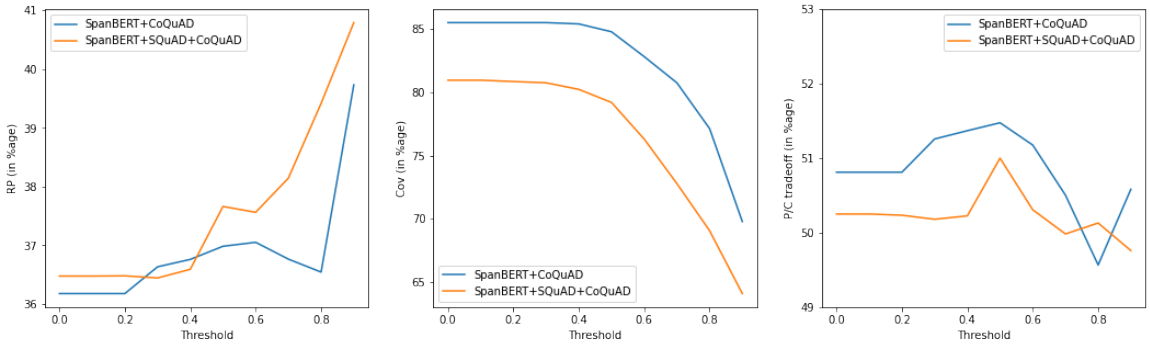


Figure 4.8: Performance of fine-tuned models on answer inference metrics, (from left to right) Relaxed Precision (RP), Coverage (Cov) and Relaxed Precision-Coverage trade-off (P/C) across different span selection thresholds.

Table 4.7: Intrinsic evaluation of the Answer Inference on consolidation alternatives. The model is SpanBERT+CoQuAD with span prediction threshold=0.5

Consolidation	**Relaxed Precision (%)**	**Coverage (%)**	**Proximity**
Median	35.4	84.7	0.611
Most Confident	37.0	84.7	0.600
Most Frequent	37.7	84.7	0.611
Weighted Median	37.7	84.7	0.620

of the CNP category classifier, we restrict ourselves to CNPs from correct predictions (RP=1). We assess the classification accuracy of CNPs for a manually labelled sample of 601 CNPs for 106 queries.

A strict synonym threshold of $\alpha = 0\%$ (CNPs equal to predicted count with cosine similarity > 0) ensures a high accuracy of 82.1% for *Synonyms* and only decreases with increasing α down to 69.1% for $\alpha = 100\%$. The accuracy of *Subgroups* is initially low (34.4%). It increases with higher levels of α, peaks at $\alpha = 60\%$ and then decreases. This happens because, as α increases, more incorrect CNPs are classified as *Synonyms*.

At lower values of α CNPs which are synonyms of the predicted count but a bit further away get classified as *Subgroups*. As α increases, these CNPs with counts a bit further away from the predicted count are correctly classified as *Synonyms* and the accuracy of the *Subgroups* increases. After α crosses 60%, CNPs which are subgroups of the predicted count are mis-classified as *Synonyms*, thereby decreasing the accuracy of the *Synonyms* and the *Subgroups*. The CNPs with very low counts are still mis-classified as *Subgroups* if they have a high semantic similarity to the representative CNP.

The number of *Incomparable* CNPs decreases with increasing α, which gives a higher accuracy but at the cost of incorrect *Synonyms* and *Subgroups*. A weighted optimum is reached at $\alpha = 30\%$, where the accuracy of the *Synonyms* does not degrade much (79.6%), and the accuracy of *Subgroups* and *Incomparables* is both above 60% (61.9% and 71.4% respectively).

Effect of Model Confidence on Answer Explanation We see the effect of thresholding the answer spans by the confidence of the span prediction model on MAP and average recall in Figure 4.9. We observe that, while the recall goes down in general with increasing model confidence, except for recall at rank 1, which stays more or less constant for thresholds between 0.1 and 0.8, the precision

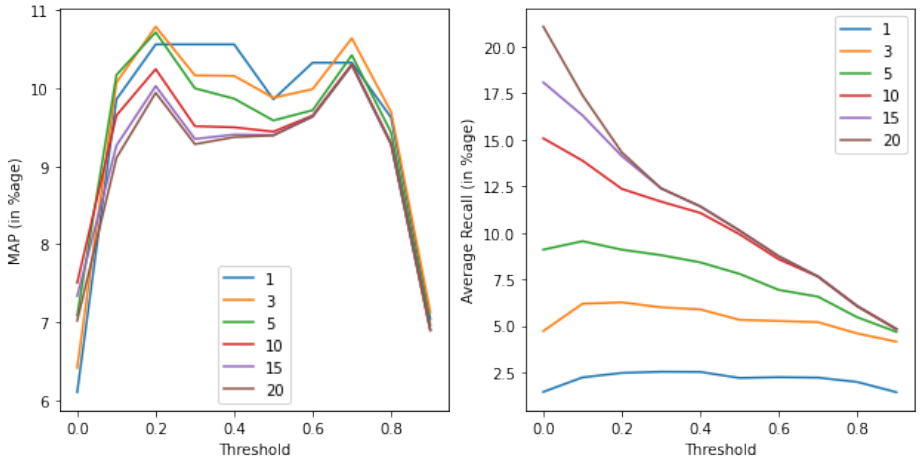


Figure 4.9: MAP and Average recall across threshold values and ranks. The values are averaged over consolidation alternatives.

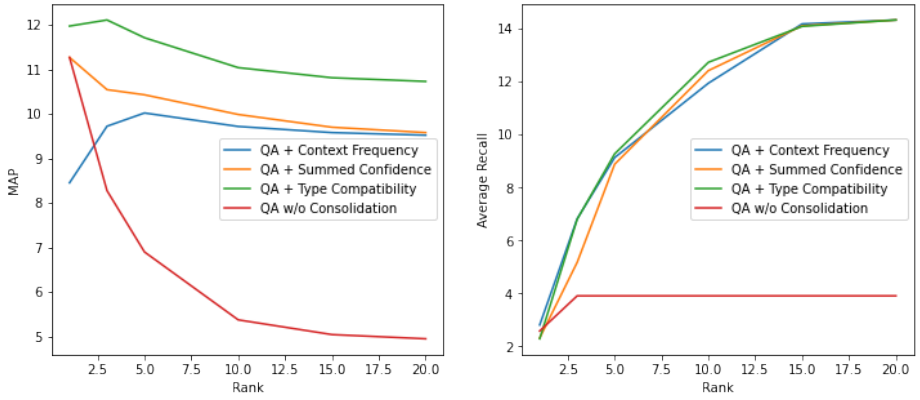


Figure 4.10: MAP and Average Recall of different consolidation strategies at ranks 1, 5 and 10 when span selection threshold=0.2.

drops sharply when model is both less and more confident and has two peaks at 0.2 and 0.7.

We can argue that the gradual drop in recall is because the model predicts less number of high confidence spans. The precision has a sharp increase initially when increasing the model confidence threshold from 0 to 0.2, because at threshold=0, we consider all predictions and this introduces a lot of noise. Whereas, the drop in precision at thresholds 0.8 and higher is probably due to the model being penalized for being too conservative and making sparse or no predictions. Choosing a model confidence threshold of 0.2, creates a balance between precision and recall values.

Best Consolidation for Answer Explanations The MAP and AR scores of different consolidation strategies are shown in Figure 4.10. Without consolidation, MAP and average recall are comparable only at rank 1, after which the gap between strategies with and without consolidation increases sharply. All consolidation strategies perform similarly in average recall and the discriminating factor is in the MAP by rank. *QA + Type Compatibility* is the best overall followed by *QA*

+ Summed Confidence. The naive *QA + context frequency* performs the worst at rank 1 implying that the most frequent named-entities may not be the correct explanations.

Thus we can say that consolidation is a determining factor in increasing performance, with different starting points at rank 1, but converging at higher ranks. *QA + Type Compatibility* is the most stable across ranks, followed by other consolidation strategies. With no consolidation, MAP decreases rapidly across ranks, unlike consolidation methods where MAP is stable across ranks. The recall without any consolidation very quickly stagnates at quite low values (3.3%).

4.7.3 User Studies

To further verify the user perception of our enhanced answers, and their extrinsic utility, we performed three user studies.

User Study 1: Intrinsic Answer Assessment We asked 120 MTurk users for pairwise preferences between answer pages that reported bare counts, and counts enhanced by either of the explanation types. The preference of different explanation types are shown in Table 4.5. 50% of participants preferred interfaces with CNPs, 80% with a snippet, 73% with instances, 63% preferred an interface with all three enabled. While snippets are already in use in search engines, the results indicate that CNPs and instances are considered valuable, too.

User Study 2: Extrinsic Utility for Assessing System Answer Correctness We also validated the merit of explanations extrinsically. We took 5 queries with correct count results, 5 with incorrect results, and presented the system output under the 5 explanation settings to 500 users. The users' task was to judge the count as correct or not based on the explanations present. The measured precision scores are in Table 4.6. All explanations had a positive effect on overall annotator precision, especially for incorrect counts.

User Study 3: User Satisfaction In this study we asked 100 MTurk users to report their satisfaction with CoQEx's output, when presented with an answer screenshot containing the answer inference, along with the different explanations. We also showed the first few annotated snippets as provenance for the system explanations. Users were then given a 5-point Likert scale to express their satisfaction, along with a text field in which they should report a justification. The evaluation was performed on the same 10 queries as before.

On the five-level Likert scale of satisfaction, 37% of the users were fully satisfied, 55% were satisfied, 2% were neither satisfied or dissatisfied, 2% were dissatisfied and 4% of the user were fully dissatisfied. Summing up, 92% of users were at least satisfied. Concerning qualitative justifications, satisfied users mostly provided short praise, some noted specific count contexts which they found useful. For instance, one user wrote that ***the eleven studio albums is good*** for the query *how many albums does eminem have*. Some of the phrases used by fully satisfied users were: ***clearly listed***, ***100% accurate***, ***thorough***, ***detailed***, ***easy to read***, ***lots of information***, ***makes sense***, ***well understood***. Two of the users who felt fully dissatisfied had technical issues with the display and one expected more recall. The users who responded as being dissatisfied commented that the answer inference seemed incorrect in light of the provided explanations. This corroborates the results from our second user study, where we found that explanations helped users in determining

Table 4.8: Number of queries with at least 1 contributing snippet under different settings and number of snippets per query satisfying the setting.

Setting	**#Queries**	**#snippets/query**
Counts Candidates	279	5.8
Correct Counts	106	6.0
Relevant CNPs	106	6.0
Instance Candidates	307	15.7
Relevant Instances	39	3.9
Counts & Instance Candidates	151	2.8
Correct Counts + Relevant Instances	3	1

the correctness of the system's answer. The users who showed neither satisfaction nor dissatisfaction explained that, despite onboarding, they did not understand the task well enough.

4.7.4 Discussion

Distribution of CNPs and Instances in Snippets In Section 4.5.2 we introduced the different answer modes in count queries. An open question is how often these are actually present in text sources. We distinguish four notable cases CoQEx encounters in the snippets:

1. Instance candidates — 95.3% of the 322 manually annotated CoQuAD queries have at least one snippet with instance candidates with an average of 15.7 snippets per query containing instance candidates.

2. Count candidates — 86.6% of the 322 queries have snippets with count spans with an average of 5.8 snippets per query containing count candidates.

3. Both instance and count candidates — 46.8% of the queries have snippets containing both count and instance candidates with an average of 2.8 snippets per query.

4. Count candidates with semantic qualifiers (e.g., *7 official languages and 30 regional languages*) — 86.6% of the queries have snippets with CNPs, with an average of 5.8 snippets per query containing CNPs.

Now that we have established that snippets are a good source of candidates, we also report on the number of queries and number of snippets per query, which contain counts that lead to correct predictions and relevant instances. These are summarized in Table 4.8. Using CoQEx we are able to identify relevant instances for 39 of our queries spread across an average of 3.9 snippets per query. Relevant CNPs and counts could be identified in more than 30% of the count queries with counts spread across an average of 6 snippets per query.

Coexistence of Counts and Instances In CoQEx the tasks concerning counts and instances are separately tackled and a natural question arises as to how counts and instances are spread across the snippets and whether they frequently coexist in the same document, like (***He wrote 73 songs, for example, Let it Be, ...***). Frequent coexistence would be very beneficial for the approach, since it would allow focusing on identifying snippets that solve both sub-problems at once.

We find that around 46.8% (151) queries contain at least one snippet with both counts and instance candidates. However, the number of snippets containing co-occurring counts and instances is less than 3 per query and, only 3 of the 151 queries have correct counts and relevant instances (see table 4.8). Thus indicating that relevant information is spread across contexts, making our task of inferring answer and explanatory evidence from multiple sources a significant contribution.

Table 4.9: Example outputs of CoQEx with confidence scores of CNPs and aggregated scores of instances in subscripts.

No.	Query	Inference	CNPs	Top-5 Instances
1.	how many songs did john lennon write for the beatles	73	**CNP*rep*:** 73 songs$_{(0.92)}$ **Synonyms:** 61 songs$_{(0.77)}$ **Subgroups:** 22 songs$_{(0.67)}$ **Incomparables:** 189 songs$_{(0.82)}$, 229 original songs$_{(0.55)}$, 229 songs$_{(0.5)}$	**x** John Lennon's$_{(0.71)}$ **x** Beatles$_{(0.55)}$ ✓Maggie Mae$_{(0.54)}$
2.	how many main islands in hawaii	8	**CNP*rep*:** eight principal islands$_{(0.96)}$ **Synonyms:** eight main islands$_{(0.91)}$, six major islands$_{(0.91)}$, 8 main islands$_{(0.9)}$, 8 largest$_{(0.83)}$	✓the Big Island$_{(0)}$
3.	how many languages are spoken in indonesia	709	**CNP*rep*:** 709 living languages$_{(0.97)}$ **Synonyms:** 653 languages$_{(0.98)}$, estimated 700 languages$_{(0.91)}$, 700 living languages$_{(0.96)}$, 725 languages$_{(0.78)}$, 800 languages$_{(0.61)}$ **Subgroups:** 300 different native languages$_{(0.94)}$	✓Malay-Indonesian$_{(0.79)}$ ✓Indonesian language$_{(0.77)}$ ✓Bahasa$_{(0.7)}$ ✓Indonesian$_{(0.35)}$
4.	how many osmond brothers are still alive	9	**CNP*rep*:** nine Osmond siblings$_{(0.89)}$ **Synonyms:** nine siblings$_{(0.87)}$, nine children$_{(0.59)}$, 7 brothers$_{(0.71)}$, 9 of the Osmond siblings$_{(0.82)}$	✓Alan Osmond$_{(0.89)}$ ✓Wayne Osmond's$_{(0.72)}$ ✓Merrill Osmond$_{(0.64)}$ ✓Donny Osmond's$_{(0.64)}$
5.	how many wives did king solomon have	700	**CNP*rep*:** 700 wives$_{(0.97)}$ **Synonyms:** 500 wives$_{(0.54)}$, seven hundred wives$_{(0.9)}$ **Subgroups:** three of his wives$_{(0.96)}$, three children$_{(0.86)}$ **Incomparables:** 700 hundred wives$_{(0.85)}$, 1,000$_{(0.98)}$	✓Moti Maris$_{(0.84)}$ **x** Memphis$_{(0.62)}$
6.	how many inactive volcanoes are in hawaii	5	**CNP*rep*:** five active volcanoes$_{(0.87)}$ **Synonyms:** four active volcanoes$_{(0.85)}$, five separate volcanoes$_{(0.73)}$ **Incomparables:** 169 potentially active volcanoes$_{(0.8)}$	✓Diamond Head$_{(0.95)}$ ✓Mauna Kea$_{(0.41)}$ ✓Haleakala$_{(0)}$

Case Study We pick up some challenging and interesting count queries which bring out the complexities of the problem and also showcase the capabilities of our system. The queries are collected in Table 4.9.

