Answering Count Queries with Explanatory Evidence

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

A challenging case in web search and question answering are count queries, 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 show the benefits of our method. To promote further research on this underexplored topic, we release an annotated dataset of 5k queries with 200k relevant text spans.

CCS CONCEPTS

• **Information systems → Question answering**.

KEYWORDS

Question Answering, Count Queries, Explainable AI

ACM Reference Format:
Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. 2022. Answering Count Queries with Explanatory Evidence. In *Proceedings of the 45th Int'l ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3477495.3531870

1 INTRODUCTION

Motivation and Problem. Question answering (QA) and web search with telegraphic queries have been greatly advanced over the last decade [1, 4, 10, 25]. 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 [7, 15, 19, 26]. If the required data is in a structured knowledge base (KB) such as Wikidata [27], 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. Besides, evaluating the additional condition *"for the Beatles"* (i.e., a subset of his songs) is beyond their scope.

Search engines 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. 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 she may have only popular songs in mind. It is, therefore, crucial to provide users with explanatory evidence.

Approach and Contribution. This paper presents CoQEx, Count Question answering with Explanatory evidence, which answers count queries via three components: i) *answer inference* ii) *answer contextualization* and, iii) *answer explanation.*

Given a full-fledged question or telegraphic query and relevant text passages, 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.
Novel 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;

SIGIR '22, July 11–15, 2022, Madrid, Spain.

ACM ISBN 978-1-4503-8732-3/22/07.
https://doi.org/10.1145/3477495.3531870

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 passages, with preliminary access for reviewing at https://github.com/ghoshs/CoQEx.

2 RELATED WORK

Where structured data is available in KBs, structured QA is the method of choice. However, for many topics, no relevant count information can be found in KBs. For example, Wikidata contains 217 songs attributed to John Lennon[1], but is incomplete in indicating whether these written for the Beatles or otherwise. Previous analyses found that 5-10% of queries in popular QA datasets are count queries [18]. In the KB-QA domain, systems like QAnswer [5] tackle count queries by aggregating instances using the SPARQL `count` modifier. This is liable to incorrect answers, when instance relations are incomplete. Recent research has also explored QA over web tables, with a focus on table retrieval and span prediction [9]. Another challenging aspect of KB-QA is reducing the semantic gap between natural language query and the SPARQL query formulation [2], which would in turn lead to better count query answering.

State-of-the-art systems typically approach QA via the machine reading paradigm [3, 6, 11, 13, 23], where the systems find the best answer in a given passage. The retriever-reader approach in open-domain QA uses several text segments to return either a single best answer [3, 28] or a ranked list of documents with the best answer per document [13]. The DPR system [13][2] returns *"approximately 180"* from its rank-1 text passage 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 [17, 29]. Search engines can answer simple count queries from their underlying KBs, if present, a trait which we exploit to create our CoQuAD dataset (Section. 4). 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.

[1]https://w.wiki/4XVq
[2]http://qa.cs.washington.edu:2020

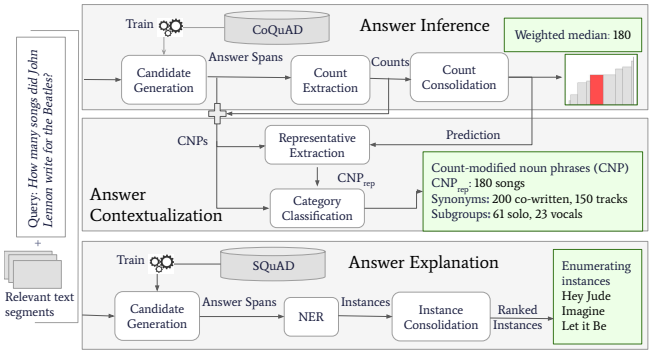

Figure 1: System overview of CoQEx.

It is recognized that just literally answering questions is often not sufficient for use cases. One line of work tackles this by returning comprehensive answer in full sentences, using templates [12]. Another line concerns long form question answering, where the QA model retrieves multiple relevant documents to generate a whole answer paragraph [14]. The ELI5 dataset [8] 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 METHODOLOGY

We approach count question answering by a combination of per-document answer span prediction, context extraction, and consolidation of counts and instances across documents. Fig. 1 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 focus on extracting counts and instances (entity-mentions) from the text segments so as to 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.

Answer Inference. For obtaining answer candidates, we use the popular SpanBERT model [11], trained on the CoQuAD train split for candidate generation. 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). These answer spans have two components - the count itself and qualifiers, which we separate with the help of the CogComp Quantifier [22]. To consolidate the resulting candidate counts, we compare four methods:

1. *Most confident:* The candidate given the highest confidence by the neural model. This is commonly used in textual QA [3, 28].
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.

