CoQEx: Entity Counts Explained

Shrestha Ghosh
Max Planck Institute for Informatics
Saarland University
Saarbruecken, Germany
ghoshs@mpi-inf.mpg.de

Simon Razniewski
Max Planck Institute for Informatics
Saarbruecken, Germany
srazniew@mpi-inf.mpg.de

Gerhard Weikum
Max Planck Institute for Informatics
Saarbruecken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

For open-domain question answering, queries on entity counts, such as *how many languages are spoken in Indonesia*, are challenging. Such queries can be answered through succinct contexts with counts: *estimated 700 languages*, and instances: *Javanese* and *Sundanese*. Answer candidates naturally give rise to a distribution, where count contexts denoting the queried entity counts and their semantic subgroups often coexist, while the instances ground the counts in their constituting entities. In this demo we showcase the CoQEx methodology (Count Queries Explained) [5, 6], which aggregates and structures explanatory evidence across search snippets, for answering user queries related to entity counts [4]. Given a entity count query, our system CoQEx retrieves search-snippets and provides the user with a distribution-aware prediction prediction, categorizes the count contexts into semantic groups and ranks instances grounding the counts, all in real-time. Our demo can be accessed at *https://nlcounqer.mpi-inf.mpg.de/*.

CCS CONCEPTS

• **Information systems → Question answering**.

KEYWORDS

Count Information, Explainable AI, Question Answering

ACM Reference Format:
Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. 2023. CoQEx: Entity Counts Explained. In *Proceedings of the Sixteenth ACM Int'l Conference on Web Search and Data Mining (WSDM '23), Feb. 27–March 3, 2023, Singapore, Singapore*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3539597.3573021

1 INTRODUCTION

Motivation and Problem. Open-domain Question Answering (QA) over structured and unstructured knowledge sources typically focuses on a single correct answer [3, 8, 11]. The long-form non-factoid QA is an alternate QA paradigm that supports passage answers or verbalization [1, 10, 14] for reasoning, *why does a year have 365 days*, or procedural queries, *how to make bread from scratch*. These approaches do not fully cater to the class of queries on entity counts [12]. Queries on entity counts are about entities in relation to a set of entities, for instance, *how many languages are spoken in Indonesia*, where the entity Indonesia is related to the set of languages spoken in the country. There may be multiple correct answers with the variance stemming from different applicable semantic qualifiers such as *estimated 700 languages*, *707 living languages*, *300 different native languages*, and alternative representations through instances such as the text span: *examples include Javanese and Sundanese*.

Standard QA and web search approaches focus on retriever-reader methods, such that the user is presented with a ranked-list of text segments (retriever) and a highlighted text span (reader) within the segment [2, 7]. While answer candidates are collected from multiple text segments, the final answer is a single span, which can be traced back to one particular text segment. State-of-the-art web search engines provide structured answers from their internal Knowledge Bases (KBs) only for a fraction of the user queries, with the majority being answered as a featured search snippet with an additional highlighted text span [1, 5]. However, users often come across varying and conflicting numbers across text segments. On web-search for the languages in Indonesia query, one comes across the count contexts *300 native languages* in one text segment, *707 living languages* in another text segment, and *an estimated 700 languages* in yet another text segment. This variance in contexts across texts reduces user comprehensibility and trust in the systems' chosen top answer. Moreover, as these QA systems are evaluated on string similarity (exact match and F1), nearly correct answers which represent semantic sub-groups or synonyms of the ideal answer stand at a disadvantage. Since very recently, the Bing search engine also reports the number of sources containing the same answer, for increased user confidence in an answer confirmed by multiple sources.

