Answering Count Questions with Structured Answers from Text

Shrestha Ghosh[a,b,*], Simon Razniewski[a] and Gerhard Weikum[a]

[a]Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbruecken, 66125, Germany
[b]Saarland University, Saarland Informatics Campus, Saarbruecken, 66125, Germany

ARTICLE INFO

Keywords:
Question Answering
Count Queries
Explainable AI

ABSTRACT

In this work we address the challenging case of answering count queries in web search, such as *number of songs by John Lennon*. Prior methods merely answer these with a single, and sometimes puzzling number or return a ranked list of text snippets with different numbers. This paper proposes a methodology for answering count queries with inference, contextualization and explanatory evidence. Unlike previous systems, our method infers final answers from multiple observations, supports semantic qualifiers for the counts, and provides evidence by enumerating representative instances. Experiments with a wide variety of queries, including existing benchmark show the benefits of our method, and the influence of specific parameter settings. Our code, data and an interactive system demonstration are publicly available at https://github.com/ghoshs/CoQEx and https://nlcounqer.mpi-inf.mpg.de/.

1. Introduction

Motivation and Problem. Question answering (QA) and web search with telegraphic queries have been greatly advanced over the last decade (Balog, 2018; Diefenbach et al., 2018; Huang et al., 2020; Usbeck et al., 2019). Nevertheless, queries that can have multiple correct answers due to variance in semantic qualifiers (*top 10 albums*, *singles albums*, *remastered albums*) and alternative representations through instances remain underexplored and pose open challenges. This paper addresses the class of *count queries*, to return the number of instances that have a certain property. Examples are:

- *How many songs did John Lennon write for the Beatles?*
- *How many languages are spoken in Indonesia?*
- *How many unicorn companies are there?*

Count queries are frequent in search engine logs as well as QA benchmarks (Rajpurkar et al., 2016; Kwiatkowski et al., 2019; Dubey et al., 2019; Voorhees, 2001). If the required data is in a structured knowledge base (KB) such as Wikidata (Vrandečić and Krötzsch, 2014), then answering is relatively straightforward. However, 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 (Ghosh et al., 2020). Besides, evaluating the additional condition *for the Beatles* (i.e., a subset of his songs) is beyond their scope. In Figure 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 hardly doubt the output of 160 songs to the SPARQL query, which contains songs jointly written by Lennon with his co-band member.

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 1). We also see incorrect spans being highlighted *22 songs* in the second snippet, which is actually a count of songs written by George Harrison.

Answering count queries from web contents 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. This paper presents CoQEx, Count Question answering with Explanatory evidence, which answers count

*Corresponding author
ghoshs@mpi-inf.mpg.de (S. Ghosh); srazniew@mpi-inf.mpg.de (S. Razniewski); weikum@mpi-inf.mpg.de (G. Weikum)
ORCID(s):