The first query is our running example *how many songs did john lennon write for the beatles* where the information need is for songs by Lennon with an additional condition that these are for the Beatles band. The complexity of this question comes through the snippets which indicate that other band members (George Harrison, and Paul Mc-Cartney) also wrote songs and that there are songs co-written by the band members.

CoQEx returns 73 as the answer inference and count contextualizations i) ***22 songs*** which belongs to *Subgroups* category, since it comes from a snippet talking about *lead guitarist george harrison wrote 22 songs*, ii) ***61 songs*** classified as a synonym comes from a competing source which says that *"Lennon wrote 61 songs credited to Lennon-McCartney all by himself"* iii) ***229 songs***, classified as *Incomparable*, comes from a snippet about all the songs The Beatles as a band has written.

Finding instances are much harder, with composition credits varying vastly across songs and albums. The top-5 instances returned are false positives (names of the band members). CoQEx identifies one joint composition ***Maggie Mae*** and one album ***A Hard Day's Night*** whose title track and the majority of the album songs are written by Lennon, but is not very confident, scoring it very low (0.01).

The second query, *how many main islands in hawaii*, is looking specifically for the ***main*** Hawaiian islands. The CNPs returned by CoQEx, express the different interpretations of *main* as being more popular (***major***) or being ordered in terms of size (***largest***). CoQEx is also able to corroborate this with correct instances.

The third query, *how many languages are spoken in indonesia* seems relatively simple with a popular entity ***Indonesia***, well-defined predicate ***spoken in*** and an answer type ***language***, but is a great example of high variance answers. The presence of the modifier ***estimated*** and multiple close numbers (653, 700, 709) in the CNPs highlight the fact that it may not be possible to have one true answer.

The fourth query, *how many osmund brothers are still alive* is a query from the CoQuAD dataset, where instances in the snippets are more prevalent than counts. The CNP ***9 siblings*** counts all brothers (8) and a sister, and ***7 brothers*** CNP belongs to snippet of the form *Melvin Wayne Osmund has 7 brothers*, where the eighth brother is instantiated. CoQEx gets correct instances except for ***Marie*** who is the Osmund sister. Since all the Osmund siblings were famous musicians, they pop up across relevant snippets.

The fifth query, *how many wives did king solomon have* is interesting since KGs have two instances of Solomon's wives, which would lead a user to believe that 2 is the correct answer. However, multiple snippets confirm that the number is 700 and also provide a relevant instance ***Moti Maris*** which is absent from current KGs[7].

The sixth query, *how many inactive volcanoes in hawaii* is another interesting query which highlights the misunderstanding of document retrievers of ***inactive*** volcanoes as ***active***, since all snippets returned deal with active volcano counts. Here, the instances are important, since those returned fall in the dormant or extinct categories of volcanoes.

Temporal and Logical Query Extensions CoQEx is currently limited to returning answers for queries with explicit evidence in text. Logical operations, such as intersection, difference or union, or temporal filters, are not supported. This refers to queries like *how many players scored more goals score than Messi*, *how many Champions League matches did Messi miss*, *how many goals by Messi after 2016*, which can only be answered if texts explicitly mention the counts.

Our syntactic analysis of count queries from CoQuAD (Section 4.5.2) found that most queries are of simple form, nonetheless, by construction, CoQuAD represents only the head of the distribution, and towards the long tail, complex queries do occur. Future work could look into using CoQEx in

[7] during the period we conducted the experiments

combination with question decomposition [Wolfson et al.(2020)], to tackle more challenging count queries involving temporal conditions, comparative notions and semantic qualifiers. Prior work dealing with entity-centric quantities [Ho et al.(2020)], such as *mountains higher than 1000m*, could be extended to address challenging comparative count queries such as, *how many peaks higher than Mount Kilimanjaro* or *who has scored more goals: Ronaldo or Messi?*.

Answer Explanation Performance The performance of the instance-retrieving answer explanation module leaves considerable space for improvement. Although it outperforms GoogleSDA and QAanswer on the challenging text case by a margin, its absolute performance stands still at only about 10% precision and recall.

Several points appear important here:

1. Going beyond snippets: For pragmatic reasons (avoiding custom scraping of individual websites), CoQEx currently only extracts information from search engine snippets. These are very short and thus naturally limited in recall, and since precision and recall can be traded to some degree, extraction from full websites might provide a handle to also improve precision.

2. Employing domain-specific NER: We are currently employing a generic coarse-grained NER module from SpaCy, along with post-filtering via text entailment. Employing fine-grained NER upfront might provide a better handle on identifying relevant entities [Choi et al.(2018), Onoe and Durrett(2020)].

3. Knowledge-base linking: Entities are currently not linked to KBs, and filtering is purely based on text entailment using the snippet context. Disambiguation to KB entities would provide helpful context that could be better used to decide on which entities to discard/retain.

Multilingualism While research often focuses on few languages, work on multilingual QA is gaining ground, as more benchmarks become available [Ruder and Sil(2021)]. In order to adapt CoQEx to a multilingual setting, we need to identify the language dependent modules. Our candidate generation models (in Section 4.4.1 and 4.4.3) are trained on reading-comprehension datasets. These models would need to be re-trained on language-specific datatsets. Recent literature addresses this by automatically translating existing datasets to other languages [Carrino et al.(2020), Nguyen et al.(2020)]. Similar approach could be applied to the CoQuAD dataset to train the count candidate generation model. The next task would be to adapt the count and instance extraction models. Here too, we can access existing models with a multilingual NER capability [Rahimi et al.(2019), Tedeschi et al.(2021)], but, count extraction in a multilingual setting is not well researched. The entailment model for checking answer type compatibility of the instances and the sentence embedding model for computing semantic relatedness that CoQEx uses are transformer-based models which can be adopted for the language at hand. One should also consider whether the use case focuses on employing multilingual sources at once or on a single language and whether this language is low-resourced when adapting CoQEx to other languages.

Unified QA system We observed that count queries are an underexplored direction for question answering, but in an organic use case, they would naturally constitute only a subset of all queries. Thus, full coverage of uses cases would require a combination of CoQEx with a regular QA system.

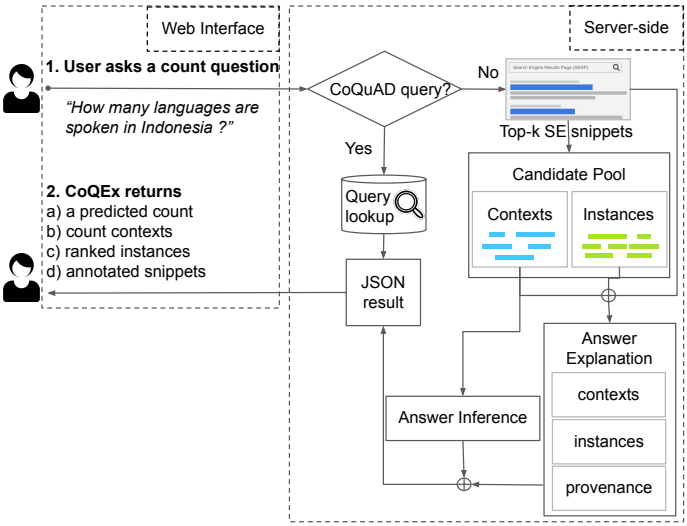


Figure 4.11: Architecture of CoQEx demonstration.

For a joint system, a main challenge would be query classification, to identify when to send queries to which of the subsystems, or to design a post-hoc aggregation of answers from subsystems, like in IBM's Watson [Ferrucci(2012)]. This entails also the special case of dealing with queries beyond the systems capabilities. The current CoQEx is designed to always attempt to return answers, while a joint system should be able to refrain from answering, when queries appear too difficult [Rajpurkar et al.(2018)].

4.8 The CoQEx System

We design CoQEx as a search interface where a user can pose queries on entity counts and expect a comprehensive answer in addition to the snippets from which the answer is derived. Figure 4.11 illustrates the complete architecture of the CoQEx demonstrator. The system supports live queries from the user and precomputed queries from the CoQuAD dataset. Our demo can be accessed at `https://nlcounqer.mpi-inf.mpg.de/`.

4.8.1 System Description

Given a query, CoQEx identifies the query components: the answer type, the named-entities and the remaining context words. It then retrieves the top 50 search-engine snippets using the Bing API. The snippets are then used to form our candidate pool of count contexts and instances. The candidate pool of count contexts contains tuples of text spans, the model confidence and the extracted count: (*700 languages*, 0.8, 700) for the count context *700 languages* predicted by the model with a confidence of 0.8. The instance candidates are processed to extract named-entities using a named-entity recognition tool. Each candidate pool has its own model confidence threshold, ranging between [0,1] with a preset value of 0.5 for count contexts and 0.4 for instances. Additionally, in order to increase recall, the threshold is dynamically lowered by 0.1 if less than 5 candidates remain the pool. This allows the model to make inferences on low confidence candidates.

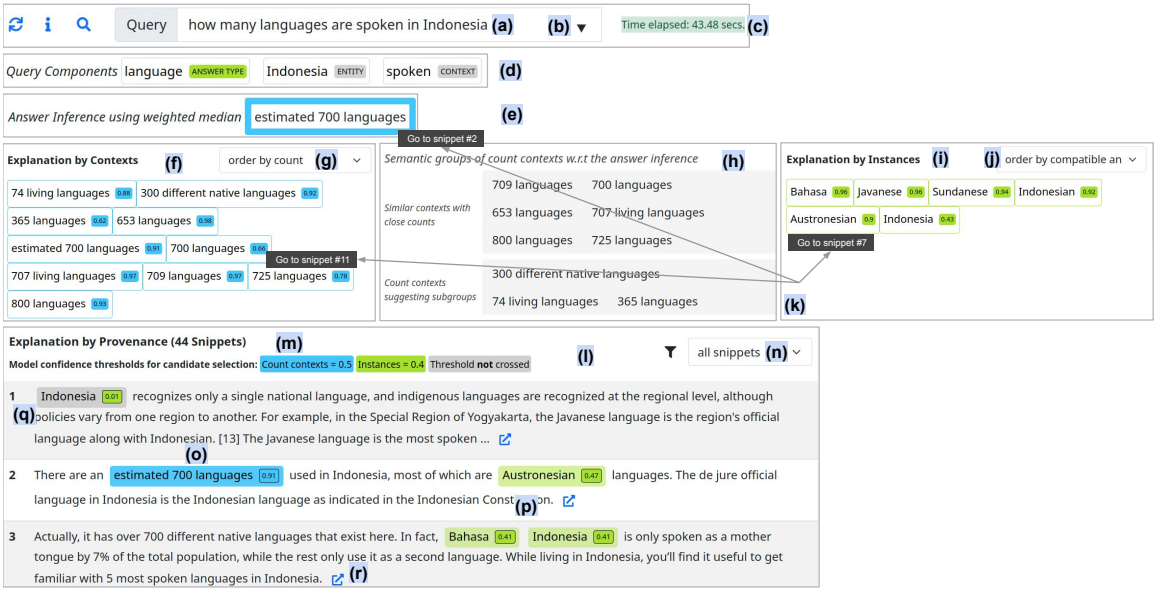


Figure 4.12: The different web interface components illustrated on the result for the query *how many languages are spoken in Indonesia*. Components (a) through (r) are described in Section 4.8.1.

The two candidate pools and the snippets are used to generate the answer comprising the following four components.

1. *Answer inference*: the predicted count context.
2. *Explanation by contexts*: the count context candidates used to infer the answer above and their semantic groups.
3. *Explanation by instances*: the instance candidates which ground the counts.
4. *Explanation by provenance*: the snippets annotated with the count context and instance candidates.

The system is implemented as a Python web application using Flask and hosted on an Apache HTTP server. JavaScript, CSS and HTML are used to build the web interface, while the backend is implemented in Python. Precomputed queries are stored in JSON files and live queries are processed in real-time on search snippets retrieved from the Bing search-engine API[8]. The computation time required for 350 CoQuAD queries is almost 4 hours averaging to roughly 40 seconds per query. Our system runs on a virtual machine with 8 GB RAM and 50 GB disk space. We limit the number of API calls to 100 per day. Nevertheless our code is publicly available for interested users to use their subscription key for making more live queries.

Web Interface Figure 4.12 shows the web interface of CoQEx with the results for the query *how many languages are spoken in Indonesia*. There is an input area for the user to type a natural language query on entity counts **(a)**. On an empty input, a user can also browse the dropdown **(b)**, showing 322 count queries from our CoQuAD dataset. Once the user hits the search button

[8] `https://www.microsoft.com/en-us/bing/apis/bing-web-search-api`

or presses enter on their keyboard, the query is sent to the server. If a query is chosen from the dropdown options, a lookup is performed on the query by computing an exact string match to retrieve the precomputed answer. If the query lookup fails, the system defaults to the live query setting, where all computations occur in real-time. The time taken for computation is displayed beside the query input **(c)** once the results are returned. The query components are displayed below **(d)**. The returned answer has four components and multiple display parameters. The first output the user sees is the answer inference **(e)**, which is the representative count context, *i.e.,* the most confident context with the count matching the consolidated count prediction. Then come the explanation components.

Explanation by counts **(f)** displays all the count contexts used in consolidation. Users can select the display option **(g)** to view the contexts ordered by model confidence and frequency. The semantic groups of the count contexts are displayed in **(h)**. Similar contexts with counts close to the answer inference are shown first. Next come the count contexts suggesting subgroups of the answer inference and finally, the incomparable count contexts. Explanation by instances **(i)** shows all instances ranked by their compatibility scores The user can also choose to view the instances ordered by **(j)** their model confidence and frequency scores as well. When the user hovers on the count contexts or instances, the snippet ID of the source snippet is displayed **(k)** and upon clicking, the user is navigated to the source snippet.

The explanation by provenance **(l)** displays the annotated search snippets. The total number of snippets, model confidence thresholds for each candidate pool **(m)** and the snippet display filter **(n)** is displayed at the top. Using the snippet filter, the user can choose to display only the snippets which have i) count contexts, ii) instances, iii) both count contexts and instances, or iv) or no candidates. The snippet annotation comprises count contexts highlighted in blue **(o)** and the instances highlighted in green **(p)**. Each highlighted component is appended by its score. If the count or instance candidate identified by CoQEx do not meet the minimum threshold requirements to be selected for consolidation, they are highlighted in grey **(q)**. The snippets links allow users to visit the source webpage **(r)**.

4.8.2 Demonstration Experience

Scenario 1: Understanding Entity Count Queries. A user wants to understand what entity count queries are and why consolidation is necessary. They go through the dropdown options in the search bar. They see three categories into which the count queries are arranged, i) *KG-answerable*: queries easily answerable from search-engine Knowledge Graphs (KGs), such as, relatives of celebrities (*how many kids does elon musk have*) or movies by an actor (*how many bruce lee movies are there*). ii) *Snippet-answerable*: queries which can be answered by search-engines using a featured snippet (*how many countries speak english*, *how many zones was germany divided into*). iii) *No direct answers*: queries for which it is difficult for search-engines to get answers from KGs or a single snippet (*how many students at harvard*, or *how many islands are in the galapagos*).

The user notices that the movies and pop culture domains are dominant in KG-answerable queries and that they are usually time-invariant. Being inquisitive about books made into movies, they pose the precomputed query *how many harry potter films*. CoQEx returns the correct answer: ***eight movies*** — and the name of the first movie: Harry Potter and ***the Philosopher's stone***. ***two-film finale*** contextualizes the final book that was made into two films. Another context, ***10***

movies, includes all movies from the Harry Potter franchise. The user makes a query on books: *how many novels did agatha christie write* and gets ***66 detective novels*** as the answer. Upon investigating the source snippet of the subgroup contextualization ***33 novels***, they realize that it represents the number of crime novels Agatha Christie wrote on Hercule Poirot.

