CardiO: Predicting Cardinality from Online Sources

Shrestha Ghosh
MPI for Informatics, SIC
Saarbruecken, Germany
ghoshs@mpi-inf.mpg.de

Simon Razniewski
Bosch Center for AI
Renningen, Germany
raz2rng@bosch.com

Damien Graux
Trinity College Dublin
Dublin, Ireland
grauxd@tcd.ie

Gerhard Weikum
MPI for Informatics
Saarbruecken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

Count questions are an important type of information need, though often present in noisy, contradictory, or semantically not fully aligned form on the Web. In this work, we propose CardiO, a lightweight and modular framework for searching entity counts on the Web. CardiO extracts all counts from a set of relevant Web snippets, and infers the most central count based on semantic and numeric distances from other candidates. In the absence of supporting evidence, the system relies on peer sets of similar size, to provide an estimate. Experiments show that CardiO can produce accurate and traceable counts better than small LLM-only methods. Although larger models have higher precision, when used to enhance CardiO components, they do not contribute to the final precision or recall.

CCS CONCEPTS

• Computing methodologies → Information extraction; **• Information systems → Question answering**.

KEYWORDS

count knowledge, cardinality, entity sets, web question answering

ACM Reference Format:
Shrestha Ghosh, Simon Razniewski, Damien Graux, and Gerhard Weikum. 2024. CardiO: Predicting Cardinality from Online Sources. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3589335.3651477

1 INTRODUCTION

Queries about the number of entities with a certain property (sets), such as the number of Physics Nobel laureates, or the number of lakes worldwide (see questions below), are important in many knowledge-intensive use cases, and form ~10% of popular QA datasets [15]. Such queries can cover a wide spectrum, from very well-defined sets (like the Nobels) to overly vague ones (like lakes).

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

In principle, there are three ways to answer these, (i) from knowledge bases (KB)s, (ii) via texts, or (iii) via large language models (LLMs). While well-defined sets are easier to query in KBs, low recall and popularity biases in KBs hinder their usage in general settings [17]. Text extraction alleviates some of these problems, as it is more tolerant to fuzzy matches. Meanwhile, large corpora allow many more than one match, and so the challenges of ranking and aggregation arise. A recent third paradigm is LLMs, mixing extractive and predictive paradigms. However, they provide poor insights into their reasoning, and are notorious for inventing answers [24].

Approach. We propose CardiO (**Cardi**nality predictor from **O**nline sources), a lightweight and modular framework for answering count questions over the Web (Fig.1). Given a query, we first identify the *question components*. Next, we retrieve top search snippets and apply a *relevance filter* to obtain sentences with possible count candidates. We then *extract count representations*, the integer values and answer types and compute the type-level, sentence-level and snippet-level relevance scores. Next, we find the most probable prediction via *aggregation*. Finally, we perform *peer calibration* through *reformulations* for empty or low-scoring predictions.

Contribution. CardiO provides explainable count prediction by combining the relevance of a count and its surrounding context to the input question with supporting evidence from multiple snippets. We contrast CardiO with previous systems, like CoQEx [7], which has high coverage but is more prone to noise, and recent LLMs like GPT3.5 and Llama2, which have high precision, but low explainability. Our lightweight CardiO beats Llama and, while large GPT models win in precision and recall, we show that they do not provide an additional advantage in CardiO. Our dataset and code is available at https://doi.org/10.5281/zenodo.10803727.

2 RELATED WORK

Count questions form almost 10% of QA datasets [15], but most research is focused on quantity representation [22]. Previously modeled together with quantities, *i.e.,* numbers with measurement units [20], counts are identified as a separate class of values [25] by popular NLP pipelines like SpaCy. In the Semantic Web field, count knowledge has been studied via cardinality constraints for n-ary relations [5, 21], and in class completeness estimation [14].

Web question answering has evolved from statistical NLP techniques used to extract crisp answers from retrieved passages [4, 16] to newer class LLMs under the retriever-reader paradigm [9, 12]. Even though benchmark Web question datasets are derived from user questions on the Web [1, 11, 13], the domain is restricted to Wikipedia. Current search engines provide much varied answers, making the case for answering count questions more challenging.

Previous work has highlighted the importance of traceability for user comprehension [7]. However, tracing, such as displaying the path of a search engine's internal KB, is available for less than 10% of user questions [3, 7]. In the case of counts, enumerating the set is most transparent but not most efficient, especially with low entity recall for bigger and less popular sets.