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. 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 within $\pm\alpha$ of the predicted count having the highest confidence as CNP_{rep}, where α is between 0 and 100%, 0 being most restrictive. The remaining CNPs are categorized as follows:

1. *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}.
2. *Synonyms*: CNPs, whose meaning is highly similar to CNP_{rep}.
3. *Incomparables*: CNPs which count instances of a completely different type.

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 [20] 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 "How many languages are spoken in Indonesia", with a prediction 700, *estimated 700 languages* would be the $\textit{CNP}_{rep}$, {*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.

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.

We again use the SpanBERT model to obtain candidates, 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. We extract named entities from the answer spans and rank them using the following alternative approaches:

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.
3. *QA + Summed Confidence.* We rank the instances based on the summed confidence of all answer spans that contain them.
4. *QA + Type Compatibility.* Here instances are ranked by their compatibility with the query's answer type, extracted via the dependency parse tree. 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 [16] to obtain entailment scores, which are again summed over all containing answer spans.

4 THE CoQuAD DATASET

Dataset construction. Existing QA datasets only incidentally contain count queries; we leverage search engine autocomplete suggestions to automatically compile count queries that reflect real user queries [24]. We provide the Google search engine with iterative query prefixes of the form *"How many X"*, where $X \in \{a, aa, \ldots, zzz\}$, similar to the candidate generation from patterns used in [21], and collect all autocomplete suggestions via SERP API[3]. We keep those with at least one named-entity (to avoid too general queries) and no measurement term (to avoid non-entity answer types). This gives us 11.3k count queries.

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 [23] to obtain best answers from the latter. This gives us KG ground truth for 131 queries, and ground truth from featured snippets for 6.7k queries. We again discard queries whose automated ground truth answer contains a measurement term, and manually annotate 100 queries from those without automated ground truth.

We next scrape the top-50 snippets per query from Bing, and obtain *text segment ground truth* by labelling answer spans returned by the count extractor [22] as positive when the count lies within ±10% from the ground truth. There are around 800 queries with no positive snippets, which we do not discard, so the system is not forced to generate an answer. In the end we have 5162 count queries with automated ground truth, and an average of 40 annotated text segments per query. We use 80% of the count queries with automated ground truth for training. The test set contains 50 count queries with KG ground truth, 100 with ground truths from featured snippets, and 100 with manually annotated ground truth for quantitative and qualitative analysis. We also manually annotate 75 queries with at least top-5 prominent instances for evaluating answer explanations.

Dataset Characteristics. Queries in CoQuAD cover a range of topics, notably entertainment (27%), social topics (20%), organizations (12%), technology (8%) and politics (7%). We find that 20% of query results are fully stable (a company's founders, casts in produced movies), 55% are low-volatile (lakes in a region, band members of an established but active band), 25% are near-continuous (employment numbers, COVID cases). Most queries count entities in a simple relation to one named entity, i.e., the avg. query length is 6.40 words, with an average of 1.08 named entities per query.

[3] https://serpapi.com

Research access to the data is at https://tinyurl.com/countqueryappendix.

5 EVALUATION

Implementation Details. The transformer model for answer inference candidate generation is trained for 2 epochs, at a learning rate of $3e^{-5}$. An input datapoint 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. For getting the instances from the answer spans, we use the pre-trained SpaCy NER model[4].

Component analysis of CoQEx. We evaluate the CoQEx components to determine the best configurations for answer inference, consolidation and explanation. While we can use regular IR metrics of precision and recall for evaluating answer explanations (Precision@k, Recall@k, Hit@k and MRR) and accuracy of classification for evaluating count context categories, we need a new metric for counts. We use *Relaxed Precision* (RP), which is the fraction of queries where the prediction lies within ±10% of the gold answer. We also report *Coverage*, which measures the fraction of queries that a systems returns an answer for.

We test the candidate generator for count spans on SpanBERT [11] finetuned on i) CoQuAD and, ii) the popular general QA dataset SQuAD [19]. *Fine-tuning* on SQuAD gives slightly higher precision scores (29.8% vs. 27.4% RP), but CoQuAD gives a higher coverage (82% vs. 73.8%) resulting in overall more correctly answered queries.

For *answer inference*, the *frequent* and *weighted median* consolidation schemes outperform the others, with *weighted median* achieving 27.4% RP just 0.6% ahead of *frequent*. 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.