Scenario 2: Live Count Query. When the user clicks on a suggested live query: *number of Arab ountries*, CoQEx returns the correct answer: ***22 Arab countries*** with helpful instances like ***Kuwait***, ***UAE***, ***Bahrain***, ***Iraq***, ***Egypt*** and ***Saudi Arabia***. The ***12 countries*** subgroup links to a snippet talking about the number of Arabic countries in Asia, while the incomparable context ***208 listed states*** can be linked to its source snippet talking about the number of sovereign states around the world.

On the query, *how many members in the United Nations*, CoQEx predicts ***193 sovereign member states*** with 10 instances. The number of security council members is contextualized in the ***15 members*** subgroup. Another similar context of ***136 United nations member states*** contextualizes the number of states being elected to the United Nations Security Council. The user follows with the query *how many agencies are under the united nations*. CoQEx returns ***16 specialized agencies***, also identifying ***24 UN agencies*** as a similar context. Instances such as, ***WFP*** (the World Food Program), ***WHO*** (the World Health Organization), ***UNICEF*** (the United Nations Children's Fund) and ***UNDP*** (the United Nations Development Programme) are returned as explanations.

Scenario 3: Querying High Variance Counts. The user suspects that the suggested query *how many languages are spoken in Indonesia* gives rise to high variance counts, and sends it off. CoQEx predicts ***estimated 700 languages*** from synonymous counts ranging from 600 up to 800. A count contexts distinguish that there are ***709 living languages*** and ***300 different native languages***. The high confidence instances returned as answer explanations are ***Javanese***, ***Bahasa***, ***Sundanese*** and ***Indonesian***.

CoQEx predicts the *number of species of fish* to be ***about 32,000*** returning informative subgroup contexts including ***40+ freshwater fish species*** and instance explanations such as, ***Bluefin Tuna***, ***whale shark*** and ***Emerald Cory Catfish***.

4.9 Conclusion

Entity counts are challenging due to variance in semantic qualifiers and incomplete entity mentions. CoQEx tackles entity count queries ranging from telegraphic to full-fledged queries. We make a distribution-aware inference over count contexts, categorize the count contexts into semantic groups, rank instances grounding the count and annotate the source snippets with count context and instance candidates. Even though it is built for entity counts, CoQEx performs reasonably well on non-entity count queries such as, *how many tigers in the world* or *how many shares of Tesla*.

We address the gap in distribution-aware prediction, assimilating semantic qualifiers from web contents and providing explanations through instances for the class of count queries. We systematically analyze count queries, their prevalence, structure and how current state-of-the-art answer them. We provide a thorough analysis of CoQEx components and how it compares to the baselines. We discuss in depth about tackled and open challenges with observations and case studies.

Improving explanation by instances has a major scope for improvement, by incorporating KB knowledge (for improving precision) or scraping list pages from search results (for improving recall). In Section 4.7 we indicate ad-hoc mechanism to identify existing contextualizations in present KB-QA systems. This can be further expanded independently, or CNPs from text could be useful in identifying relevant semantic qualifiers in the KB. In the next chapters, we will look into comparing large entity sets, such as classes, and creating traceable pipelines with LLMs for predicting cardinality from online sources.

Chapter 5

Cardinality Comparison

Contents

5.1 Introduction

Are there more astronauts or Physics Nobel laureates? More nuclear power plants or catholic cathedrals? More lakes or rivers? More airports on this planet than satellites in orbit, or vice versa? Comparative questions of this kind tickle our curiosity, yet are often surprisingly hard to answer. For some comparisons, there are authoritative official sources that provide reliable (albeit not necessarily up-to-date) numbers. In most cases, though, the natural resort is to tap into online sources like knowledge bases (KBs, e.g., Wikidata), search engine (SE, e.g., Bing) results, or large language models (LLMs, e.g., GPT). However, all of these come with biases in what they cover and what not, and often give treacherous signals that lead humans to wrong conclusions.

For example, while we found the number of rivers to be close to 300,000 [Grill et al.(2019)], the number of castles is quite inaccessible and lies for Europe alone between 400,000 to 1.3 million[1]. Wikidata suggests that there are more than 400,000 rivers and roughly 23,000 castles. Major search engines, when probed with different query formulations, pick up on the number of rivers as 250,000 (in the United States) and 10,000 (medieval European) castles, suggesting that there are far more rivers than castles, contradicting reality. Even the GPT-3 language model, which performs well

[1] https://quora.com/How-many-castles-are-there-in-Europe/answer/Michael-Burke-339; https://dw.com/does-germany-really-have-25000-castles/a-42350502

on question answering tasks, is completely off. When prompted with three example questions and their answers, followed by the question asking for the number of rivers, it returns *"an estimated 1.3 million rivers on Earth"*, and for castles it returns, *"There are around 900 castles in the world."* Obviously, neither of these major sources is a true mirror of reality. The online world is inherently hampered by incompleteness and bias. We illustrate two more comparisons in Example 5.1, one comparing the number of lawyers and police officers and another comparing the number of board games and satellites.

Example 5.1: Entity Set Cardinalities on the Web vs. Ground Truth

Source	Lawyers	Police Officers	Rivers	Castles
Wikidata	96,000	8,000	2900	6700
SE results	1.3 million	18,000	17	2600
GPT-3	1.3 million	1.3 million	1500	6000
Ground-truth	3.5 million	13 million	86000	4600

All data collected as of Mar 2023.

Smart humans, on the other hand, are sometimes able to judiciously select online sources as cues, then combine multiple cues in a clever way, and eventually arrive at reasonable estimates at class cardinalities (at least the right order of magnitude) and relative comparisons between classes (just asserting which is bigger). Enrico Fermi, a Physics Nobel Laureate from the early 20th century, was known to be a master of such estimates; hence this kind of problem is also known as Fermi Problem [Wikipedia(2022)].

In this chapter, we aim to emulate a smart human approach. We introduce and study a variety of online signals that could be applied, gaining insights on strengths and weaknesses for different domains of entity classes (e.g., occupations vs. creative works vs. organizations). Moreover, we propose novel techniques for aggregating signals with partial coverage into more reliable estimates on which of the two given classes has more real-world instances.

Approach and Contribution We focus on *dominance estimation*: which of two classes has the higher cardinality. We obtain cues for the numeric cardinalities from three sources: the Wikidata KB via SPARQL queries [Vrandečić and Krötzsch(2014)], the Bing search engine with judicious queries using the CoQEx method [Ghosh et al.(2022b)], and the GPT-3 language model [Brown et al.(2020)] with various prompts. Absolute cardinalities from these sources are often completely wrong; so we interpret them merely as signals to be used for further inference.

The key idea of mitigating these bias effects is to additionally inspect *subgroups* of classes, such as actors or airports by country or georegion (e.g., North America, East Asia, etc.). Such subgroups are orthogonal to the classes under comparison. The hypothesis that we study is that the estimates for subgroups of classes can give more reliable cardinality estimates, at least for some subgroups and for relative comparison. This larger set of finer-grained signals are then aggregated using different techniques proposed in this chapter.

The evaluation dataset consists of 4005 class pairs from 6 diverse domains. Our major finding is that the novel technique of aggregating signals substantially improves the dominance estimation, achieving over 80% accuracy compared to direct source signals.

5.2 Related Work

We focus on bias in the digital world and completeness in information sources, which would most affect dominance estimation. Bias on the Web can have adverse effects unless taken into account in designing systems that use Web data [Baeza-Yates(2018)]. Popular text collections from the Web include Wikipedia, a major source of general knowledge used in construction of large KBs [Suchanek et al.(2007), Auer et al.(2007), Vrandečić(2012)] and popular QA benchmarks [Rajpurkar et al.(2016), Kwiatkowski et al.(2019)], C4 [Raffel et al.(2020)] and the Pile [Gao et al.(2020)] datatsets, publicly available huge datasets used for training some well-known LLMs, such as T5 [Gao et al.(2020)], GPT3 [Brown et al.(2020)], LLaMA [Touvron et al.(2023)]. Previous work has looked into such data sources as well as KBs and LLMs created from them, to assess their quality and study existing social biases that get reflected in the online world [Janowicz et al.(2018), Raffel et al.(2020), Sun and Peng(2021), Dodge et al.(2021)]. Most relevant for our work are the prevalent unbalanced frequency distributions that lead to inconsistencies between distribution in the real-world vs. the Web. One of the first works to highlight this discrepancy call this a reporting bias and measure it in the context of information extraction from texts [Gordon and Van Durme(2013)]. Recent work has studied the effect of inconsistent frequency distribution in the context of training LLM [Wei et al.(2021), Razeghi et al.(2022), Zevallos et al.(2023)].

It is well known that general-knowledge KBs are incomplete [Razniewski et al.(2016), Weikum et al.(2021)] even with their increased coverage over time [Razniewski and Das(2020)]. Techniques like mark and recapture assume sampling from the real-world and, applying them to KB edit history makes severe underestimations, for instance predicting that there are roughly 4M humans [Luggen et al.(2019)][2]. Current powerful SEs are now capable of providing structured answers from their internal KBs, and highlighting the most probable answer in the top snippet. Even so, structured answers are not the norm for more complex or less popular questions. LLMs have been shown to be effective in recalling factual information [Petroni et al.(2019), Karpukhin et al.(2020)]. Nevertheless, they lack scrutability [Hewitt et al.(2023)], rely on high quality prompts, [Jiang et al.(2020)], and are known to struggle with numeric/count information [Lin et al.(2020), Wei et al.(2022b)].

5.3 Design Rationale

Let us continue with the question: *Are there more rivers than castles?* In principle, such questions can be decomposed into cardinality questions: *how many rivers are there?* and *how many castles are there?*, the answers of which are then compared. We therefore identify three problem statements related to comparison questions, of varying informativeness and difficulty.

Problem 5.1: Cardinality Estimation

Determine the absolute cardinality of an input entity set.
What are the absolute cardinalities of the classes rivers and castles?

While cardinality estimation provides the most information, this is also the most challenging task. Estimation methods may give impractical results that are orders of magnitude away from the ground-truth, and for some classes, there might not even exist a widely agreed ground-truth.

[2] `https://cardinal.exascale.info/`

Problem 5.2: Proportionality Estimation

Determine the ratio of the cardinalities of two input entity sets.
What is the ratio of the cardinalities of the classes rivers and castles?

Looking at the relation of pairs of classes reduces the impact of uncertainty: Even if there is no widely agreed count for the number of castles, most estimates might agree that there are more castles than rivers. Determining the actual proportion would be desirable, though it is also subject to uncertainty. Moreover, obtaining ground truths for proportions is still difficult.

Problem 5.3: Dominance Estimation

Determine the larger of two input entity sets.
Are there more rivers or castles?

In this work, we focus on the most approachable task of *dominance estimation*: to determine whether one class is larger than the other. Intuitively, as humans, we can deduce that there are more castles than rivers from our observation that along a river, there are typically several castles. However, *machines are incapable of such reasoning.* Nevertheless, there exists evidence in the form of KB entities and actual counts in web documents, which can be leveraged to predict the bigger of the two classes.

5.4 Methodology

We identify cardinality signals to predict whether a class A is greater than B through the output variable $O_{|A|>|B|}$, defined as:

$$O_{|A|>|B|} = \begin{cases} 1 & \text{if } |A| \text{ is predicted to be bigger than } |B|, \\ -1 & \text{if } |A| \text{ is predicted to be smaller than } |B|, \\ 0 & \text{if the predictor abstains.} \end{cases} \tag{5.1}$$

5.4.1 Basic Cardinality Signals

We obtain cardinality signals from three sources and of two types.

Signal sources We use three different signal sources: KBs, SEs, and LLMs. Each provides a different angle: of entities covered in a KB, of what is popular on the web, and of what information has been distilled by LLMs. Specifically, we look into Wikidata [Vrandečić and Krötzsch(2014)], the top-50 search results by Bing [Microsoft(2022)] and GPT-3 [Brown et al.(2020)].

1. **Knowledge base (KB).** Formulate SPARQL queries to retrieve the count of entities per class from Wikidata, preferring hand-annotated queries over KB-QA systems. KB-QA systems perform worse than hand-annotated queries due to the available system's inability to construct the most accurate SPARQL queries.

2. **Search engine (SE).** Return the most confident cardinality of a class using the CoQEx system for inferring counts from top-50 SE result snippets [Ghosh et al.(2022a)].

3. **Large language model (LLM).** In a few-shot setting, given n cardinality questions and their answers as a prompt, retrieve the cardinality of the class in the $(n+1)^{th}$ question. Specifically, we provide three examples along with the intended question to the GPT-3 model. The examples and the model parameters remain constant for all queries. We process the text output to extract the class cardinality using Python Quantulum3 library[3].

Example 5.2: Signal Sources for Politicians

Wikidata SPARQL Query	`select (count(distinct ?s) as ?cnt) where{`
	`?s wdt:P106 ?o.`
	`?o (wdt:P31|wdt:P279)* wd:Q82955.}`

GPT-3 input prompt	*Q: How many countries are there?* *A: 195 countries.* *Q: How many butterfly species are there?* *A: Around 17500 butterfly species.* *Q: How many people are there?* *A: 7.8 billion people.* *Q: how many politicians are there?* A:
CoQEx input question	*how many politicians are there?*

Signal types We consider two types of signals.

1. **Root signals**. Signals for the count of entities in the class of interest itself, e.g., *rivers* (worldwide).

2. **Subgroup signals**. Signals for counts of subgroups orthogonal to the classes being compared, e.g., *castles in Germany*.

Unlike subclasses, which divide a class based on a type hierarchy and are by design class-specific, we use subgroups which are applicable across classes. In principle, subgroups signals can be computed for a range of subgroups, e.g., by 195 countries, by year/decade, by status, etc. Thus, subgroups provide a dimension for comparing two classes from the same or different domains. If we restrict the domains, we could also focus on specific subgroups, such as subgroups by gender for occupations. In the following we focus on the G20-group of countries [Wikipedia(2023)], as for these, sources tend to have more reliable information.

Note also that subgroup signals normally *do not sum up* to root signals, both for pragmatic reasons (data for certain subgroups is unavailable/incomplete), as well as principled reasons (an entity belonging to several subgroups, or to none).

[3] `https://pypi.org/project/quantulum3/`

5.4.2 Signal Aggregation

Besides using the root signals directly to predict class comparison, we aggregate basic signals in three levels: First we aggregate subgroup signals, after which we include root signals, and then aggregate the resulting signals by sources.

I. Subgroup aggregation

1. By majority: $O^{M}_{|A|>|B|}$ is 1, if at least θ_M percent of the subgroups of A have more entities than the those of B, -1 if at least θ_M have less, else 0.

2. By significance: We perform a one-sided t-test, testing for A bigger than B when the mean of the subgroup cardinalities of A is greater than that of B. We test for A less than B when the reverse is true. In the former case, if the p-value $\leq \alpha$, the subgroup distribution of A is significantly greater than B and $O^{S}_{|A|>|B|}$ is 1. If the p-value $\leq \alpha$ in the latter case, then the reverse is true and $O^{S}_{|A|>|B|}$ is -1. If the p-value $> \alpha$ in either of the cases, the prediction is 0.

II. Root and subgroup aggregation Here, the final prediction is the average over the predictions obtained from i) comparing root signals, ii) majority vote over subgroup signals, and iii) significance test over subgroup signals. We use weights, $W \in [0, 1]$ for the majority and significance predictions, such that,

$$Ensemble^{source}_{|A|>|B|} = \frac{1}{3}(O^{source}_{|A|>|B|} + W_M \cdot O^{source,M}_{|A|>|B|} + W_S \cdot O^{source,S}_{|A|>|B|}) \tag{5.2}$$

Hereby, the weight W_M is the majority ratio, when majority is $> \theta_M$, *i.e.* A bigger than B, and, $W_M = (1 - \textit{majority ratio})$ when B is greater. The significance prediction is weighted by $(1 - \textit{p-value})$, such that the lower the p-value, the higher the significance.