Methodology. Through our CoQEx system, based on [5], we address three main challenges while answering entity count queries: **I.** explanation by count contexts, **II.** explanation by instances and **III.** explanation by provenance. We make a distribution-aware count prediction (**I**), categorize the count contexts into semantic groups (**I**), rank the instances based on their compatibility with the answer type (**II**) and annotate count context and instance candidates in the snippets (**III**). Evidence-based answers can increase user comprehension by providing a consolidated picture of answer variance and complexity. This distribution-aware prediction could be extended to population estimation queries, such as *how many tigers in the world*, where text sources are richer in candidate answers and more variant. Another application would be to complement KB-QA especially in incomplete entity mentions and sparsely populated long-tailed entities, when standard KB approaches of aggregation on entities would fail.

When a popular query topic (regarding movies, actors, books) or a query on a popular entity (Meryl Streep, Albert Einstein, Jane

WSDM '23, February 27–March 3, 2023, Singapore, Singapore
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9407-9/23/02.
https://doi.org/10.1145/3539597.3573021

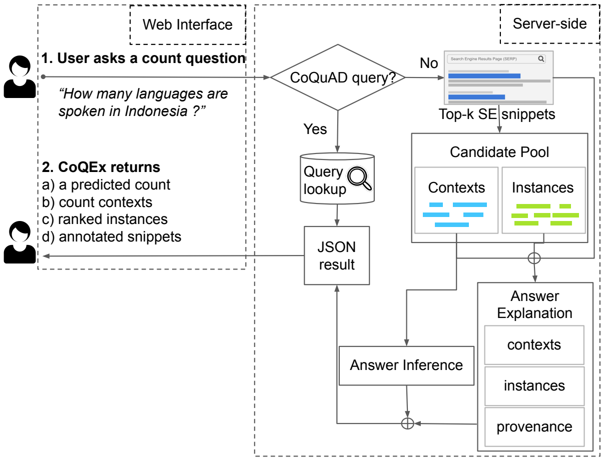


Figure 1: Architecture of CoQEx demonstration.

Austen) comes through, for example, *how many books on Famous Five*, a popular children’s detective book, search engines provide an answer from their internal KB, accompanied by the query path (*The Famous Five* > *Books* > *Count*). However, if the query is changed to *number of books by Enid Blyton*, asking for all books by the author, the search engine goes back to standard search snippets, highlighting counts ranging from *185 novels* she wrote, *15 Secret Seven* to *21 Famous Five* books, the latter two denoting the number of books in a particular series, *762 books* referring to the fiction titles up to the *3621 books* including songs, book reviews, etc. that she wrote, which might be confusing for the end user.

Demonstration. The system demonstrated in this paper is called CoQEx. Essentially, given a full-fledged or a telegraphic query on entity counts, CoQEx retrieves the top-50 search-engine snippets from Bing. The system uses a span-based QA model to separately extract candidate count contexts and instances. The answer inference is achieved by computing a distribution-aware inference over count contexts. The count contexts are further classified into semantic groups with respect to the inferred answer depending on whether they contexts are quite similar to the inferred answer or they represent a subset of the inferred answer or if they are incomparable. The instances are ranked based on their compatibility with the answer type, where the answer type is obtained from the query. Refer to [5] and [6] for details on the CoQEx methodology and the Count Queries Dataset (CoQuAD) used to train the span prediction model for predicting count contexts.

2 COQEX: SYSTEM DESCRIPTION

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 1 illustrates the complete architecture of the CoQEx demonstrator. The system supports live queries from the user and precomputed queries from the CoQuAD dataset. 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[1]. 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.

2.1 Methodology

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 count contexts are obtained by running a QA task on a transformer-based span prediction model [9] finetuned on the CoQuAD dataset [5] over all snippets. Then an extraction module extracts the numbers from these text spans. 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 obtained by running a QA task on another copy of the span prediction model finetuned on SQuADv2 dataset [13] over all snippets. The text spans obtained are then processed to extract named-entities using named-entity recognition tool. Each candidate pool has its own model confidence threshold, ranging between [0,1] with the preset value of 0.5 for count contexts and 0.4 for instances. Additionally, 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.