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

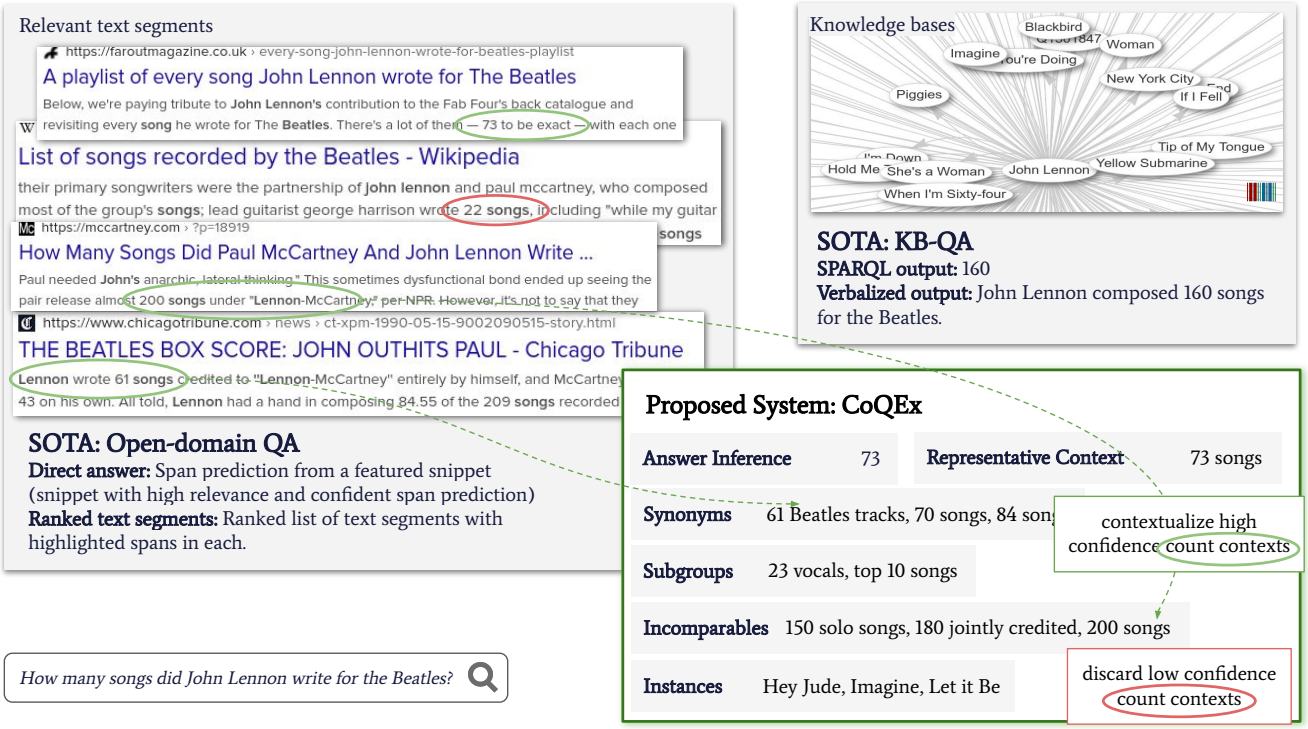

Figure 1: User experience with state-of-the-art QA systems against our proposed methodology for count question answering. In the existing setting, a user would often encounter varying counts in returned text snippets (open-domain QA), or limited context (KB-QA). CoQEx provides a more comprehensive answer, by aggregating evidence across text snippets, contextualizing contexts and providing instances as count explanations.

queries via three components: i) *answer inference* ii) *answer contextualization* and, 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. releasing an annotated data resource with 5k count queries and 200k text segments, available at `https://github.com/ghoshs/CoQEx`.

Previous publication. The present manuscript substantially extends a short paper by Ghosh et al. (2022) by a thorough analysis of the problem of count question answering. In particular, we analyse the characteristic of real count queries, and dissect the difficulty and tractability of each of the subproblems. The major extensions to the short paper are:

1. We substantially expanded the task data, and, based on novel annotation, provide an in-depth analysis of count query characteristics in Section 5.2. In particular, we group queries by complexity (in reach for today's structured approaches, in reach for textual approaches, out of reach), and analyze baseline and system performance for each of them (Tables 3 and 5 in 7.2 and 7.5). We also analyze the domains and stability properties of count queries (Figures 5 and 6), and added another related dataset (NaturalQuestions) to corroborate the findings from our newly-built CoQuAD dataset. Our findings include that our proposed system is resilient to harder query types, with comparable precision at moderate loss of recall.
2. We assess the quality of the automated ground truth extractions in our CoQuAD dataset in comparison with human annotations in Section 5.1. The automated extractions match the human annotations in 84% of the cases accepting small deviations and in 81% of the cases they match exactly. Upon examining the causes of incorrect ground truth extractions and we found that a majority of errors stem from counts of subsets.

3. We provide a novel analysis of count contextualization across different QA paradigms in Section 7.3, finding that KB-QA and search engines struggle with this for principled and/or pragmatic reasons, thus further motivating our answer contextualization step.

4. We provide a full component analysis on our CoQEx system in Section 8, specifically through Table 8 and Figures 8, 9 and 10. A main takeaway is how significant thresholding choices are in answer inference and answer explanation, compared with a lower impact of the pre-training choice and consolidation strategy. For answer contextualization, we observe that synonyms are easier to discern than subgroups and incomparables.

5. We conduct a third user study to verify user satisfaction with the answers provided by CoQEx in Section 7.6. 92% of the users were at least satisfied with the system provided answers on a five-level Likert scale thus reinforcing the effectiveness of our system.

6. We provide a discussion in Section 9 which highlights open challenges for future work, specifically, how count contexts and instance explanations coexist and contribute to better user comprehension. We also examine insightful case studies which show the potential of CoQEx in explaining count queries through hierarchical count contexts and instances.