III. Source aggregation In the last level of aggregation, we combine the predictions from all sources by:
Majority vote: if any two sources agree, that label is selected, else the predictor abstains.
Weighted vote: we train a Logistic Regression classifier for each of the signal aggregation cases, to learn the weights of each source.

$$ln(p(|A| > |B|)) = -ln(1 + e^{-\sum_{source} W_{source} \cdot Ensemble^{source}_{|A|>|B|}}) \tag{5.3}$$

$$Ensemble^{weighted}_{|A|>|B|} = \begin{cases} 1 & \text{if } p(|A| > |B|) > p(|A| < |B|), \\ -1 & \text{if } p(|A| > |B|) < p(|A| < |B|), \\ 0 & \text{if } p(|A| > |B|) = p(|A| < |B|) = 0.5. \end{cases} \tag{5.4}$$

5.5 Experiments

Dataset We create a ground-truth (GT) dataset of 90 classes spanning 6 domains (Table 5.1)[4]. Our evaluation set comprises $\binom{90}{2} = 4005$ combinations of class pairs, of which $6 \times \binom{15}{2} = 630$ are in-domain pairs, i.e., both classes belong to the same domain and the remaining 3375 class pairs are inter-domain, i.e., both classes belong to different domains. In order to assess the difficulty of the

[4]Dataset link: `https://github.com/ghoshs/class_cardinality_comparison`.

Table 5.1: Domains and example classes.

Domain	#Classes	Examples
creative work	15	film, board game, book
geographical entities	15	lake, castle, dam
man-made object	15	satellite, submarine
occupation	15	politician, actor, physicist
organization	15	university, football club
species	15	snake, insect, fish

Table 5.2: Accuracy (in %) of cardinality signals. Baselines are highlighted.

Source	Direct Comparison	Subgroup Aggregations		Root and Subgroup Signal Ensembles		
	Root (1)	Majority (2)	T-test (3)	(1)+(2)	(1)+(3)	(1)+(2)+(3)
KB	64.7	57.0	36.0	61.5	**65.9**	61.8
SE	65.4	67.2	39.6	65.7	65.4	**68.4**
LLM	74.4	75.8	57.3	77.1	75.8	**79.4**
Ensemble over Sources (KB, SE, LLM)						
Majority Vote	77.8	76.3	42.9	76.8	78.7	**78.9**
Weighted Vote	78.2	76.2	81.2	79.3	**83.7**	81.3
Non-expert human baseline (direct comparison of class pairs)						
Closed-book	75.0					
Open-book	76.0					

task, we compute the order of magnitude of the ratio of the ground-truth cardinalities of all class pairs. We argue that class pairs with close cardinalities would be more difficult to predict than class pairs whose cardinalities differ by several orders of magnitude. The dataset has more pairs with close cardinalities, and less than 8% of the class pairs have cardinality ratios more than 10^4 orders of magnitude. For instance, airlines (5,000) and national parks (3,369) has a cardinality ratio of $1.4:1$ (higher:lower) while airlines to politicians (6,500,000) has a cardinality ratio of $1.3 \times 10^3 : 1$.

Evaluation Metrics We primarily measure the performance of signals by

- **accuracy**: the percentage of correct predictions relative to all samples.
- **abstention rate**: the percentage of empty predictions, so as to analyze whether low accuracies stem from abstentions or wrong predictions.
- **precision**: the percentage of correct predictions relative to non-abstentions.

Parameters We use the *text-curie-001* model of GPT-3 with temperature set to 0 and maximum tokens to generate set to 15. The subgroup aggregation parameters are θ_M, which we set to 0.5 (more than 50% majority) and α, which we set to 0.05. We train the Logistic Regression classifiers, by dividing the dataset into train and test splits (80:20). We perform a 5-fold cross-validation on the training data to determine the regularization hyperparameter. We evaluate on the whole dataset of 4005 class pairs, except for the weighted vote ensemble, which is evaluated on the test set of 800 pairs.

Table 5.3: Accuracy (in %) of aggregation over root and subgroups by domain.

Domain	**KB**	**SE**	**LLM**	**Best**
Creative work	62.8	54.2	**83.8**	LLM
Geographical entity	**77.1**	60.9	70.4	KB
Man-made object	26.6	77.1	**96.1**	LLM
Occupation	57.1	**80.0**	74.2	SE
Organization	58.0	72.3	**88.5**	LLM
Species	61.9	**78.0**	63.8	SE
Interdomain	62.6	68.0	**79.4**	LLM
All	61.8	68.4	**79.4**	LLM

Table 5.4: Examples from our dataset, ordered from easier to harder.

Class 1	**Class 2**	**GT ratio**	**Root signals**			**Signal ensembles***			**Source ensembles****		**Comment**
			KB	SE	LLM	KB	SE	LLM	Majority	Weighted	
Websites	Religious texts	$6 \times 10^7 : 1$	✓	✓	✓	✓	✓	✓	✓	✓	Strong signals from all sources.
Bacteria species	Bee species	$3.15 \times 10^5 : 1$	✓	x	x	✓	x	x	x	x	KB root signal ratio of 4.8:1 gives correct prediction.
Books	Paintings	$1.3 \times 10^3 : 1$	x	x	✓	x	x	✓	x	✓	Weighted vote picks up weak signal from LLM.
School teachers	Hospitals	$6.07 \times 10^2 : 1$	x	✓	x	x	✓	✓	✓	✓	Correct signals from subgroup agg. in SE and LLM.
Cities	Islands	$9.7 : 1$	x	✓	x	x	✓	✓	✓	x	Weighted vote fails due to strong incorrect KB + weak correct LLM signals.
Actors	Architects	$1.48 : 1$	✓	x	x	✓	x	x	x	✓	Weighted vote leverages strong KB signal.

** Best ensemble for KB: (1)+(3); SE and LLM: (1)+(2)+(3)* from basic signals: Root (1), Majority agg. (2) and T-test agg. (3).
*** Best source ensemble for majority vote: (1)+(2)+(3); weighted vote: (1)+(3).*

Baselines We use three state-of-the-art baselines, namely, the root signals obtained from the three sources: Wikidata (KB), CoQEx (SE), and GPT-3 (LLM). We also have two human baselines obtained in closed-book and open-book settings.

Human Baseline We sampled 100 class pairs, 50 in-domain and 50 inter-domain, to evaluate human performance, i.e., general non-expert users. Each class pair was annotated by three MTurkers, who were given brief descriptions of the classes, and were asked two multiple-choice questions. i) Which class has more entities in real life? Options: *Class 1, Class 2, Equal, Cannot determine*, and ii) How certain are you? Options: *Very sure, Estimation, Guess, No idea.*

In addition, annotators could justify their responses in a free text field. The task had two settings, (i) the closed-book setting, where annotators should answer without consulting external sources, and (ii) the open-book setting, where the annotators were encouraged to perform web research. For the closed-book settings, annotators had 3 minutes per question, for the open-book setting, 8 minutes.

The accuracy in both settings was comparable (75% vs. 76%), although the precision increased substantially more (from 79% to 85%), i.e., in the open-book setting, the additional evidence made annotators more often abstain, instead of guessing.

5.6 Analysis

Performance by Source Table 5.2 shows the accuracy of the sources by their signals. Of the three sources, only LLM surpasses the human baseline. We find that an ensemble over the root and the aggregated subgroup signals performs well for all three sources. Aggregating over subgroup and root signals reduces the rate of abstention from 0.5% to 5%, down to less than 1% consistently across sources. Upon further inspection we find that the t-test subgroup aggregations have the lowest accuracy, despite good precision (LLM: 87%, SE: 83%), due to very high abstention rates (LLM:34%, SE: 52%).

Aggregation over Sources From Table 5.2, we see that learning weights using supervised learning increases robustness across aggregation strategies, as the accuracy does not drop below 76%, hence performing better than any individual source. On the contrary, majority vote over sources is most effective only on root signals. *Notably*, in the weighted vote strategy, subgroup aggregations by t-test does remarkably well, achieving 81% when aggregating just the t-test signals and 83.7% when aggregating the root and t-test signals. In both cases, we noticed high coefficient for LLM, followed by lower coefficients for SE and then KB.

Performance by Domain We analyze the class pairs by each of the 6 domains on the ensemble over root and subgroup signals in Table 5.3. Notice that there is no one winning source, and relying on the best source per domain gives an accuracy above 77%. Estimates of everyday objects such as bicycles or smartphones, are easily available on the web, as reflected in SE (77%) and LLM (96%) accuracies. KB is most accurate on geographical entities (77.14%), outperforming the other two sources by a large margin.

Discussion In Table 5.4 we highlight a few examples from our dataset and the predictions of root signals and the best performing ensembles from Table 5.2. Easy cases, such as the number of websites vs the number of religious texts, have high order of magnitude of the cardinality ratio and are predicted accurately by root signals as well. As the difficulty increases, i.e., the cardinality ratio is closer to 1, we see that most of the time only of the root signals have the correct prediction. In these cases, the ensembles shine by picking up weak signals available from aggregated signals over different sources.

While it is possible, direct comparison questions are more challenging for current SEs and LLMs. For instance, Bing returns irrelevant answers to the question *are there more rivers or lakes?*, while GPT-3 returns a definitive answer that there are more lakes. When prompted for an explanation, the answer by GPT-3 is reversed. LLMs have shown promise on answering questions more reliably when provided with contexts [Lewis et al.(2020)]. We will explore this in the next chapter.

5.7 Conclusion

In this work, we bypass the uncertainty in predicting cardinality of classes and tackle dominance estimation of two classes in the real-world. We simulate a smart-human approach to judiciously select cues from the Web to arrive at our final prediction. Our method identifies cardinality signals by sources and types. Additionally, we propose techniques to combine these signals by aggregating over

orthogonal subgroups and over multiple sources. Our dataset comprises entity sets representing large classes that can be effectively divided in a pre-defined set of subgroups. While this is a pragmatic choice for this work, pre-defining a single set of subgroup (G20 countries) cannot be universally applied: for instance, comparing the number of castles in Germany vs. United Kingdom, or the number of lawyers vs. scientists in India would require different approaches to obtain subgroups. Nevertheless, when direct cardinality signals are unavailable, having subgroups and multiple sources is helpful. Through our experiments on over 4000 class pairs, we show that ensembles capture weak signals thereby performing 9.3% better in accuracy than any single root signal in dominance estimations.

Future directions could also look into an alternative method of addressing cardinality dominance, for instance, via commonsense reasoning for object classes. For instance, a movie has usually one or a few directors and many actors. Hence, we can conclude that there are more actors than directors. Another source of cardinality signal is species estimation methods, which estimate cardinality of a set by capturing and re-capturing random samples. While very popular in biological sciences, we found it particularly challenging to adapt the species estimation technique to KBs. In KBs, the cardinality of a class can be orders of magnitude lower than the ground-truth. Hence, the estimators usually underestimate. In the next chapter, we turn back to predicting cardinality from online sources, focusing on supporting evidences, peers, and the role of LLMs in creating traceable pipelines.

Chapter 6

Cardinality Estimation

Contents

6.1 Introduction

Queries about the number of entities with a certain property (sets), such as the number of Physics Nobel laureates, or the number of lakes worldwide (see questions below), are important in many knowledge-intensive use cases, and form ∼10% of popular QA datasets [Mirza et al.(2018)]. Such queries can cover a wide spectrum, from very well-defined sets (like the Nobels), to overly vague ones (like lakes) and their counts are often present in noisy, contradictory, or semantically not fully aligned form on the Web.

Example 6.1: Spectrum of Count Questions

Q1. *how many Nobel laureates in Physics?*
Q2. *how many films produced by Warner Bros?*
Q3. *how many beaches are Blue Flag certified?*
Q4. *how many lakes are there in the world?*

As defined in Chapter 5, we return to the problem of cardinality estimation. We incorporate important takeaways from previous chapters for addressing this problem on the search engine snippets. From Chapter 4, we learn that **evidence tracing is important for user comprehension.** Counting entities for cardinality estimation is the most transparent approach. Indeed, there exist species sampling approaches based on counting [Luggen et al.(2019)]. But, counting entities from big general-purpose KBs in real-time is inefficient for large sets. Consider, the case where we would like to count the number of scientists. The individual entities may not be direct instances of the class of scientists, but belong to a subclass or subsubclass and so on. The execution time increases rapidly as the path lengths vary. SE snippets provide better coverage. Moreover, in Chapter 5 we observed that KBs as a source are the least accurate in predicting dominance estimation. Additionally, we learn that **LLMs have high accuracy, but give highly volatile answers**. Hence, we depend on SE results for provenance and incorporate LLMs in different stages of our pipeline in a more transparent manner.

Approach and Contribution In this chapter, we propose CardiO (**Cardi**nality predictor from **O**nline sources), a lightweight and modular framework for predicting cardinality on the Web. CardiO extracts all counts from a set of relevant Web snippets and infers the most central count based on semantic and numeric distances from other candidates. In the absence of supporting evidence, the system relies on peer sets of similar size, to provide an estimate. Experiments show that CardiO can produce accurate and traceable counts better than the small (comprising 7 billion parameters) generative LLMs. Although larger models have higher precision, when used to enhance CardiO components, they do not improve the final precision or recall.

CardiO provides an explainable count prediction by combining the relevance of a count and its surrounding context to the input question, with supporting evidence from multiple snippets. We contrast CardiO with previous systems, like CoQEx [Ghosh et al.(2022a)], which has high coverage but is more prone to noise, and recent LLMs like GPT3.5 and Llama2, which have high precision, but low traceability to the information source. Developing techniques to ground LLM generated predictions is an area of active research. Our lightweight CardiO beats Llama-7B and, while large GPT models win in precision and recall, we show that they do not provide additional advantage in CardiO.

6.2 Related Work

Web question answering has evolved from statistical NLP techniques used to extract crisp answers from retrieved passages [Dumais et al.(2002), Radev et al.(2002)] to the newer class LLMs under the retriever-reader paradigm [Karpukhin et al.(2020), Guu et al.(2020)]. Even though benchmark Web question datasets are derived from user questions on the Web [Berant et al.(2013), Kwiatkowski et al.(2019), Joshi et al.(2017)], the domain is restricted to Wikipedia. Current SEs provide varied answers, making the case for answering count questions more challenging.

In Chapter 4, we highlighted the importance of traceability for user comprehension. However, tracing, such as displaying the path of a search engine's internal KB, is available for less than 10% of user questions [Ghosh et al.(2022a), Bolotova et al.(2022)]. In the case of counts, enumerating the set is the most transparent but not the most efficient, especially with low entity recall for bigger and less popular sets. Current LLMs, with billions of parameters, have shown promise in

instruction following tasks [Ouyang et al.(2022), Touvron et al.(2023), Chung et al.(2024)], but are inscrutable black-box functions [Hewitt et al.(2023)]. LLMs are also prone to hallucinations [Ji et al.(2023), Bang et al.(2023)] and validating LLM-generated answers is an active line of research [Tafjord et al.(2022), Wei et al.(2022a), Liu et al.(2023), Asai et al.(2023)].

We apply question reformulation to increase the chances of estimating the cardinality of the queried set by identifying its peers. Conventionally, the motivation for reformulating questions has been to provide clarification to the original question [Rieh and Xie(2006), Boldi et al.(2011)] and more recently reformulations are popular in conversational QA models [Kaiser et al.(2021), Vakulenko et al.(2021)]. While specification and generalization are valid moves to obtain lower and upper bounds, respectively; evaluating the precision of bounds is hard and out of our scope here.