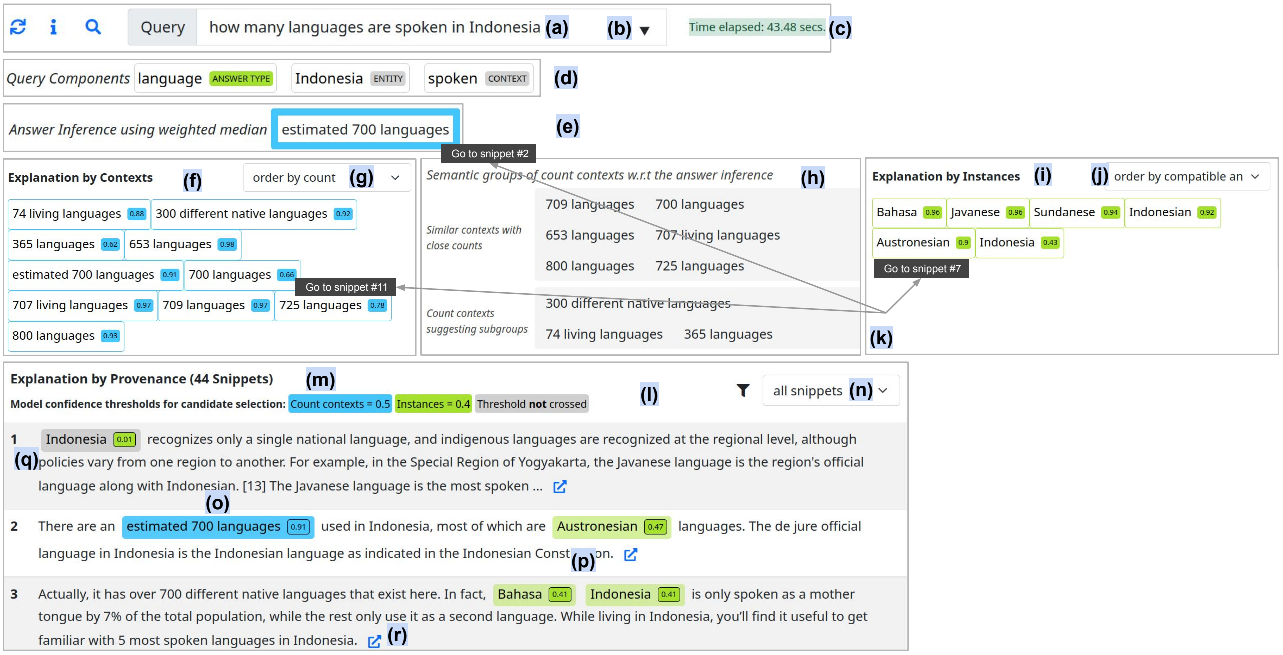
The two candidate pools and the snippets are used to generate the answer to the given query, which has four components.

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 ***answer inference*** module consolidates the count contexts to make a single prediction. We consolidate the counts using weighted median - median of the counts weighted by the model confidence. Once we have the count, we select the representative context to go with this prediction. If there exists multiple contexts with counts equal to the predicted count then the context with the highest model confidence is chosen as the representative context. The representative context is the main answer displayed to the user.

In ***explanation by contexts*** the count contexts used to derive the weighted median is displayed with the model confidence. The user can also view the contexts ordered by the model confidence or the count frequency, both of which are alternate consolidation methods. Next come the semantic groups of count contexts in relation to the representative context, as either similar alternatives, likely subgroups or incomparables. This is achieved by comparing the count value as well as the semantic similarity of the count context with the answer inference.

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



1 Indonesia 0.01 recognizes only a single national language, and indigenous languages are recognized at the regional level, although policies vary from one region to another. For example, in the Special Region of Yogyakarta, the Javanese language is the region's official language along with Indonesian. [13] The Javanese language is the most spoken ...