2. Related Work

Where structured data is available in KBs, structured QA is the method of choice for count question answering, and previous work of Ghosh et al. (2020) has looked at identifying count information inside KBs. However, for many topics, no relevant count information can be found in KBs. For example, Wikidata contains 217 songs attributed to John Lennon[2], but is incomplete in indicating whether these written for the Beatles or otherwise. In the KB-QA domain, systems like QAnswer developed by 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. Attempts have also been made to improve recall by hybrid QA over text and KB, yet without specific consideration of counts (Lu et al., 2019; Xu et al., 2016).

State-of-the-art systems typically approach QA via the machine reading paradigm (Karpukhin et al., 2020; Joshi et al., 2020; Sanh et al., 2019; Chen et al., 2017; Dua et al., 2019), where the systems find the best answer in a given text segment. The retriever-reader approach in open-domain QA uses several text segments to return either a single best answer (Chen et al., 2017; Wang et al., 2018) or a ranked list of documents with the best answer per document (Karpukhin et al., 2020). The DPR system by Karpukhin et al. (2020)[3] returns *approximately 180* from its rank-1 text segment to both, the simple John Lennon query, 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). Thus, QA systems are not robust (yet) and lack explanatory evidence beyond merely returning the top-ranked text snippet.

Attempts have also been made to improve recall by hybrid QA over text and KB, yet without specific consideration of counts (Lu et al., 2019; Xu et al., 2016; Saha Roy and Anand, 2020). Search engines can answer simple count queries from their underlying KBs, if present, a trait which we exploit to create our CoQuAD dataset (Section 5). But more often they return informative text snippets, similar to QA-over-text systems. 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*. For the latter case, the top-ranked snippet talks about the composer duo *John Lennon and Paul McCartney*. When refining the query by qualifiers, this already puzzling situation becomes even more complex with *84.55 of 209 songs* being ranked first followed by varying counts such as *18 Beatles songs* (co-written with McCartney) and *61* (written separately). Because of the lack of consolidation, the onus is on the user to decide whether there are multiple correct answers across text segments.

In Mirza et al. (2018), it is reported that 5%-10% of queries in popular QA datasets are count queries. Current text-based QA systems and datasets largely ignore consolidation over multiple documents, since the target is to produce a single answer span or document. 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) and on KBs with datasets such as LC-QuAD 2.0 (Dubey et al., 2019), WebQuestions (Berant et al., 2013) and QALD (Usbeck et al., 2018). 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, but no further context. More recent KB-QA datasets, such as VQuAnDa (Kacupaj et al., 2020) and 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 evaluation metrics for reading-comprehension style benchmarks typically employ strict matching requirements, like measuring accuracy, F1-score and exact match (Zeng et al., 2020). On a question level, these metrics measure the token-level overlap, which does not transfer well to count

[2] https://w.wiki/4XVq

[3] http://qa.cs.washington.edu:2020

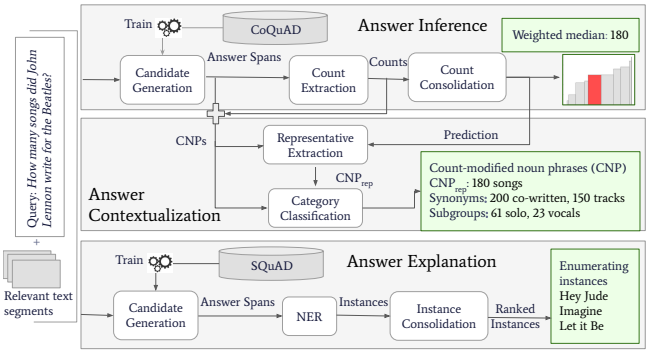

Figure 2: System overview of CoQEx from Ghosh et al. (2022).

queries, especially counts which do not have one authoritative answer and are an inference over multiple documents. We propose relaxed metrics for evaluation in Section 6.

It is recognized that just literally answering questions is often not sufficient for use cases. One line of work by Kacupaj et al. (2020, 2021), focuses on answer verbalizations of the formal SPARQL query responses. Kacupaj et al. (2020) tackles this by returning comprehensive answer in full sentences, using templates, and (Kacupaj et al., 2021) is a framework for generating verbalized answers given a natural language question, its logical form and the extracted answer. 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 ELI5 dataset by Fan et al. (2019) 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 evidenced on multiple documents, while we focus on answering count queries by consolidating counts and grounding them in instances.

3. Design Space and Architecture

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 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 explain the methodology of CoQEx in the next section.

4. 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.

4.1. Answer Inference

In order to generate count candidates, we use the popular SpanBERT model from 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`$(17)\rightarrow 17$, `float`$(17.0)\rightarrow 17.0$), followed by a dictionary based look-up for worded to integer conversion (*seventeen*$\rightarrow 17$) 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.