6.3 Design Rationale

In Chapter 5, we investigated three sources to extract cardinality signals, (i) from KBs, (ii) via texts, or (iii) via LLMs. While well-defined sets are easier to query in KBs, low recall and popularity biases in KBs hinder their usage in general settings [Razniewski et al.(2024)]. Moreover, counting entities, especially large sets, is inefficient and has been attempted over static KB versions [Luggen et al.(2019)]. Text extraction alleviates some of these problems, as it is more tolerant to fuzzy matches. Wikipedia as a source for text corpora allows for clean and non-redundant matches, but this does not align with the real-world scenario of a user who performs a search over the Web. Large corpora allow many more than one match, and so the challenges of ranking and aggregation arise. A recent third paradigm is LLM, mixing extractive and predictive paradigms. However, they provide poor insights into their reasoning, and are notorious for inventing answers [Wei et al.(2022a), West et al.(2023)].

Count information in text is usually accompanied by context that informs about the type of entities being counted ("*songs*", "*solo songs*", "*singles*"). Similar to CoQEx [Ghosh et al.(2022a)], we extend the notion of answer type to define the *type of entities being counted*, instead of a number and, we use SE snippets as a pragmatic approach to access diverse information. Given a question, we use the Bing API to retrieve the top 50 snippets. We diverge from CoQEx on two key points:

1. We design more interpretable scores for count contexts to account for semantic similarity of the snippet and sentence to the input query and of the count context to the answer type, in contrast to model confidence scores used by CoQEx.

2. We apply controlled LLM enhancements in different modules of the CardiO pipeline:

   - relevance filter,
   - count representation extraction, and,
   - peer calibration

3. We compare our method with strong LLM-only baselines.

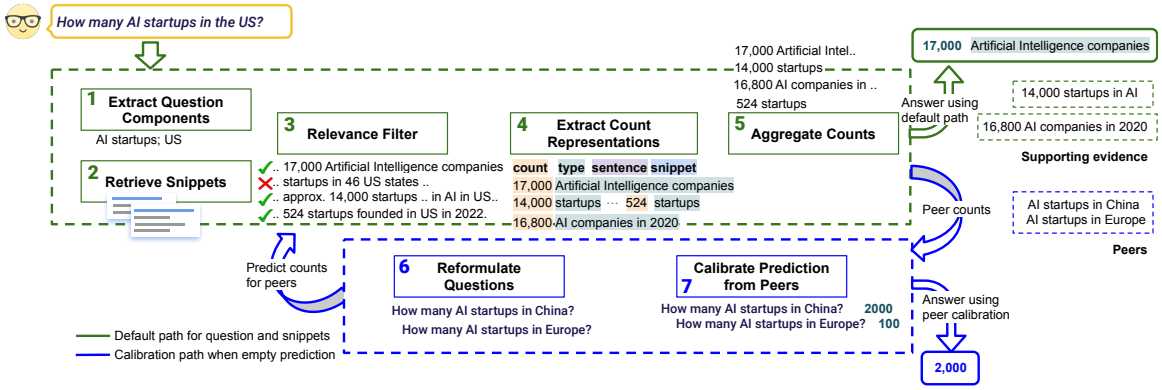


Figure 6.1: A count question with traceable answers, using CardiO for the default path and when peer calibration is applied.

6.4 CardiO Framework

Figure 6.1 provides an overview of the CardiO framework with an illustrating example. We will now go through each component.

Question Components. Given a question q, the *answer type* is the main component that defines the set of entities to be counted. The rest of the components can be seen as constraints on the answer type. Constraints comprise *named-entities*, and other *context* cues such as relations and temporal markers. These components are later used to compute relevance and reformulations.

We use dependency parsing (via SpaCy) to obtain the linguistic features of the question. The *answer type* is identified by the first noun phrase, which contains a noun or a proper noun and its modifiers. The *named entities* are obtained directly from the named-entity recognizer. *Context* tokens include *relation*, which is the first verb in the question, and remaining key phrases in the question.

Relevance Filter. We spot sentences having cardinal mentions, and compute the relevance of each snippet and count-containing sentence to the question from the cosine similarity of the L_2 normalized embedding vectors of the snippet and the question. The embedding vectors come from a sentence transformer encoding model [Reimers and Gurevych(2019)]. We use the *multi-qa-mpnet-base-cos-v1*, which is trained on question-answer pairs for the purpose of semantic search.

Count Representation Extraction. In order to be able to perform numerical operations, we extract the integer value of the counts, and the type being counted. We use the SpaCy pipeline to identify all noun phrases containing counts, extract the quantity using Quantulum[1], and the remaining text is the type. We score the relevance of the *count type* from the snippet to the *answer type* identified in the question using the same embedding technique used for the snippet and sentence relevance scoring.

[1] `https://github.com/nielstron/quantulum3`

Count Aggregation. At this point, we have with us count representations comprising the integer value, the type, the source sentence, and the source snippet. The goal is to find the count that provides a good estimate of the initial user query, q. Each count is initially scored based on the joint relevance scores from the snippet, sentence, and the count type, $R(c) = Rel(c_{snippet}, q) \times Rel(c_{sentence}, q) \times Rel(c_{type}, q_{type})$. We keep only the highest scoring count for each snippet. We use the highest relevance count as a baseline and explore two feature-rich aggregation methods that rely on supporting evidence from other snippets.

- **Most Relevant.** The count with the highest relevance score, $R(c)$, is the final answer. We use this strategy in our **vanilla** setting.

- **Consistent.** Counts with supporting evidence of similar counts from other snippets are highly scored. The final score of a count (Eq. 6.1) is the weighted sum of its relevance score, and the consistency score R_{cons} (the average score of the k nearest neighbors). This is similar to the consistency-based re-scoring of facts [Ho et al.(2021)].

$$F(c) = \beta R(c) + (1 - \beta) R_{cons}(c) \tag{6.1}$$

- **Central.** Using Eq. 6.2, the *centrality* of a count is measured based on its distance from other counts. The closer the relevant counts, the higher is the centrality of a particular count.

$$C(c) = (N - 1) \left(\sum_{\forall c' \neq c} D(c, c') \right)^{-1} \times Rel(c_{sentence}\ ,\ q) \tag{6.2}$$

Both consistent and central aggregations measure the distance between count representations using Eq. 6.4, which is a weighted average of the cosine distance between the sentences and the absolute order of magnitude difference between the counts (Eq. 6.3).

$$D_{order}(c, c') = \log_{10} \left(\frac{max(c, c')}{min(c, c')} \right) \tag{6.3}$$

$$D(c, c') = \alpha D_{sentence}(c, c') + (1 - \alpha) min(1, D_{order}(c, c')) \tag{6.4}$$

Question Reformulation. When there is a lack of count evidence in search snippets, we can rely on peers of the queried entity set to predict the cardinality. These peers are semantic siblings sharing similar answer type and close to each other in size (cardinality). We avoid a lookup of semantic siblings, on popular taxonomies such as WordNet or Wikipedia categories. While the WordNet taxonomy is very restrictive due to canonicalization, leading to poor matches between surface forms, the Wikipedia categories are very extensive, especially for large classes. In this case, harnessing LLM's ability to form semantic associations can be very helpful in returning a handful of most relevant reformulations. We use ChatGPT (version `gpt-3.5-turbo-0613`.) to reformulate a count question, when a named-entity is present, prompting the LLM to replace the entity with others comparable in terms of the answer type. For example, reformulations for the number of *films produced by Warner Bros* are film produced by Walt Disney Pictures, Sony Pictures, and the 20^{th} Century Studios.

Peer Calibration. After having generated reformulations, we typically obtain 5 peer questions per question. These peers are then sorted by their similarity to the original question, removing any peers with a prediction score lower than a threshold. We perform peer calibration when the original prediction is also lower than this threshold. We do not want to replace one weak prediction with another. Finally, the top peer is returned.

LLM-Enhancements. Recent work has shown LLMs to generate better results when prompted to generate steps leading to the final answer, as opposed to directly providing an answer [Wei et al.(2022a)]. However, evaluating such generations is challenging [Gehrmann et al.(2023)]. We apply LLM generations in CardiO as follows, in a more controlled and structured format, and observe the changes in the performance metrics.

- Generating reformulations for peer calibration as explained in **Question Reformulation**. We illustrate this in Example 6.2.

- Tagging sentences that contain counts relevant to the user question in the **Relevance Filter** module. The goal is to improve precision by removing non-matching cardinal mentions early on. For this, we use LLMs to tag sentences having counts related to the question. The model is prompted with the question, a snippet name, and the corresponding snippet, and asked if a given sentence from the snippet has the answer to the question (see Example 6.3).

- **Extracting structured count representations** from sentences. Further, we extract structured count representations by prompting LLMs with the question, a snippet, and a sentence containing one or more cardinals to return a structured representation comprising the text span, the quantity, the type and, a bound (lower, upper, equal, approximate). We illustrate extracted counts from a positively tagged sentence in Example 6.4.

Example 6.2: LLM Generated Reformulations

Prompt:

System: Reformulate the following question by replacing "Bob Dylan" with entities comparable in terms of number of songs. Return each question on a separate line. Start each question with "how many".

User: Question: how many songs did Bob Dylan write for other artists
Reformulations:

Output:

How many songs did Paul McCartney write for other artists?
How many songs did Neil Young write for other artists?
How many songs did Joni Mitchell write for other artists?
How many songs did Elton John write for other artists?
How many songs did Bruce Springsteen write for other artists?

Example 6.3: LLM-enhanced Sentence Tagging

Prompt:

System: Given a question, a sentence from a text and the title of the text, your job is to determine if the sentence contains the answer related to the question. Answer in the provided format, where "short" takes only yes or no and "long" takes a one-line explanation.

User: Question: how many politicians are there in the world
Sentence: 78,382 politicians from 233 countries (so far) Find representatives from your country:
Title: EveryPolitician: Political data for 233 countries
Format: "short": str, "long": str
Answer:

Output:

{"short": "yes", "long": "The sentence has the number of politicians from 233 countries."}

Example 6.4: LLM-based Structured Count Extraction

Prompt:

System: Extract all counts and the types being counted in the provided snippet. Return a list of JSON objects in the provided format. Each value is explained below.
SPAN: sentence substring that represents a count.
TYPE: a string that represents the nouns being counted.
QUANTITY: the integer value in SPAN.
If SPAN is a range, return a list like this: [lower bound, upper bound].
LIMIT: a value from ["upper", "lower", "exact", "approximate", "range"] denoting the uncertainty of the in the SPAN.

User: Snippet: 78,382 politicians from 233 countries (so far) Find representatives from your country:
Format: {"span": SPAN, "type": TYPE, "quantity": QUANTITY, "limit": LIMIT}
Answer:

Output:

[{"span": "78,382", "type": "politicians", "quantity": 78382, "limit": "exact"},
{"span": "233", "type": "countries", "quantity": 233, "limit": "approximate"}]

6.5 Experiments

6.5.1 Cardinality Benchmarks

Few question-answering benchmarks specifically identify count questions. Their answers are reading-comprehension style as text spans. We use the **CoQuAD** test set comprising 312 count questions [Ghosh et al.(2022a)]. These are short entity-specific count questions extracted from search-engine

Table 6.1: Statistics of the CQ dataset by question and ground-truth characteristics.

Label (Value)	Explanation	%age	Examples
Question Characteristics			
popular (1)	on popular topics	49.4	how many cities have hosted the Olympics
popular (0)	on less popular topics	50.6	how many museums are there in the world
specific (1)	on specific entity-set	50.6	how many Ballon d'or winners in Real Madrid
specific (0)	on fuzzy entity-set	49.4	how many bakeries are there in France
has NE (1)	has named-entities	66.2	how many film scores has Hans Zimmer composed
has NE (1)	has no named-entities	33.8	how many skyscrapers are there in the world
Ground-truth Characteristics			
exact (1)	exact cardinality	56.6	how many short film submissions to Sundance film festival in 2022
exact (0)	estimate	43.4	how many bridges in Amsterdam
direct(1)	direct mention	64	how many films has Stephen Spielberg directed
direct(0)	manually aggregated	36	how many films has Sridevi acted in
variant (1)	dynamic / high time-variance	54	how many users in Youtube from India
variant (0)	static / low time-variance	46	how many uninhabited islands in Sweden

query logs. We report on the 84 count questions from **Natural Questions (NQ)** [Kwiatkowski et al.(2019)] extracted by [Ghosh et al.(2022a)].

Cardinality Questions (CQ). We create the CQ benchmark comprising 500 count questions, where we can control the type of counts that can be encountered in the Web. We build our data set on the 90 classes identified in Chapter 5 [Ghosh et al.(2023)]. Ground truth annotations are accompanied by additional labels, which describe whether we have the exact ground truth or an estimate, whether the ground truths are directly available on websites or aggregated manually, whether the entity set is popular or not, and whether the entity set is specific or not. The labels, **question has named-entity, topic is popular, entity set is specific**, characterize the question, and, the labels **ground-truth is exact, ground-truth is directly available, ground-truth is time-variant**, characterize the ground truths. We controlled the dataset to be equally balanced between more popular/less popular entity sets and specific/fuzzy sets. Around 44% of all ground-truths are estimates, reinforcing the importance of an evaluation other than exact match. 36% of all ground-truths are aggregated manually, while 64% are directly available on the Web, further supporting our approach of cardinality prediction from direct mentions. 15% of the ground-truths are static, 31% have low-variance, 23% have high-variance and 31% are dynamic. For simplicity, we club static and low-variance questions as time-invariant and the other two we club as time-variant. We share top-level statistics of the dataset in Table 6.1.

CQ slices. Additional labels of our CQ dataset, allow us to slice the dataset based on combinations of the label values. We have six labels, each of which can be active when it takes a value (popular/less popular, specific/fuzzy, with/without named-entities, exact/estimate, direct mention/manually aggregated, time-variant/time-invariant), or inactive when we don't care about what value the label takes. We can slice the dataset into different slices depending on which of the six labels are active or inactive. This gives us $3^6 - 1 = 728$ possible slices. We compare CardiO (vanilla) and Snippets + Llama2-70B-chat on 553 of these slices, which have more than 10 questions, and an average of 55 questions per slice.

Table 6.2: Performance on cardinality benchmarks, CQ, NQ [Kwiatkowski et al.(2019)] and CoQuAD [Ghosh et al.(2022a)], grouped by answer traceability.

Answer Traceability	Method	CQ (n=500)			NQ (n=84)			CoQuAD (n=312)		
		EP	OMP	OMR	EP	OMP	OMR	EP	OMP	OMR
Not possible	0-shot Llama2-7B-chat	0.056	0.578	0.560	0.250	0.695	0.662	0.137	0.584	0.547
	0-shot Llama2-70B	0.113	0.670	0.521	0.288	0.725	0.630	0.190	0.692	0.548
	0-shot Llama2-70B-chat	0.079	0.653	0.631	0.325	0.775	0.738	0.220	0.646	0.621
	0-shot GPT3.5	0.133	0.716	0.689	0.438	0.797	0.759	0.273	0.724	0.689
Medium	Snippets + Llama2-7B-chat	0.213	0.640	0.553	0.289	0.674	0.610	0.224	0.635	0.572
	Snippets + Llama2-70B	0.242	0.723	0.567	0.338	0.723	0.611	0.320	0.743	0.581
	Snippets + Llama2-70B-chat	0.278	0.743	0.728	0.494	0.838	0.808	0.302	0.749	0.715
	Snippets + GPT3.5	0.303	0.825	0.751	0.548	0.834	0.724	0.381	0.815	0.726
High	CoQEx [Ghosh et al.(2022a)]	0.175	0.631	0.577	0.329	0.665	0.626	0.266	0.696	0.611
	CardiO (vanilla)	0.192	0.659	0.659	0.298	0.661	0.661	0.228	0.618	0.618
	+ LLM Sentence filter	0.191	0.663	0.653	0.293	0.657	0.634	0.229	0.657	0.634
	+ LLM Count extraction	0.174	0.571	0.571	0.298	0.661	0.661	0.205	0.601	0.601
	+ Peer calibration	0.190	0.657	0.657	0.310	0.654	0.654	0.234	0.623	0.623
	+ Central aggregation	0.196 (±0.029)	0.630 (±0.013)	0.630 (±0.013)	0.298	0.634	0.634	0.253	0.649	0.649
	+ Consistent aggregation	0.186 (±0.064)	0.653 (±0.023)	0.653 (±0.023)	0.274	0.660	0.660	0.240	0.642	0.642

6.5.2 Evaluation Metrics

The scope of count questions extend from small sets, such as awards, to intermediate, and very large sets. As the set size increases, ground truths tend to be estimates. In order to handle evaluation of estimates, we compare the order-of-magnitude differences between the prediction and the ground truth. All precision scores are measured on non-empty predictions.