We assess the classification accuracy of CNPs for a manually labelled sample of 294 CNPs for 64 queries. While a strict threshold of 10% ensures a high accuracy of 90% for *Synonyms*, which only decreases with increasing α, the accuracy of *Subgroups* and *Incomparables* is initially low (at or less than 50%), and peaks only at a much higher α. A weighted optimum is reached at $\alpha = 30\%$, where the accuracy of the *Synonyms* does not degrade much (89%), and the accuracy of *Subgroups* and *Incomparables* is both above 60% (61% and 62% respectively).

In *answer explanation*, *QA + context frequency* consolidation performs consistently well. While *QA w/o consolidation* has highest P (11.5%) and R (2.3%) at rank 1, its performance decreases subsequently to only 2.7% P and 5.1% R at rank 10, compared to 10.8% P and 18.2% R of *QA + context frequency*. This indicates that the QA model is tailored to the typical setting QA of a single correct answer, and that consolidation helps beyond that.

Comparing CoEx with baselines. We compare our proposed system with two complementary paradigms.

1. Knowledge-base question answering: QAnswer [5].
2. Commercial search engine QA: Google Search Direct Answers (GoogleSDA).

For fairness to QAnswer, we post-processed the results to extract count and instances. For evaluating instances by GoogleSDA, we post-processed knowledge graph and featured snippet of the search engine result page, keeping items from list-like structures as instances ranked in their order of appearance.

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

Table 1: Comparing answer inference results (in percentages).

System	CoQuAD		LCQuAD$_{count}$		Stresstest	
	RP	Cov	RP	Cov	RP	Cov
QAnswer [5]	6.6	**95.6**	**29.8**	**50.0**	3.0	45.0
GoogleSDA	**84.6**	60.0	5.7	10.5	22.0	37.0
CoQEx	27.4	82.0	6.7	45.2	**38.6**	**88.3**

Table 2: Precision@k (P@k), Recall@k (R@k), Hit@10 and MRR for the answer explanations of CoQEx and baselines.

System	P@1	P@5	P@10	R@1	R@5	R@10	Hit@10	MRR
QAnswer [5]	4.7	4.5	3.7	**2.1**	0.8	1.1	12.8	0.064
GoogleSDA	**7.7**	**14.2**	**21.7**	1.3	2.2	3.2	12.8	0.089
CoQEx	6.4	12.8	10.8	1.3	**11.7**	**18.2**	**55.1**	**0.218**

Table 3: Extrinsic user study (precision in percent).

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**

For *answer inference*, Table 1 compares the three systems on the 250 annotated test CoQuAD queries. We also present the performances on 100 count queries from an existing dataset LCQuAD [7], and a manually curated dataset of 100 challenging count queries called *Stresstest*. While GoogleSDA has a high precision on CoQuAD (consisting 150 KG and snippet answerable queries), CoQEx not only provides high coverage but a decent RP. On LCQuAD, a dataset designed specifically for KG queries, CoQEx loses to LCQuAD while still maintaining a better coverage and RP score compared to GoogleSDA. The results indicate that reliance on structured KBs (QAnswer) is not sufficient for general queries, and robust consolidation from crisper text segments is necessary.

For *answer explanation*, the comparison results on the 75 instance-annotated CoQuAD queries are in Table 2. CoQEx provides more R than the baselines, with competitive P at rank 1, and better at ranks 5 and 10. CoQEx performs consistently better at hits@k and MRR, losing only slightly in precision at ranks 1 and 5. Both baselines answer less than 25 queries at rank 5, and at rank 10 less than 20 queries, and QAnswer performs poorly in the returned answers. GoogleSDA operates extremely conservative, thus maintaining precision at lower ranks, at the cost of tiny recall.

Qualitative Comparison. We looked further into the 250 queries (50 KG answerable, 100 snippet answerable and 100 with ranked snippets) for understanding how baselines tackle progressively difficult queries. As explained in the CoQuAD dataset creation, we encountered three ways in which Google answers count queries. Around 1% of the answers came from the Google KG, 59% through featured snippets, 40% through page results. Besides these high-level categories, answers can be categorized by the following aspects:

1. Listing only instances (e.g., for "mayors of New York"),
2. Listing only counts (e.g., for "employees of NHS"),
3. Listing both instances and counts, or

4. Listing counts refined with semantic qualifiers (e.g., "7 official languages and 30 regional languagues").

On the above 250 queries, QAnswer returns mostly counts (95.6%) and rarely instances (2%) or no answers (2.4%). Among the KG-based answers, GoogleSDA returns both counts and instances for 90% of the queries, and only counts for the remaining 10%. Among the featured-snippets-based answers, 85% contain only counts, and just 15% contain both counts and instances. Semantic qualifiers are common in featured snippets, coming up for 73% of queries. While semantic qualifiers can be expressed in KG answers ("volcanic islands in Hawaii"⇒islands→Hawaii→volcanoes), this rarely shows up, unless the queried entity and the qualifier are extremely popular.