For example, for the candidate set $\{150_{0.9}, 160_{0.8}, 180_{0.4}, 180_{0.4}, 210_{0.3}\}$ (confidences as subscripts), most confident would output 150, most frequent and median would return 180, and the weighted median 160.

Algorithm 1 Extracting 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.

$C \leftarrow \{\}$ ▷ *Passed to Algorithm 2*
$WeightedC \leftarrow \{\}$ ▷ *Counts with confidence*
for $d \in D$ **do**
  $c \leftarrow$ SPANPREDICTION(d, q)
  if $c.Span \neq None$ *and* $c.Conf > \theta$ **then**
    $i \leftarrow$ EXTRACTCOUNT$(c.Span)$
    if $i \neq None$ **then**
      $C \leftarrow C \cup (c, i)$
      $WeightedC \leftarrow WeightedC \cup (i, c.Span)$
if $WeightedC \neq \{\}$ **then**
  $WeightedC \leftarrow$ SORTASCENDING$(WeightedC)$
  ▷ *Return the weighted median of the counts.* ◁
  $C_{pred} \leftarrow$ CONSOLIDATE$(WeightedC)$
else
  $C_{pred} \leftarrow Null$
return C_{pred}, C

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*, with a prediction 700, *estimated 700 languages* would be the CNP_{rep} since it has the highest confidence (see

Table 1
CNPs with their categories for a query and confidence scores as subscripts.

Query	*How many languages are spoken in Indonesia?*
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$_{(0.5)}$

Table 1. {*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 (Table 1).

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), so as 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}

```
1:  q' ← q.replace("how many", "which")
2:  I ← {}                          ▷ Inverted dictionary of instances.
3:  S ← {}                          ▷ Consolidated score for instances.
4:  for d ∈ D do
5:      c ← SpanPrediction(d, q')
6:      ▷ Get all instances from the span.                              ◁
7:      if c.Span ≠ None and c.Conf > θ then
8:          I_d ← NER(c.Span)
9:          for i ∈ I_d do
10:             I[i] ← I[i] ∪ (d, c)
11: for i ∈ I do
12:     S[i] ← ConScore(I[i], D)
13: I_Ranked ← SortDescending(I, key = lambda i : S[i])
14: return I_Ranked
```

5. The CoQuAD Dataset

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[4] to collect the query autocomplete suggestions for the above

[4] https://serpapi.com

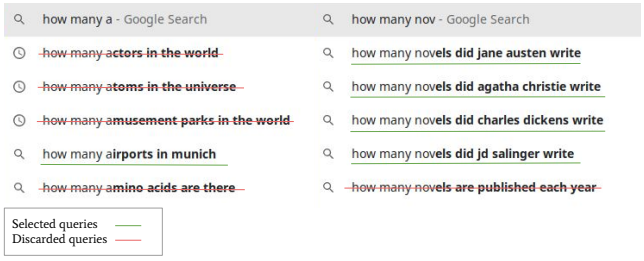

Figure 3: Query autocomplete suggestions for the query prefixes: *how many a* and *how many nov*. Queries with no named-entity mentions are discarded (here, red strike-through).

query prefixes and keep those with at least one named-entity (to avoid too general queries). Figure 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.

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 $\pm 10\%$ 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 both the test data, with the automated ground truths, and queries without any direct answers. 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.

Quality Assessment. Our evaluation data comprises around 69% of automated GT extractions from the Google KG and featured text snippets and 31% of manually annotated ground truths. We manually searched and wrote down ground truths for the 222 queries, to obtain insights into the accuracy of the automated ground truths. We gave ourselves access to the search-engine result page (see Figure 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 Mmachine 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 from a snippet which say *"There are over 900 T.G.I. Friday's branded restaurants in 60 countries worldwide … There are 566 restaurants in the U.S. … "*.

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?

Figure 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.

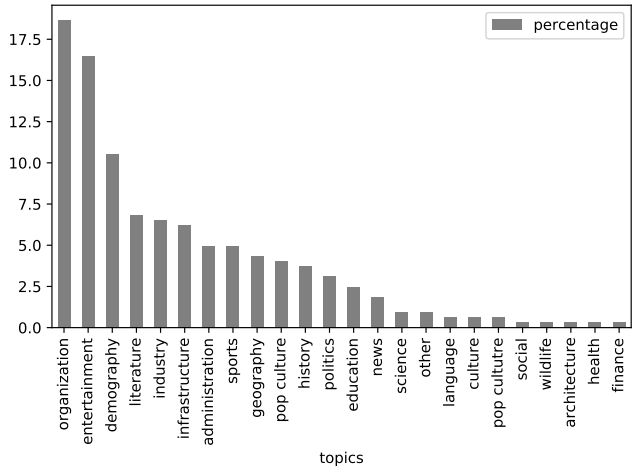

Figure 5: Distribution of topics in CoQuAD.