2 There are an estimated 700 languages 0.91 used in Indonesia, most of which are Austronesian 0.47 languages. The de jure official language in Indonesia is the Indonesian language as indicated in the Indonesian Constitution.

3 Actually, it has over 700 different native languages that exist here. In fact, Bahasa 0.41 Indonesia 0.41 is only spoken as a mother tongue by 7% of the total population, while the rest only use it as a second language. While living in Indonesia, you'll find it useful to get familiar with 5 most spoken languages in Indonesia.

Figure 2: 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 2.2.

In ***explanation by instances*** the instance candidates are ranked based on their answer-type compatibility scores. The instances can also be ordered by two alternative consolidation strategies: the model confidence and the frequency of the instances summed across snippets.

In ***explanation by provenance*** all snippets are annotated with the originating url, the context and instance boundaries and their corresponding model confidence scores. This helps in identifying which candidates belong to which snippets and the candidates that did not cross the model confidence threshold.

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.

2.2 Web Interface

Figure 2 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)**, which contains 322 count queries from the CoQuAD dataset. Once the user hits the search button 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)**.

3 DEMONSTRATION SCENARIOS

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***.

Limitations. CoQEx returns reasonably good answers to long-tailed and low resource topics, as it relies on consolidation from multiple snippets. We identify three sources of errors that occur during (i) *retrieval*, where the quality of snippets and robustness to small variations in the query is dependent on the retrieval model of the search-engines. (ii) *Candidate extraction*, where candidates may be missed or upon identification do not cross the selection thresholds. (iii) *Consolidation*, where the entailment model fails to determine the compatibility with the answer type. For example, on the query, *how many works by Premchand*, CoQEx predicts ***107 original works*** and identifies ***14 novels*** as relevant subgroup context. It also identifies Premchand's most famous work *Godan*, but with zero confidence, indicating that the entailment model failed. Other relevant candidate instances like *Gaban*, *Shatranz*, *Karambhoomi* do not cross the minimum threshold of the model.

4 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*. Studying the effects of other text-specific signals, including the presence of query components (entity and context terms) in the text segments to determine salience or the date of creation of the text to determine recency are possible extensions for future work.

REFERENCES

[1] Valeriia Bolotova, Vladislav Blinov, Falk Scholer, W. Bruce Croft, and Mark Sanderson. 2022. A Non-Factoid Question-Answering Taxonomy. In *SIGIR*.

[2] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *ACL*.

[3] Dennis Diefenbach et al. 2017. Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information Systems* (2017).

[4] Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. 2020. Uncovering hidden semantics of set information in knowledge bases. *JWS* (2020).

[5] Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. 2022. Answering Count Queries with Explanatory Evidence. In *SIGIR*.

[6] Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. 2022. Answering Count Questions with Structured Answers from Text. *JWS* (2022).

[7] Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. Retrieve, Read, Rerank: Towards End-to-End Multi-Document Reading Comprehension. In *ACL*.

[8] Zhen Huang et al. 2020. Recent Trends in Deep Learning Based Open-Domain Textual Question Answering Systems. *IEEE Access* (2020).

[9] Mandar Joshi et al. 2020. Spanbert: Improving pre-training by representing and predicting spans. *TACL* (2020).

[10] Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to Progress in Long-form Question Answering. In *NAACL*.

[11] Xiaolu Lu et al. 2019. Answering complex questions by joining multi-document evidence with quasi knowledge graphs. In *SIGIR*.

[12] Paramita Mirza, Simon Razniewski, Fariz Darari, and Gerhard Weikum. 2018. Enriching knowledge bases with counting quantifiers. In *ISWC*.

[13] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*.

[14] Ming Zhu, Aman Ahuja, Da-Cheng Juan, Wei Wei, and Chandan K Reddy. 2020. Question answering with long multiple-span answers. In *Findings of EMNLP*.