Exact Precision (EP). For a given question, EP measures if the predicted count matches the ground truth count exactly.

Order of Magnitude Precision (OMP). For a given question, it measures if the predicted count is within 1 order of magnitude of the ground truth and by how much (Eq. 6.5). We calculate the order difference first using Eq. 6.3. If the prediction is within one order difference, it returns the degree to which the prediction deviates from the ground truth between $[0, 1]$, else 0.

$$OMP_q = 1.0 - min(1, D_{order}(c_{prediction}, c_{ground\ truth}) \tag{6.5}$$

Order of Magnitude Recall (OMR). It penalizes empty predictions in the presence of ground-truth answers by returning 0. Non-empty predictions are measured using Eq. 6.5.

6.5.3 Baselines

We evaluate CardiO (vanilla), and subsequently report the effect of LLM enhancements, and rich aggregations. We compare CardiO with CoQEx [Ghosh et al.(2022a)], open and semi-open LLMs like Llama2 (70B, 7B-chat, 70B-chat) [Touvron et al.(2023)], and proprietary LLMs like GPT3.5[2]. The LLMs are tested with 0-shot and snippet-augmented 0-shot prompts. In both cases, they are prompted to return a number and a one-line explanation. We then apply our count extractor to get the integer value. We also tested for answers returned in a specified JSON format to avoid the

[2]ChatGPT (`https://openai.com/blog/chatgpt`). Model used: `gpt-3.5-turbo-0613`

count extractor. Since this did not always help, we report only on the performance of the free-form answers.

6.5.4 Parameter Setting

We perform a 5-fold cross-validation and determine the best value for the parameters $\alpha \in [0, 1]$ in Eq. 6.4, $\beta \in [0, 1]$ in Eq. 6.1 and the number of neighbors k to consider for R_{cons}, which are then used on other benchmarks. We report the standard deviation of the *central* and *consistent* aggregation methods in Table 6.2.

6.6 Analysis

Baseline Comparison. In Table 6.2, we notice a gap in the OMP scores of the generative and other lightweight models. Even CoQEx, which uses a supervised masked LM, is almost 0.18 points behind GPT3.5 when augmented with snippets. Our unsupervised CardiO (vanilla) (i) beats the 0-shot models in most settings, (ii) is better than the snippet-augmented Llama2-7B-chat model, and (iii) has higher OMR than the bigger Llama2-70B model. The difference between 0-shot and snippet-augmented LLMs is much higher than that between the pretrained and finetuned Llama models, emphasizing the importance of the snippets. Even so, the snippet-augmented models lack traceability to source snippet, characterizing the count type, providing bound and peer sets provided by CardiO.

Lack of Traceable Baselines. In the 0-shot mode, the LLMs return one-line claims, which require additional verification. When asked *"how many uninhabited islands in Sweden?"*, GPT3.5 returns: ***0** - Sweden does not have any uninhabited islands*, and Llama2 models return: ***221 uninhabited islands** in Sweden (Source: Swedish Agency for Marine and Water Management)*[3]. With snippets, Llama2-7B-chat just returns a number ***267,570***, GPT3.5 returns: *There are over **267,570 islands** in Sweden, with fewer than 1000 of them being inhabited*, which can but need not come from a snippet. CardiO provides the total number of islands, the exact sentence, the snippet, and additional snippets all with numbers in the 200,000s, confirming the answer's order of magnitude. As Sweden's peers CardiO returns Finland, Norway, Indonesia, Canada & Australia. For the number of *musicians who have won a Nobel Prize*, where the snippets do not have the count, CardiO uses peers to predict 5 (7 being the ground truth), while LLM-only models generate names sometimes from the snippets[4]: GPT3.5 names Bob Dylan in both settings, while Llama2-70B-chat returns Bob Dylan and twice Rabindranath Tagore with snippets, and in 0-shot returns Bob Dylan and three names, which on further inspection were not found in the snippets.

LLM Enhancements do not Improve Overall Performance. Firstly, we notice that the LLM sentence filter does not provide much improvement over the vanilla CardiO setting on our CQ and CoQuAD datasets and even lower OMR on the NQ dataset. Upon comparing the effect of the LLM sentence filter with vanilla setting on the CQ dataset, we find that the OMP increases for 7% of the questions and decreases for 11% of the questions. Similarly, when comparing the LLM count

[3]Note that the citation is itself generated, and can, but need not be correct.
[4]Bob Dylan was the most frequent in the snippets, followed by Rabindranath Tagore

extraction method to the vanilla setting, we find that OMP increases for 11% of the questions and decreases 27% of the questions. Peer calibration, when used conditionally with the default path, affects 8 questions, of which 5 gain in OMP. In order to further test the effectiveness of the peer calibration, we evaluate CardiO such that it only makes prediction using peers. Since we obtain peers of named-entity mentions in the question, we perform this evaluation on such queries which have a named-entity mention. CardiO achieves 0.28 in both OMP and OMR on 331 queries, using just peer calibration, indicating high variance among peers.

Effect of Aggregation Strategies. Supporting evidence-based central and consistent aggregation methods boost CardiO (vanilla)'s performance in the CoQuAD dataset, and the central aggregation method has effects on the exact precision. For the central aggregation strategy $\alpha = 0.9$, i.e., the distance metric in Equation 6.4 places more weight on snippet similarity than difference in counts. For the consistent aggregation strategy $\alpha = 0.7$, $\beta = 0.3$ and $k = 3$ perform the best. In all the 5 folds, the number of neighbors, k, is 2 or 3, and the weight for consistency of neighbors, i.e., $\beta \in 0.2, 0.3$. This is a hopeful indication that finding consistent neighbors can boost precision scores. As the number of supporting evidence changes with the questions, these metrics may perform better with thresholding the distance metric than taking top-k neighbors.

Investigating CQ Slices. Of the 553 slices we investigated, we found 73 slices where CardiO performs better in either OMP or OMR or both. In general, the slices CardiO outperforms Snippets + Llama2-70B-chat are small and very specific (4 or more active labels). Most notable conditions where CardiO shines are when questions do not contain named-entities or are on fuzzy entity sets or popular entity sets or a combination of these. We analyze one such question in Example 6.5.

Example 6.5: Prediction analysis: Snippets + Llama2-70B-chat

Question: *how many school teachers are there in the world*
Generated text: *"According to the UNESCO Institute for Statistics, there were approximately 63 million teachers worldwide in 2019. ..(one more explanation sentence)"*
Extracted count: 63,000,000
Ground truth: 85,000,000
Error Analysis: i) "UNESCO Institute for Statistics" is a **hallucinated source**, though there are several snippets referring to data from UNESCO. ii) "approximately 63 million teachers" is a **hallucinated context**. There are no exact matches, with the closest relevant counts in the snippets being "93.7 million teachers worldwide", "between 2000 and 2019, from 62 million to 94 million teachers".

CardiO prediction: 93,700,000
Source context: about 93.7 million teachers worldwide
Source snippet: Data on teachers | Teacher Task Force. teacher data: sdg indicators and other sources. in 2019, there were about 93.7 million teachers worldwide: 11.6 million in pre-primary; 32.6 million in primary; 20.7 in lower secondary; 15.7 in upper secondary; and 13.1 in tertiary education.

Another notable observation is that the questions in these 73 slices are difficult even for the

Llama model. See Example 6.6 Consider that 491 slices where the Llama model is better than CardiO in either OMP or OMR or both. The average OMP and OMR of the Llama model is 0.735 and 0.720, respectively, while that of CardiO is 0.642 for both OMP and OMR. On the 73 slices where CardiO has better performance, its average OMP and OMR is 0.646. In contrast, the Llama model has an average OMP of 0.598 and an average OMR of 0.585.

Example 6.6: Prediction analysis: Snippets + Llama2-70B-chat

Question: *how many AI researchers are there*
Generated text: *"According to the article, there are approximately 300 full-time technical AI safety researchers, 100 full-time non-technical AI safety researchers, and 400 AI safety researchers in total today. ..(one more explanation sentence)"*
Extracted counts: [300, 100, 400] (we evaluate the first count)
Ground truth: 300,000
Error Analysis: The model generates exact text from an article on a **focused subset** - AI safety researchers. Moreover, which article is referred here is unknown.

CardiO prediction: 300,000
Source context: just 300,000 AI researchers
Source snippet: blueTencent says there are only 300,000 AI engineers worldwide, but ... According to the study, compiled by the Tencent Research Institute, there are just 300,000 "AI researchers and practitioners" worldwide, but the "market demand" is for millions of roles.

Resources Used. Retrieving 50 snippets costs 0.05$/question. The cost of prompting GPT models ranges from 0.000075$/question for 0-shot prompts (no traceability) and reformulations, to 0.01$/question for snippet-augmented prompts (limited traceability). Sentence-level extractions are a better compromise between cost and transparency. Both GPT and Llama models are resource- and parameter-heavy models, compared to the small local models used by CardiO.

6.7 Conclusion

In this chapter, we propose CardiO, a modular and lightweight framework to comprehensively answer count questions on the Web. CardiO's simple framework beats small pure LLM-based Llama models. While we observe a performance gap between monolithic LLM-only systems and more transparent and traceable systems, investigating different dataset slices reveal that CardiO performs better than 70-billion chat-based Llama2 model with snippets on more difficult slices. We experiment with different LLM-enhancements in the more transparent CardiO framework, though it does not provide any substantial improvement. The main challenge with LLM-only methods are that the generated answers are not directly traceable to the source, can contain hallucinations, and require additional extraction of a structured count representation. We propose an alternative to generate more traceable results, but the gap in performance suggests room for improvement. Future work could look into methods for effectively extracting structured count representations. Reasoning about the counts could also lead to more accurate and transparent cardinality predictions.

Chapter 7

Conclusion

7.1 Summary

The Web holds abundant information, which is accessed by end users through search and question-answering. The nature of user information needs has evolved from simple lookups queries to queries requiring additional operations, such as aggregation, comparison, and reasoning on retrieved results. In such a scenario, creating transparent systems that allow tracing model predictions to information sources is as important as providing correct answers.

In this context, this thesis focuses on discovering count information on the Web by *extracting* entity counts and *estimating* cardinalities from Web data. To recapitulate, count information models the cardinality of a set of entities, expressed directly as an integer or as a list, which should be enumerated. We focus on count information, as this appropriately captures the coexistence of complementary information, through counts and enumerations, and the variance of source information. In Chapter 1, we motivate the case for count information through multiple examples and describe the challenges associated with extracting, estimating, and organizing count information on the Web.

Throughout the development of this thesis, we have observed promising changes in popular search-engine result interfaces, from Google presenting aggregations on top of its internal KB to Bing highlighting answers appearing in multiple sources. The current Wikidata ontology has extended set predicates with rich properties. For instance, the property *memberCount* imposes a constraint on the types of entities this predicate can count, as opposed to constraints related to measurement units. Examples of the types that can counted by this predicate include country, member states, musical groups and organizations. These developments though promising are not yet as widely used.

We propose the CounQER method, in Chapter 3, to *firstly* identify KB predicates storing count information through supervised classification and to *secondly*, connect semantically related cardinality assertions and enumerations using statistical and linguistic inferences. In order to illustrate the utility of the semantic alignments, we develop an online demonstration that can be used to query count information about KB entities.

We propose CoQEx, in Chapter 4, to answer questions about entity counts from Web snippets, which has higher coverage of information than KBs, but is also prone to noise. We show how to consolidate count distribution from multiple snippets into a comprehensive answer and provide explanatory context to the user. We corroborate the utility of explanatory evidence through user

studies. Further, we develop an online demonstration of CoQEx for answering live count questions on the Web.

In Chapter 5, we introduce the generalized problem of cardinality estimation and develop a method to predict the larger of two classes, when the precision of the class cardinality is unknown. We investigate different sources of information, namely, KBs, LLMs and Web snippets, to compare cardinalities of entity classes. We utilize direct cardinalities as well as subgroup cardinalities to determine the larger of two classes. We find that ensembles perform better since they overcome coverage bias of independent sources.

Finally, we propose CardiO, in Chapter 6, to investigate the performance of small unsupervised models in cardinality estimation. Our modular approach allows us to incorporate LLMs in different stages in our pipeline in a more transparent manner. We find that there exists a trade-off between traceability and precision. Although larger models have higher precision, when used to enhance CardiO components, there is little to no impact on the final precision or recall.

Our work on organizing count information provides crucial insights into the nature of count information online and how to tackle variance in counts, but it is far from complete. Our experiments show that there is a high scope for improvement, especially in deploying LLMs transparently, in incorporating semantic ordering early on in the pipeline and in effectively evaluating bounds. The present suite of precision-driven evaluation metrics does not effectively capture the expressiveness of a rich set of bounds that from different semantic groups — lower bounds from subgroups, upper bounds from subsuming groups and approximations from synonymous groups — captured by the count contexts in search results.

Moreover, while it is easy for humans to classify counts in the search results into semantic groups with respect to the entity set in the input query, we found pre-defined taxonomies, such as WordNet or the Wikipedia category tree, to be quite restrictive for Web snippets and embeddings-based similarity metrics to only infer associations without inducing any order. Training large language models, using distant supervision or prompt-based in-context learning, to infer subsumption hierarchy of count contexts is a promising future direction.

7.2 Outlook

In this section, we highlight some promising directions for improving utilization of count information and extending its usage beyond general-purpose Web search.

Improving Structured Count Representations from LLMs. In this thesis, we show that counts on the Web have a distribution and that they have a rich context. The distribution comes from the variations in semantics, time, and noise. Our methods, CoQEx and CardiO, provide high coverage from Web snippets and full traceability to source snippets from extracted count components but lag behind black-box generative LLMs in overall precision. Our experiments on incorporating these LLMs transparently in existing pipelines show that they do not perform as well as they do in LLM-only settings.

Future directions could explore supervised finetuning of LLMs to extract structured count representations that capture different constraints such as modifiers and temporal indicators. Dataset collection for this supervised tuning could follow our method for collecting and annotating CoQuAD

dataset in Chapter 4. The question and ground-truth characteristics identified through our CQ benchmark in Chapter 6 could be used to ensure high quality questions and annotations.

Increasing Explainability with Reasoning. Count Information provides a unique setting for applying reasoning to support the predictions to count questions. For instance, predicting the larger of two classes, say teachers and pilots, can be solved using commonsense reasoning — i.e., one profession is more specialized than the other, and hence, employs fewer people — without access to the actual counts. This line of approach assumes the existence of such commonsense knowledge that can be accessed by the answering model. Another approach could look at reasoning with semantic variations, constraints, and available bounds and apply numerical operations when required to make a cardinality prediction.

Consider the count question about the number of crime novels written by Agatha Christie. If, on the one hand, the only available count contexts are: *"33 Poirot novels"* and *"a total of 91 novels"*, then a more appropriate answer would be *"between 33 and 91 books"*, given that other snippets suggest that she has written novels around multiple detectives. On the other hand, if there is evidence in one snippet about the *"33 Poirot books"* and another snippets talks about the *"12 novels where Miss Marple appeared"*, one can deduce that there were at least 45 detective novels that Christie wrote.