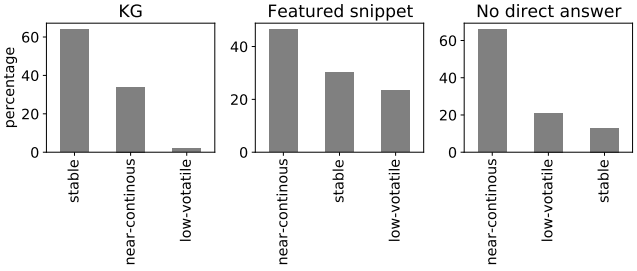

Figure 6: Time variance of CoQuAD queries by answer source.

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 process, we realize that there exists structure to the results provided by the search engines as illustrated in Figure 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 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 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 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.

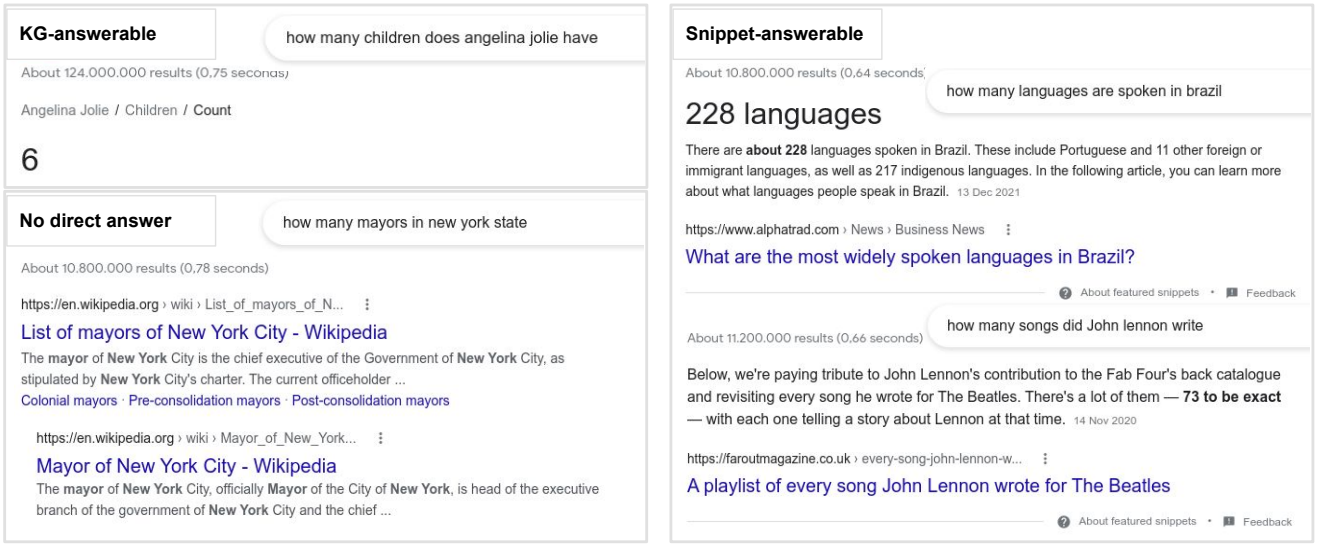

Figure 7: Nature of ground truths extracted. **KG-answerable** queries show the path (entity and relation) and aggregate used (Count). **Snippet-answerable** queries have a featured snippet with a highlighted answer displayed at the top of the snippet (228 languages) or within the snippet which then 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.

Since all KG-answerable and snippet-answerable queries have ground truth counts, we focus on the CNPs and instances. If we go by the conventional method of QA without any consolidation, and analyse the paths returned by KG answerable queries and the featured snippets, we find that KG-answerable queries are usually simply related and have rare occurrences of any semantic qualifiers (equivalent to CNPs) with 0 occurrences in the CoQuAD annotated queries. 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 is 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 9, 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 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 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[5] to obtain the query components as follows.

- *Named entity*: tokens were extracted if NER returned a label or if the tokens were tagged `propoer noun` by the POS tagger.