User Studies. We asked 120 MTurk users for pairwise preferences between answer pages that reported bare counts, and counts enhanced by either of the explanation types. 50% of participants preferred interfaces with CNPs, 80% with a snippet, 73% with instances, 63% preferred an interface with all three enabled (remaining percentage: prefer bare count answer/same/cannot decide). While snippets are already in use in search engines, the results indicate that CNPs and instances are considered valuable, too.

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 3. All explanation had a positive effect on overall annotator precision, especially for incorrect counts.

6 CONCLUSION

In this work we highlighted the distinct challenges of count query answering, in particular, that count answer candidates form numeric distributions, that answer contexts stand in hierarchical relations, and that counts coexist with instances, that ideally should be returned as well. We introduced an approach for count query answering that contains these three components.

We construct the CoQEx dataset of count queries that reflect real user needs. Through CoQEx, we show how to provide a better trade-off between precision and coverage than the current deployed systems, on multiple challenging datasets. User studies show that explanatory evidence improves user comprehension, especially through CNPs when there exist similar competing answers attributed to specific qualifiers. To foster further research, we release all datasets[5].

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable feedback and suggestions.

REFERENCES

[1] Krisztian Balog. 2018. *Entity-Oriented Search.* Springer.
[2] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*.
[3] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *ACL*.
[4] Dennis Diefenbach, Vanessa López, Kamal Deep Singh, and Pierre Maret. 2018. Core techniques of question answering systems over knowledge bases: a survey. In *Knowl. Inf. Syst.*
[5] Dennis Diefenbach, Pedro Henrique Migliatti, Omar Qawasmeh, Vincent Lully, Kamal Singh, and Pierre Maret. 2019. QAnswer: A Question Answering prototype bridging the gap between a considerable part of the LOD cloud and end-users. In *WWW*.
[6] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In *NAACL-HLT*.
[7] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and DBpedia. In *ISWC*.
[8] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long Form Question Answering. In *ACL*.
[9] Jonathan Herzig, Thomas Mueller, Syrine Krichene, and Julian Eisenschlos. 2021. Open Domain Question Answering over Tables via Dense Retrieval. In *NAACL*.
[10] Zhen Huang, Shiyi Xu, Minghao Hu, Xinyi Wang, Jinyan Qiu, Yongquan Fu, Yuncai Zhao, Yuxing Peng, and Changjian Wang. 2020. Recent Trends in Deep Learning Based Open-Domain Textual Question Answering Systems. In *IEEE Access*.
[11] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. In *TACL*.
[12] Endri Kacupaj, Hamid Zafar, Jens Lehmann, and Maria Maleshkova. 2020. VQuAnDa: Verbalization question answering dataset. In *ESWC*.
[13] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *EMNLP*.
[14] Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to Progress in Long-form Question Answering. In *NAACL*.
[15] Tom Kwiatkowski et al. 2019. Natural Questions: A Benchmark for Question Answering Research. In *TACL*.
[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized bert pretraining approach. In *arXiv*.
[17] Xiaolu Lu, Soumajit Pramanik, Rishiraj Saha Roy, Abdalghani Abujabal, Yafang Wang, and Gerhard Weikum. 2019. Answering complex questions by joining multi-document evidence with quasi knowledge graphs. In *SIGIR*.
[18] Paramita Mirza, Simon Razniewski, Fariz Darari, and Gerhard Weikum. 2018. Enriching Knowledge Bases with Counting Quantifiers. In *ISWC*.
[19] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*.
[20] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP*.
[21] Julien Romero, Simon Razniewski, Koninika Pal, Jeff Z. Pan, Archit Sakhadeo, and Gerhard Weikum. 2019. Commonsense properties from query logs and question answering forums. In *CIKM*.
[22] Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about Quantities in Natural Language. In *TACL*.
[23] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *EMC2*.
[24] Danny Sullivan. 2020. How Google autocomplete predictions are generated. https://blog.google/products/search/how-google-autocomplete-predictions-work/
[25] Ricardo Usbeck, Michael Röder, Michael Hoffmann, Felix Conrads, Jonathan Huthmann, Axel-Cyrille Ngonga Ngomo, Christian Demmler, and Christina Unger. 2019. Benchmarking question answering systems. In *SWJ*.
[26] Ellen M Voorhees. 2001. Overview of the TREC 2001 question answering track. In *TREC*.
[27] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. In *Communications of the ACM*.
[28] Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018. Evidence Aggregation for Answer Re-Ranking in Open-Domain Question Answering. In *ICLR*.
[29] Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Hybrid question answering over knowledge base and free text. In *COLING*.

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