LLMs for Generating Long Lists. In this thesis, we focus on directly available counts. For large sets, deriving cardinality estimates from counts is easier than counting individual members, since lists are often incomplete. Nevertheless, it would be interesting to develop approaches that actually enumerate a set of entities to access individual members. Wikipedia has about 5 million list pages that contain easily accessible entity enumerations or more difficult to collate lists of lists. Special databases, such as IMDB (for movies) provide advanced search options to access lists of movie-related entities fulfilling certain criteria. LLMs provide a more approachable way of accessing these lists, by way of natural language prompts. Instruction-tuned LLMs are very good at generating answers in the format provided by a user in the prompt. Nevertheless, deriving the most effective prompt is often a result of trial and error, and the generated answer for the same user intent varies widely for small changes to the prompt. Such LLMs also refrain from returning long answers. An interesting direction is to develop systematic methods to generate long lists using LLMs with or without external knowledge.

Count Information as Exploratory Search. Exploratory search is a promising application for count information, especially for larger and unfamiliar entity sets. A user who starts out with a fuzzy search query can make more refined subsequent searches based on the topic-specific variations returned by the search model. Consider a user who wants to know more about tech employment in Europe and searches for the "number of developers in the EU". The resulting search snippets cover a diverse set of aspects, such as the *job market* (number of developers jobs available in the EU in 2023, 2024, ..), the *skill set* (number of Python, Java, .., engineers), and the *geographic distribution* (number of software developers in Germany, France, ..). This also ties back to long lists, where instead of enumerating the whole list, the user is presented with relevant aspects which deal with smaller, more manageable subgroups.

Adding Temporal Context and Heterogeneous Sources. In this thesis, we assume temporally relevant counts to be present in the retrieved set of information. However, specific entity sets would benefit from temporal tracking. Continuing our example of tech developers in the EU, search results with varying temporal information could be presented as a time series. Entity sets related to demographics, such as population, employment, businesses, have high time-variance. When it comes to heterogeneous sources, web tables and list pages found on Wikipedia, are a rich source of count information. Web tables bring their own challenges, such as dealing with varied layouts and extracting context from headers and/or accompanying text. List pages also carry similar challenges, with the additional challenge of accessing list of list pages in real time.

Beyond Counts of Named Entities. We can extend the notion of count information beyond sets of named entities to include sets of any object — number of free kicks by a player, number of cars manufactured in a year, or the number of experiments reported in a study. Thus, we can utilize count information in specialized settings, such as sports analysis, financial statements, and academic works. The counts in such other settings also include a type (of the objects being counted) and constraints in the form of modifiers, relation, and temporal information that would be useful for answering user queries.

List of Algorithms

List of Figures

List of Tables

Bibliography

[Abujabal et al.(2017)] Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*.

[Arnaout(2023)] Hiba Arnaout. 2023. *Enriching open-world knowledge graphs with expressive negative statements*. Ph. D. Dissertation. MPI for Informatics, Max Planck Society.

[Arnaout et al.(2021)] Hiba Arnaout, Simon Razniewski, Gerhard Weikum, and Jeff Z Pan. 2021. Negative statements considered useful. *Journal of Web Semantics* (2021).

[Asai et al.(2023)] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *The Twelfth International Conference on Learning Representations (ICLR 2023)*.

[Auer et al.(2007)] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *International Semantic Web Conference*.

[Baeza-Yates(2018)] Ricardo Baeza-Yates. 2018. Bias on the web. In *Communications of the ACM*.

[Baidu(2020)] Baidu. 2020. Introducing Qian Yan, Baidu's New Plan to Build 100 Chinese NLP Datasets in Three Years. http://research.baidu.com/Blog/index-view?id=146.

[Bang et al.(2023)] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers) (IJCNLP-ACL 2023)*.

[Bast and Haussmann(2015)] Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on Freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM 2015)*.

[Bauer et al.(2018)] Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for Generative Multi-Hop Question Answering Tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*.

[Berant et al.(2013)] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*.

[Bing(2022)] Search Bing. 2022. Bing Web Search API. https://www.microsoft.com/en-us/bing/apis/bing-web-search-api.

[Boldi et al.(2011)] Paolo Boldi, Francesco Bonchi, Carlos Castillo, and Sebastiano Vigna. 2011. Query reformulation mining: models, patterns, and applications. *Information retrieval* (2011).

[Boldyrev et al.(2018)] Natalia Boldyrev, Marc Spaniol, and Gerhard Weikum. 2018. Multi-Cultural Interlinking of Web Taxonomies with ACROSS. *Journal of Web Science* (2018).

[Bollacker et al.(2008)] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

[Bolotova et al.(2022)] Valeriia Bolotova et al. 2022. A non-factoid question-answering taxonomy. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022)*.

[Brachman and Levesque(2004)] Ronald Brachman and Hector Levesque. 2004. *Knowledge representation and reasoning*. Elsevier.

[Brown et al.(2020)] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

[Calvanese et al.(2017)] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. 2017. Ontop: Answering SPARQL queries over relational databases. *SWJ* (2017).

[Calvanese et al.(1998)] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. 1998. Description logics for conceptual data modeling. In *Logics for databases and information systems*.

[Carrino et al.(2020)] Casimiro Pio Carrino, Marta Ruiz Costa-jussà, and José A. R. Fonollosa. 2020. Automatic Spanish Translation of SQuAD Dataset for Multi-lingual Question Answering. In *Proceedings of the Twelfth Language Resources and Evaluation Conference (LREC 2020)*.

[Chah(2018)] Niel Chah. 2018. OK Google, What Is Your Ontology? Or: Exploring Freebase Classification to Understand Google's Knowledge Graph. *arXiv preprint arXiv:1805.03885* (2018).

[Chen et al.(2019)] Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Evaluating question answering evaluation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*.

[Chen et al.(2017)] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL 2017)*.

[Choi et al.(2018)] Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-Fine Entity Typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL 2018)*.

[Christmann et al.(2023)] Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum. 2023. Explainable Conversational Question Answering over Heterogeneous Sources via Iterative Graph Neural Networks. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2023)*.

[Chung et al.(2024)] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research* (2024).

[Crawl(2008)] Common Crawl. 2008. Common crawl dataset. `https://commoncrawl.org/`.

[Darari et al.(2015)] Fariz Darari, Radityo Eko Prasojo, and Werner Nutt. 2015. Expressing no-value information in RDF. In *Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC 2015)*.

[Day(2015)] Charles Day. 2015. One million physicists. (2015).

[Devlin et al.(2019)] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the NAACL-HLT*.

[Dhingra et al.(2022)] Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics* (2022).

[Diefenbach et al.(2018)] Dennis Diefenbach, Vanessa López, Kamal Deep Singh, and Pierre Maret. 2018. Core techniques of question answering systems over knowledge bases: a survey. (2018).

[Diefenbach et al.(2019)] Dennis Diefenbach, Pedro Henrique Migliatti, Omar Qawasmeh, Vincent Lully, Kamal Singh, and Pierre Maret. 2019. QAnswer: A Question Answering prototype bridging the gap between a considerable part of the LOD cloud and end-users. In *The World Wide Web Conference (WWW 2019)*.

[Dodge et al.(2021)] Jesse Dodge et al. 2021. Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021)*.

[Dong(2019)] Xin Luna Dong. 2019. Building a broad knowledge graph for products. In *2019 IEEE 35th International Conference on Data Engineering (ICDE 2019)*.

[Dua et al.(2019)] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (NAACL-HLT 2019)*.

[Dubey et al.(2019)] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and DBpedia.

[Dumais et al.(2002)] Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: Is more always better?. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*.

[Dziri et al.(2024)] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. 2024. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems* (2024).

[Elazar et al.(2019)] Yanai Elazar et al. 2019. How Large Are Lions? Inducing Distributions over Quantitative Attributes. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*.

[Elazar et al.(2023)] Yanai Elazar, Akshita Bhagia, Ian Helgi Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Evan Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, et al. 2023. What's In My Big Data?. In *The Twelfth International Conference on Learning Representations (ICLR 2023)*.

[Euzenat and Shvaiko(2007)] Jérôme Euzenat and Pavel Shvaiko. 2007. *Ontology matching*. Springer.

[Fader et al.(2011)] Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*.

[Fan et al.(2019)] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long Form Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*.

[Ferrucci(2012)] David A Ferrucci. 2012. This is Watson. *IBM Journal of Research and Development* (2012).

[Galárraga et al.(2017)] Luis Galárraga, Simon Razniewski, Antoine Amarilli, and Fabian M Suchanek. 2017. Predicting Completeness in Knowledge Bases. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM 2017)*.

[Galárraga et al.(2013)] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological

knowledge bases. In *Proceedings of the 22nd International Conference on World Wide Web (WWW 2013)*.

[Gao et al.(2020)] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027* (2020).

[Gehrmann et al.(2023)] Sebastian Gehrmann et al. 2023. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *Journal of Artificial Intelligence Research* (2023).

[Gelman et al.(2008)] Andrew Gelman, Aleks Jakulin, Maria Grazia Pittau, Yu-Sung Su, et al. 2008. A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics* (2008).

[Ghosh et al.(2022a)] Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. 2022a. Answering Count Queries with Explanatory Evidence. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022)*.

[Ghosh et al.(2022b)] Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. 2022b. Answering Count Questions with Structured Answers from Text. In *Journal of Web Semantics*.

[Ghosh et al.(2023)] Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. 2023. Class Cardinality Comparison as a Fermi Problem. In *Companion Proceedings of the ACM Web Conference 2023 (WWW 2023)*.

[Giacometti et al.(2019)] Arnaud Giacometti, Béatrice Markhoff, and Arnaud Soulet. 2019. Mining significant maximum cardinalities in knowledge bases. In *International Semantic Web Conference (ISWC 2019)*.

[Google(2010)] Google. 2010. Google Custom Search API. https://developers.google.com/custom-search/v1/introduction.

[Gordon and Van Durme(2013)] Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction (AKBC 2013)*.

[Grill et al.(2019)] Günther Grill et al. 2019. Mapping the world's free-flowing rivers. In *Nature*.

[Gu et al.(2021)] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021 (WWW 2021)*.

[Guan et al.(2022)] Saiping Guan, Xueqi Cheng, Long Bai, Fujun Zhang, Zixuan Li, Yutao Zeng, Xiaolong Jin, and Jiafeng Guo. 2022. What is event knowledge graph: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[Guu et al.(2020)] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*.

[Helwe et al.(2021)] Chadi Helwe, Chloé Clavel, and Fabian Suchanek. 2021. Reasoning with transformer-based models: Deep learning, but shallow reasoning. In *International Conference on Automated Knowledge Base Construction (AKBC 2021)*.

[Hewitt et al.(2023)] John Hewitt, John Thickstun, Christopher D Manning, and Percy Liang. 2023. Backpack Language Models. In *The 61st Annual Meeting Of The Association For Computational Linguistics (ACL 2023)*.

[Ho et al.(2020)] Vinh Thinh Ho, Koninika Pal, Niko Kleer, Klaus Berberich, and Gerhard Weikum. 2020. Entities with Quantities: Extraction, Search, and Ranking. In *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM 2020)*.

[Ho et al.(2021)] Vinh Thinh Ho, Koninika Pal, Simon Razniewski, Klaus Berberich, and Gerhard Weikum. 2021. Extracting contextualized quantity facts from web tables. In *Proceedings of the Web Conference 2021 (WWW 2021)*.

[Ho et al.(2022)] Vinh Thinh Ho, Daria Stepanova, Dragan Milchevski, Jannik Strötgen, and Gerhard Weikum. 2022. Enhancing knowledge bases with quantity facts. In *Proceedings of the Web Conference 2022 (WWW 2022)*.

[Hoffart et al.(2013)] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial intelligence* (2013).

[Hogan et al.(2021)] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. *Comput. Surveys* (2021).

[Hollunder and Baader(1991)] Bernhard Hollunder and Franz Baader. 1991. Qualifying number restrictions in concept languages. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR 1991)*.

[Hopkinson et al.(2018)] Andrew Hopkinson, Amit Gurdasani, Dave Palfrey, and Arpit Mittal. 2018. Demand-Weighted Completeness Prediction for a Knowledge Base. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)* (2018).

[Imani et al.(2023)] Shima Imani, Liang Du, and Harsh Shrivastava. 2023. MathPrompter: Mathematical Reasoning using Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track) (ACL 2023)*.

[Izacard and Grave(2021)] Gautier Izacard and Édouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume (EACL 2021)*.

[Jain et al.(2010)] Prateek Jain, Pascal Hitzler, Amit P Sheth, Kunal Verma, and Peter Z Yeh. 2010. Ontology alignment for linked open data. In *International Semantic Web Conference (ISWC 2010)*.

[Jain et al.(2020)] Prachi Jain, Sushant Rathi, Soumen Chakrabarti, et al. 2020. Temporal Knowledge Base Completion: New Algorithms and Evaluation Protocols. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*.

[Janowicz et al.(2018)] Krzysztof Janowicz et al. 2018. Debiasing Knowledge Graphs: Why Female Presidents are not like Female Popes. In *ISWC Workshops*.

[Järvelin and Kekäläinen(2002)] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* (2002).

[Ji et al.(2021)] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems* (2021).

[Ji et al.(2023)] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surveys* (2023).

[Jiang et al.(2021)] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics* (2021).

[Jiang et al.(2020)] Zhengbao Jiang, Frank F. Xu, J. Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know?. In *Transactions of the Association for Computational Linguistics*.

[Joshi et al.(2020)] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics* (2020).

[Joshi et al.(2017)] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL 2017)*.

[Kacupaj et al.(2021)] Endri Kacupaj, Barshana Banerjee, Kuldeep Singh, and Jens Lehmann. 2021. ParaQA: A Question Answering Dataset with Paraphrase Responses for Single-Turn Conversation. In *European Semantic Web Conference (ESWC 2021)*.

[Kacupaj et al.(2020)] Endri Kacupaj, Hamid Zafar, Jens Lehmann, and Maria Maleshkova. 2020. VQuAnDa: Verbalization question answering dataset. In *European Semantic Web Conference (ESWC 2020)*.

[Kaiser et al.(2021)] Magdalena Kaiser, Rishiraj Saha Roy, and Gerhard Weikum. 2021. Reinforcement learning from reformulations in conversational question answering over knowledge graphs. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*.

[Karpukhin et al.(2020)] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*.

[Kojima et al.(2022)] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems* (2022).

[Krishna et al.(2021)] Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to Progress in Long-form Question Answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[Kwiatkowski et al.(2019)] Tom Kwiatkowski et al. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics* (2019).

[Lehmann and Hitzler(2010)] Jens Lehmann and Pascal Hitzler. 2010. Concept learning in description logics using refinement operators. *Machine Learning* (2010).

[Lehmann et al.(2015)] Jens Lehmann, Robert Isele, et al. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *SWJ* (2015).

[Lenat(1995)] Douglas B Lenat. 1995. CYC: A large-scale investment in knowledge infrastructure. *Commun. ACM* (1995).

[Lewis et al.(2020)] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* (2020).

[Lin et al.(2020)] Bill Yuchen Lin et al. 2020. Birds have four legs?! NumerSense: Probing Numerical Commonsense Knowledge of Pre-Trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*.

[Lin et al.(2015)] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2015)*.

[Ling and Weld(2010)] Xiao Ling and Daniel S Weld. 2010. Temporal information extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2010)*.

[Liu and Singh(2004)] Hugo Liu and Push Singh. 2004. ConceptNet—a practical commonsense reasoning tool-kit. *BT technology journal* (2004).

[Liu et al.(2023)] Kevin Liu, Stephen Casper, Dylan Hadfield-Menell, and Jacob Andreas. 2023. Cognitive Dissonance: Why Do Language Model Outputs Disagree with Internal Representations of Truthfulness?. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*.

[Liu et al.(2019)] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized bert pretraining approach. In *arXiv*.

[Lu et al.(2019)] Xiaolu Lu, Soumajit Pramanik, Rishiraj Saha Roy, Abdalghani Abujabal, Yafang Wang, and Gerhard Weikum. 2019. Answering complex questions by joining multi-document evidence with quasi knowledge graphs. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*.