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

- *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 4.3 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, *sibling, movie, employee, nfl stadium, real estate agent, czech player*. The queries spanned over 49 different relations.
Our research data is made publicly available[6].

6. Experimental Setup

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.

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 systems 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.

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[7] twice - the original count query (for the count answer) and a modified variant as in Section 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.

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:

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

[7] QAnswer API at `https://qanswer-core1.univ-st-etienne.fr/swagger-ui.html`

Table 2
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**

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.

6.4. Implementation Details

The candidate generation steps for answer inference and explanation uses two instances of a span prediction model Joshi et al. (2020). 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[8]. The model for answer explanation is trained on SQuAD 2.0.

7. Results

7.1. Baseline Comparison on Answer Inference

Table 2 shows the answer inference performance of CoQEx against the baselines on different datasets. We improve upon Ghosh et al. (2022) by using a refined count extractor and also report the performance on a new dataset. The RP-Coverage trade-off metric highlights the advantages provided by CoQEx.

In both CoQuAD and Natural Questions datasets GoogleSDA 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.

7.2. Effect of Query Types on Answer Inference

In Table 3, we analyse 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.

7.3. 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 (*volcanic islands in Hawaii⇒islands→Hawaii→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 detailed

[8] https://spacy.io on the en_core_web_sm model.

Table 3
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**

Table 4
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

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 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 query equivalent 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.

7.4. Baseline Comparison on Answer Explanation

The results on instance-annotated CoQuAD and natural Questions dataset are in Table 4. 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 sub-section.

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 i) noise due to non-entity terms recognized as entities ii) 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*, iii) entities satisfying a more general criteria, for instance returning other airports from Vietnam when asked for airports in Ho Chi Minh City, iv) 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.

7.5. Answer Explanation by Query Type

We analyse the system performances on answer explanation by the query answerability: KG-answerable, snippet-answerable in Table 5. 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

Table 5
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**

Table 6
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 7
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**

page itself, either as an answer from its KG or in the form of direct answers (featured snippets).

7.6. 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 6. 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 7. 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 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.

8. Component analysis of CoQEx

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

8.1. Span Prediction Model for Answer Inference

We test the candidate generator for count spans on SpanBERT (Joshi et al., 2020) finetuned on i) CoQuAD and, ii) the popular general QA dataset SQuAD (Rajpurkar et al., 2016) on different span selection thresholds in Figure 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

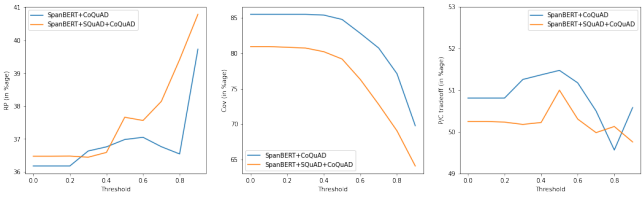

Figure 8: Performance of fine-tuned models on answer inference metrics, (from left to right) Relaxed Precision (RP), Coverage (Cov) and Relaxed Precision-Coverage trade-off (P/C) across different span selection thresholds.

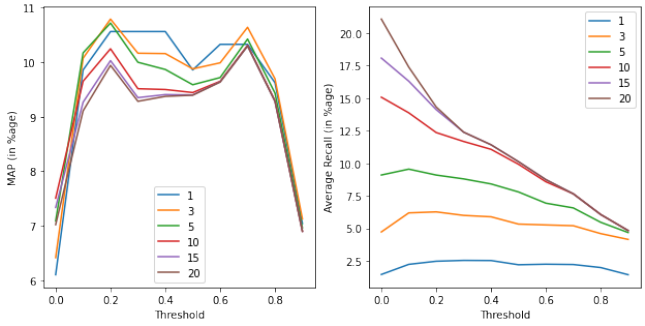

Figure 9: MAP and Average recall across threshold values and ranks. The values are averaged over consolidation alternatives.

Table 8
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

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.

8.2. 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 8). 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.

8.3. 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 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).

8.4. 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 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 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

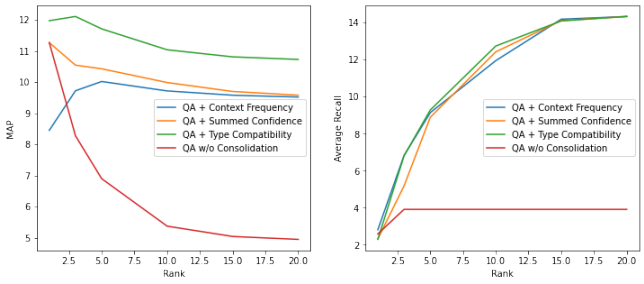

Figure 10: MAP and Average Recall of different consolidation strategies at ranks 1, 5 and 10 when span selection threshold=0.2.