[Luggen et al.(2019)] Michael Luggen, Djellel Difallah, Cristina Sarasua, Gianluca Demartini, and Philippe Cudré-Mauroux. 2019. Non-parametric class completeness estimators for collaborative knowledge graphs—the case of wikidata. In *International Semantic Web Conference (ISWC 2019)*.

[McGuinness et al.(2004)] Deborah L McGuinness, Frank Van Harmelen, et al. 2004. OWL web ontology language overview. *W3C recommendation* (2004).

[Microsoft(2022)] Microsoft. 2022. Bing search API. `https://www.microsoft.com/en-us/bing/apis/bing-custom-search-api`.

[Miller(1995)] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* (1995).

[Min et al.(2023)] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *Comput. Surveys* (2023).

[Mirza et al.(2017)] Paramita Mirza, Simon Razniewski, Fariz Darari, and Gerhard Weikum. 2017. Cardinal Virtues: Extracting Relation Cardinalities from Text. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (ACL 2017)*.

[Mirza et al.(2018)] Paramita Mirza, Simon Razniewski, Fariz Darari, and Gerhard Weikum. 2018. Enriching Knowledge Bases with Counting Quantifiers. In *International Semantic Web Conference (ISWC 2018)*.

[Mirza et al.(2016)] Paramita Mirza, Simon Razniewski, and Werner Nutt. 2016. Expanding Wikidata's Parenthood Information by 178%, or How To Mine Relation Cardinalities. In *Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016)*.

[MusicBrainz(2003)] MusicBrainz. 2003. MusicBrainz Database. `https://musicbrainz.org/`.

[Navarro(2001)] Gonzalo Navarro. 2001. A guided tour to approximate string matching. (2001).

[Navigli and Ponzetto(2010)] Roberto Navigli and Simone Paolo Ponzetto. 2010. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*.

[Neumaier et al.(2016)] Sebastian Neumaier, Jürgen Umbrich, Josiane Xavier Parreira, and Axel Polleres. 2016. Multi-level semantic labelling of numerical values. In *The Semantic Web–ISWC 2016: 15th International Semantic Web Conference (ISWC 2016)*.

[Nguyen et al.(2020)] Kiet Van Nguyen, Duc-Vu Nguyen, Anh Gia-Tuan Nguyen, and Ngan Luu-Thuy Nguyen. 2020. A Vietnamese Dataset for Evaluating Machine Reading Comprehension. In *Proceedings of the 28th International Conference on Computational Linguistics*.

[Nguyen et al.(2021)] Tuan-Phong Nguyen, Simon Razniewski, and Gerhard Weikum. 2021. Advanced semantics for commonsense knowledge extraction. In *Proceedings of the Web Conference 2021 (WWW 2021)*.

[Niepert et al.(2010)] Mathias Niepert, Christian Meilicke, and Heiner Stuckenschmidt. 2010. A probabilistic-logical framework for ontology matching. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2010)*.

[Onoe and Durrett(2020)] Yasumasa Onoe and Greg Durrett. 2020. Fine-grained entity typing for domain independent entity linking. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2020)*.

[Ouyang et al.(2022)] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* (2022).

[Patel-Schneider(2018)] Peter F Patel-Schneider. 2018. Contextualization via qualifiers. In *Emerging Topics in Semantic Technologies at ISWC 2018*.

[Paulheim(2017)] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web* (2017).

[Pennington et al.(2014)] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[Petroni et al.(2019)] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases?. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*.

[Qian(2013)] Richard Qian. 2013. Understand Your World with Bing. https://blogs.bing.com/search/March-2013/Understand-Your-World-with-Bing.

[Radev et al.(2002)] Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. 2002. Probabilistic question answering on the Web. In *Proceedings of the 11th International Conference on World Wide Web (WWW 2002)*.

[Radford et al.(2018)] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[Raffel et al.(2020)] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. In *Journal of Machine Learning Research*.

[Rahimi et al.(2019)] Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively Multilingual Transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*.

[Rahm and Bernstein(2001)] Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *VLDB Journal* (2001).

[Rajpurkar et al.(2018)] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (ACL 2018)*.

[Rajpurkar et al.(2016)] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (EMNLP 2016)*.

[Razeghi et al.(2022)] Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of Pretraining Term Frequencies on Few-Shot Numerical Reasoning. *Findings of the Association for Computational Linguistics: EMNLP 2022* (2022).

[Razniewski et al.(2024)] Simon Razniewski, Hiba Arnaout, Shrestha Ghosh, and Fabian Suchanek. 2024. Completeness, Recall, and Negation in Open-World Knowledge Bases: A Survey. *Comput. Surveys* (2024).

[Razniewski and Das(2020)] Simon Razniewski and Priyanka Das. 2020. Structured Knowledge: Have we made progress? An extrinsic study of KB coverage over 19 years. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM 2020)*.

[Razniewski et al.(2019)] Simon Razniewski, Nitisha Jain, Paramita Mirza, and Gerhard Weikum. 2019. Coverage of Information Extraction from Sentences and Paragraphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*.

[Razniewski et al.(2016)] Simon Razniewski, Fabian Suchanek, and Werner Nutt. 2016. But What Do We Actually Know?. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction (AKBC 2016)*.

[Reddy et al.(2019)] Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A Conversational Question Answering Challenge. *Transactions of the Association for Computational Linguistics* (2019).

[Reimers and Gurevych(2019)] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*.

[Rieh and Xie(2006)] Soo Young Rieh and Hong Xie. 2006. Analysis of multiple query reformulations on the web. *Information Processing and Management* (2006).

[Rogers et al.(2023)] Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2023. Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *Comput. Surveys* (2023).

[Romero et al.(2019)] Julien Romero, Simon Razniewski, Koninika Pal, Jeff Z. Pan, Archit Sakhadeo, and Gerhard Weikum. 2019. Commonsense properties from query logs and question answering forums. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM 2019)*.

[Roy and Anand(2022)] Rishiraj Saha Roy and Avishek Anand. 2022. *Question answering for the curated web: Tasks and methods in qa over knowledge bases and text collections*.

[Roy et al.(2015)] Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about Quantities in Natural Language. *Transactions of the Association for Computational Linguistics* (2015).

[Ruder and Sil(2021)] Sebastian Ruder and Avik Sil. 2021. Multi-Domain Multilingual Question Answering. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts* (2021).

[Safavi et al.(2021)] Tara Safavi et al. 2021. Report on the first workshop on bias in automatic knowledge graph construction at AKBC 2020. In *ACM SIGIR Forum*.

[Saha et al.(2017)] Swarnadeep Saha, Harinder Pal, and Mausam. 2017. Bootstrapping for Numerical Open IE. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (ACL 2017)*.

[Saha Roy and Anand(2020)] Rishiraj Saha Roy and Avishek Anand. 2020. Question Answering over Curated and Open Web Sources. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020)*.

[Sanh et al.(2019)] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*.

[Sap et al.(2019)] Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI conference on artificial intelligence (AAAI 2019)*.

[Sarawagi(2008)] Sunita Sarawagi. 2008. Information extraction. *Foundations and Trends® in Databases* (2008).

[Sarawagi and Chakrabarti(2014)] Sunita Sarawagi and Soumen Chakrabarti. 2014. Open-domain quantity queries on web tables: annotation, response, and consensus models. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[Shvaiko and Euzenat(2013)] Pavel Shvaiko and Jérôme Euzenat. 2013. Ontology matching: state of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering* (2013).

[Siciliani et al.(2022)] Lucia Siciliani, Pierpaolo Basile, Pasquale Lops, and Giovanni Semeraro. 2022. MQALD: Evaluating the impact of modifiers in question answering over knowledge graphs. *Semantic Web* (2022).

[Singhal(2012)] Amit Singhal. 2012. Introducing the knowledge graph: Things, not strings. https://www.blog.google/products/search/introducing-knowledge-graph-things-not/.

[Singhania et al.(2023)] Sneha Singhania, Simon Razniewski, and Gerhard Weikum. 2023. Extracting Multi-valued Relations from Language Models. In *Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023)*.

[Soulet et al.(2018)] Arnaud Soulet, Arnaud Giacometti, Béatrice Markhoff, and Fabian M Suchanek. 2018. Representativeness of Knowledge Bases with the Generalized Benford's Law. In *International Semantic Web Conference (ISWC 2018)*.

[Staab and Studer(2013)] Steffen Staab and Rudi Studer. 2013. *Handbook on ontologies*. Springer Science & Business Media.

[Stelmakh et al.(2022)] Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. ASQA: Factoid Questions Meet Long-Form Answers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022)*.

[Subercaze(2017)] Julien Subercaze. 2017. Chaudron: extending DBpedia with measurement. In *The Semantic Web: 14th International Conference, ESWC 2017*.

[Suchanek(2020)] Fabian Suchanek. 2020. The need to move beyond triples. In *International Workshop on Narrative Extraction from Texts*.

[Suchanek and Weikum(2013)] Fabian Suchanek and Gerhard Weikum. 2013. Knowledge harvesting in the big-data era. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*.

[Suchanek et al.(2011)] Fabian M Suchanek, Serge Abiteboul, and Pierre Senellart. 2011. PARIS: Probabilistic alignment of relations, instances, and schema. *VLDB* (2011).

[Suchanek et al.(2007)] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web (WWW 2007)*.

[Sullivan(2020)] Danny Sullivan. 2020. How Google autocomplete predictions are generated. `https://blog.google/products/search/how-google-autocomplete-predictions-work/`

[Sun and Peng(2021)] Jiao Sun and Nanyun Peng. 2021. Men Are Elected, Women Are Married: Events Gender Bias on Wikipedia. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers) (ACL-IJCNLP 2021)*.

[Tafjord et al.(2022)] Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2022. Entailer: Answering Questions with Faithful and Truthful Chains of Reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processin (EMNLP 2022)*.

[Talmor and Berant(2018)] Alon Talmor and Jonathan Berant. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

[Tandon et al.(2014)] Niket Tandon, Gerard De Melo, Fabian Suchanek, and Gerhard Weikum. 2014. Webchild: Harvesting and organizing commonsense knowledge from the web. In *Proceedings of the 7th ACM international conference on Web Search and Data Mining (WSDM 2014)*.

[Tanon et al.(2017)] Thomas Pellissier Tanon, Daria Stepanova, Simon Razniewski, Paramita Mirza, and Gerhard Weikum. 2017. Completeness-Aware Rule Learning from Knowledge Graphs. In *International Semantic Web Conference (ISWC 2017)*.

[Tedeschi et al.(2021)] Simone Tedeschi, Valentino Maiorca, Niccolò Campolungo, Francesco Cecconi, and Roberto Navigli. 2021. WikiNEuRal: Combined Neural and Knowledge-based Silver Data Creation for Multilingual NER. In *Findings of the Association for Computational Linguistics: EMNLP 202*.

[Tibshirani(1996)] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society* (1996).

[Touvron et al.(2023)] Hugo Touvron et al. 2023. Llama 2: Open foundation and fine-tuned chat models. (2023). arXiv:2307.09288

[Trushkowsky et al.(2013)] Beth Trushkowsky, Tim Kraska, Michael J. Franklin, and Purnamrita Sarkar. 2013. Crowdsourced enumeration queries. In *IEEE 29th International Conference on Data Engineering (ICDE 2013)*.

[Usbeck et al.(2018)] Ricardo Usbeck, Ria Gusmita, Muhammad Saleem, and Axel-Cyrille Ngonga Ngomo. 2018. 9th Challenge on Question Answering over Linked Data (QALD-9). In *International Semantic Web Conference (ISWC 2018)*.

[Vakulenko et al.(2021)] Svitlana Vakulenko, Shayne Longpre, Zhucheng Tu, and Raviteja Anantha. 2021. Question rewriting for conversational question answering. In *Proceedings of the 14th International Conference on Web Search and Data Mining (WSDM 2021)*.

[Voorhees(2001)] Ellen M Voorhees. 2001. Overview of the TREC 2001 question answering track. In *Proceedings 2001 Text Retrieval Conference (TREC 2001)*.

[Vrandečić(2012)] Denny Vrandečić. 2012. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st international conference on World Wide Web (WWW 2012)*.

[Vrandečić and Krötzsch(2014)] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* (2014).

[Wang et al.(2018)] Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018. Evidence Aggregation for Answer Re-Ranking in Open-Domain Question Answering. In *Proceedings of the 6th International Conference on Learning Representation.*

[Wang et al.(2013)] Zhichun Wang, Juanzi Li, Yue Zhao, Rossi Setchi, and Jie Tang. 2013. A unified approach to matching semantic data on the Web. *Knowledge-Based Systems* (2013).

[Wang et al.(2014)] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2014).*

[Wei et al.(2022a)] Jason Wei et al. 2022a. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* (2022).

[Wei et al.(2021)] Jason Wei, Dan Garrette, Tal Linzen, and Ellie Pavlick. 2021. Frequency Effects on Syntactic Rule Learning in Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021).*

[Wei et al.(2022b)] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022b. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research* (2022).

[Weikum et al.(2021)] Gerhard Weikum et al. 2021. Machine Knowledge: Creation and Curation of Comprehensive Knowledge Bases. In *Foundations and Trends in Databases.*

[Weikum and Theobald(2010)] Gerhard Weikum and Martin Theobald. 2010. From information to knowledge: harvesting entities and relationships from web sources. In *ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.*

[Weischedel et al.(2012)] Ralph Weischedel et al. 2012. *OntoNotes Release 5.0. Linguistic Data Consortium.* Technical Report.

[West et al.(2023)] Peter West, Ximing Lu, Nouha Dziri, Faeze Brahman, Linjie Li, Jena D Hwang, Liwei Jiang, Jillian Fisher, Abhilasha Ravichander, Khyathi Chandu, et al. 2023. THE GENERATIVE AI PARADOX:"What It Can Create, It May Not Understand". In *The Twelfth International Conference on Learning Representations.*

[Wienand and Paulheim(2014)] Dominik Wienand and Heiko Paulheim. 2014. Detecting incorrect numerical data in DBpedia. In *The Semantic Web: Trends and Challenges: 11th International Conference, ESWC 2014.*

[Wikimedia(2007)] Wikimedia. 2007. Wikipedia dataset. `https://archive.org/details/wikimediadownloads`.

[Wikipedia(2022)] Wikipedia. 2022. Fermi Problem. `https://en.wikipedia.org/wiki/Fermi_problem`.

[Wikipedia(2023)] Wikipedia. 2023. G20 countries. `https://en.wikipedia.org/wiki/G20`.

[Wolfson et al.(2020)] Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break It Down: A Question Understanding Benchmark. *Transactions of the Association for Computational Linguistics* (2020).

[Wu et al.(2014)] Honghan Wu, Boris Villazon-Terrazas, Jeff Z Pan, and Jose Manuel Gomez-Perez. 2014. How redundant is it?: An empirical analysis on linked datasets. *Proceedings of the 5th International Workshop on Consuming Linked Data (COLD 2014) co-located with the 13th International Semantic Web Conference (ISWC 2014)* (2014).

[Xu et al.(2016)] Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Hybrid question answering over knowledge base and free text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

[Yu et al.(2022)] Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. 2022. KG-FiD: Infusing Knowledge Graph in Fusion-in-Decoder for Open-Domain Question Answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL 2022)*.

[Zaveri et al.(2016)] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. 2016. Quality assessment for linked data: A survey. *Semantic Web* (2016).

[Zeng et al.(2020)] Changchang Zeng, Shaobo Li, Qin Li, Jie Hu, and Jianjun Hu. 2020. A survey on machine reading comprehension—tasks, evaluation metrics and benchmark datasets. *Applied Sciences* (2020).

[Zevallos et al.(2023)] Rodolfo Zevallos, Mireia Farrús, and Núria Bel. 2023. Frequency Balanced Datasets Lead to Better Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

[Zhu et al.(2021)] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering. In *arXiv*.