Table 9
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

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.

8.5. Best Consolidation for Answer Explanations

The MAP and AR scores of different consolidation strategies are shown in Figure 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%).

9. Discussion

9.1. Distribution of CNPs and instances in snippets

In Section 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 9. 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.

9.2. 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 9). Thus indicating that relevant information is spread across contexts making our task of inferring answer and explanatory evidence from multiple sources a significant contribution.

9.3. 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 10.

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

Table 10
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$_{(0)}$

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 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 of 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 provides a relevant instance *Moti Maris* which is not present in the current KGs.

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.

Preliminary access to the system demonstrator is available at `https://nlcounqer.mpi-inf.mpg.de/`, where users can query pre-fetched snippets used in the CoQuAD dataset or make limited live queries dealing with counts.

9.4. Limitations and Future Work

Here we discuss major limitations of this study, and outline future research directions.

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 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 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.

Multilinguality. 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 4.1 and 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. 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).

10. Conclusion

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 analysed count queries, their prevalence, structure and how current state-of-the-art answer them. We provide a thorough analysis of our system components and how CoQEx compares to the baselines in different query settings. A thorough discussion on tackled and open challenges are provided in the discussion section (Section 9) 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 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.

To foster further research, we release all datasets[9] and provide access to the system demonstrator[10] to make queries and see results through an interactive interface.

References

K. Balog, Entity-Oriented Search, The Information Retrieval Series, Springer, 2018.

D. Diefenbach, V. López, K. D. Singh, P. Maret, Core techniques of question answering systems over knowledge bases: a survey (2018).

Z. Huang, S. Xu, M. Hu, X. Wang, J. Qiu, Y. Fu, Y. Zhao, Y. Peng, C. Wang, Recent trends in deep learning based open-domain textual question answering systems, in: IEEE Access, 2020.

R. Usbeck, M. Röder, M. Hoffmann, F. Conrads, J. Huthmann, A. N. Ngomo, C. Demmler, C. Unger, Benchmarking question answering systems, in: SWJ, 2019.

P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang, SQuAD: 100,000+ questions for machine comprehension of text, in: EMNLP, 2016.

T. Kwiatkowski, et al., Natural questions: A benchmark for question answering research, in: TACL, 2019.

M. Dubey, D. Banerjee, A. Abdelkawi, J. Lehmann, LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and DBpedia, in: ISWC, 2019.

E. M. Voorhees, Overview of the trec 2001 question answering track, in: TREC, 2001.

D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase (2014).

S. Ghosh, S. Razniewski, G. Weikum, Uncovering hidden semantics of set information in knowledge bases, in: JWS, 2020.

S. Ghosh, S. Razniewski, G. Weikum, Answering count queries with explanatory evidence, SIGIR (2022).

D. Diefenbach, P. H. Migliatti, O. Qawasmeh, V. Lully, K. Singh, P. Maret, QAnswer: A question answering prototype bridging the gap between a considerable part of the LOD cloud and end-users, in: WWW, 2019.

X. Lu, S. Pramanik, R. Saha Roy, A. Abujabal, Y. Wang, G. Weikum, Answering complex questions by joining multi-document evidence with quasi knowledge graphs, in: SIGIR, 2019.

K. Xu, Y. Feng, S. Huang, D. Zhao, Hybrid question answering over knowledge base and free text, in: COLING, 2016.

V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering, in: EMNLP, 2020.

M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, O. Levy, SpanBERT: Improving pre-training by representing and predicting spans, in: TACL, 2020.

V. Sanh, L. Debut, J. Chaumond, T. Wolf, DistilBERT, a distilled version of bert: smaller, faster, cheaper and lighter, in: EMC2, 2019.

D. Chen, A. Fisch, J. Weston, A. Bordes, Reading Wikipedia to answer open-domain questions, in: ACL, 2017.

D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, M. Gardner, DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs, in: NAACL-HLT, 2019.

S. Wang, M. Yu, J. Jiang, W. Zhang, X. Guo, S. Chang, Z. Wang, T. Klinger, G. Tesauro, M. Campbell, Evidence aggregation for answer re-ranking in open-domain question answering, in: ICLR, 2018.

R. Saha Roy, A. Anand, Question answering over curated and open web sources, in: SIGIR, 2020.

P. Mirza, S. Razniewski, F. Darari, G. Weikum, Enriching knowledge bases with counting quantifiers, in: ISWC, 2018.

S. Reddy, D. Chen, C. D. Manning, CoQA: A conversational question answering challenge, in: TACL, 2019.

M. Joshi, E. Choi, D. Weld, L. Zettlemoyer, TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension, in: ACL, 2017.

J. Berant, A. Chou, R. Frostig, P. Liang, Semantic parsing on Freebase from question-answer pairs, in: EMNLP, 2013.

R. Usbeck, R. Gusmita, M. Saleem, A.-C. Ngonga Ngomo, 9th challenge on question answering over linked data (QALD-9), in: ISWC, 2018.

E. Kacupaj, H. Zafar, J. Lehmann, M. Maleshkova, VQuAnDa: Verbalization question answering dataset, in: ESWC, 2020.

E. Kacupaj, B. Banerjee, K. Singh, J. Lehmann, Paraqa: A question answering dataset with paraphrase responses for single-turn conversation, in: ESWC, 2021.

L. Bauer, Y. Wang, M. Bansal, Commonsense for generative multi-hop question answering tasks, in: EMNLP, 2018.

C. Zeng, S. Li, Q. Li, J. Hu, J. Hu, A survey on machine reading comprehension—tasks, evaluation metrics and benchmark datasets, Applied Sciences 10 (2020) 7640.

E. Kacupaj, S. Premnadh, K. Singh, J. Lehmann, M. Maleshkova, Vogue: Answer verbalization through multi-task learning, in: ECML/PKDD, 2021.

K. Krishna, A. Roy, M. Iyyer, Hurdles to progress in long-form question answering, in: NAACL, 2021.

A. Fan, Y. Jernite, E. Perez, D. Grangier, J. Weston, M. Auli, Eli5: Long form question answering, in: ACL, 2019.

F. Zhu, W. Lei, C. Wang, J. Zheng, S. Poria, T.-S. Chua, Retrieving and reading: A comprehensive survey on open-domain question answering, in: arXiv, 2021.

S. Roy, T. Vieira, D. Roth, Reasoning about quantities in natural language, in: TACL, 2015.

N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: EMNLP, 2019.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: A robustly optimized bert pretraining approach, in: arXiv, 2019.

D. Sullivan, How google autocomplete predictions are generated, 2020. URL: https://blog.google/products/search/how-google-autocomplete-predictions-work/.

J. Romero, S. Razniewski, K. Pal, J. Z. Pan, A. Sakhadeo, G. Weikum, Commonsense properties from query logs and question answering forums, in: CIKM, 2019.

G. Navarro, A guided tour to approximate string matching (2001).

T. Wolfson, M. Geva, A. Gupta, M. Gardner, Y. Goldberg, D. Deutch, J. Berant, Break it down: A question understanding benchmark, Transactions of the Association for Computational Linguistics 8 (2020) 183–198.

V. T. Ho, K. Pal, N. Kleer, K. Berberich, G. Weikum, Entities with quantities: Extraction, search, and ranking, Proceedings of the 13th International Conference on Web Search and Data Mining (2020).

E. Choi, O. Levy, Y. Choi, L. Zettlemoyer, Ultra-fine entity typing, in: ACL, 2018.

Y. Onoe, G. Durrett, Fine-grained entity typing for domain independent entity linking, in: AAAI, 2020.

S. Ruder, A. Sil, Multi-domain multilingual question answering, Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts (2021).

C. P. Carrino, M. R. Costa-jussà, J. A. R. Fonollosa, Automatic spanish translation of squad dataset for multi-lingual question answering, in: LREC, 2020.

K. V. Nguyen, D.-V. Nguyen, A. G.-T. Nguyen, N. L.-T. Nguyen, A vietnamese dataset for evaluating machine reading comprehension, in: COLING, 2020.

A. Rahimi, Y. Li, T. Cohn, Massively multilingual transfer for ner, in: ACL, 2019.

S. Tedeschi, V. Maiorca, N. Campolungo, F. Cecconi, R. Navigli, Wikineural: Combined neural and knowledge-based silver data creation for multilingual ner, in: EMNLP, 2021.

D. A. Ferrucci, This is Watson, IBM Journal of Research and Development (2012).

P. Rajpurkar, R. Jia, P. Liang, Know what you don't know: Unanswerable questions for squad, in: ACL, 2018.

[9] https://github.com/ghoshs/CoQEx

[10] https://nlcounqer.mpi-inf.mpg.de/