WWW '24 Companion, May 13–17, 2024, Singapore, Singapore
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0172-6/24/05
https://doi.org/10.1145/3589335.3651477

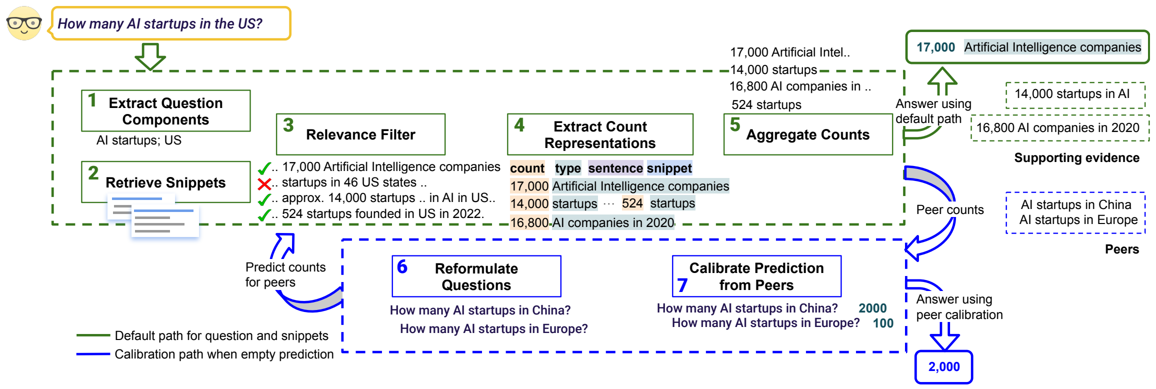
How many AI startups in the US?
1 Extract Question Components
AI startups; US
2 Retrieve Snippets
3 Relevance Filter
✓.. 17,000 Artificial Intelligence companies
✗.. startups in 46 US states ..
✓.. approx. 14,000 startups .. in AI in US..
✓.. 524 startups founded in US in 2022.
4 Extract Count Representations
count type sentence snippet
17,000 Artificial Intelligence companies
14,000 startups ··· 524 startups
16,800 AI companies in 2020
5 Aggregate Counts
17,000 Artificial Intel..
14,000 startups
16,800 AI companies in ..
524 startups
Answer using default path
17,000 Artificial Intelligence companies
14,000 startups in AI
16,800 AI companies in 2020
Supporting evidence
Peer counts
AI startups in China
AI startups in Europe
Peers
Predict counts for peers
6 Reformulate Questions
How many AI startups in China?
How many AI startups in Europe?
7 Calibrate Prediction from Peers
How many AI startups in China? 2000
How many AI startups in Europe? 100
Answer using peer calibration
2,000
Default path for question and snippets
Calibration path when empty prediction

Figure 1: A count question with traceable answers, using CardiO for the default path and when peer calibration is applied.

Conventionally, the motivation for reformulating questions has been to provide clarification to the original question [2, 19]. We apply question reformulation to increase the chances of estimating the cardinality of the queried set by identifying its peers. While specification and generalization are valid moves to obtain lower and upper bounds, respectively; evaluating the precision of bounds is hard and out of our scope here.

3 CARDIO FRAMEWORK

Count information in text is usually accompanied by a context that informs the type of entities being counted ("*songs*", "*solo songs*", "*singles*"). We therefore extend the notion of answer type to define the *type of entities being counted*, instead of a number. Similar to [7], we use search-engine snippets as a pragmatic approach to access diverse information. Given a question, we use the Bing API to retrieve the top 50 snippets. However, the only measurable metric directly available is the rank of the snippets.

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

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

Relevance Filter. We spot sentences having cardinal mentions and compute the relevance of each snippet and count-containing sentence to the question from the cosine similarity of the L_2 normalized embedding vectors of the snippet and the question. The embedding vectors come from a sentence transformer encoding model [18][1].

Count Representation Extraction. In order to be able to perform numerical operations, we extract the integer value of the counts, and the type being counted. We use the SpaCy pipeline to identify all noun phrases containing counts, extract the quantity using Quantulum[2], and the remaining text is the type. We score the relevance of the count type to the answer type identified in the question using the same embedding technique used for the snippet and sentence relevance scoring.

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

Max relevance: The count with the highest relevance score, $R(c)$, is the final answer.

Consistent: Counts with supporting evidence of similar counts from other snippets are highly scored. The final score of a count (Eq. 1) is the weighted sum of its relevance score, and the consistency score R_{cons} (the average score of the k nearest neighbors). This is similar to the consistency-based re-scoring of facts [10].

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

Central: Using Eq. 2, the *centrality* of a count is measured based on its distance from other counts. The closer the relevant counts, the higher is the centrality of a particular count.

$$C(c) = (N - 1)\left(\sum_{\forall c' \neq c} D(c, c')\right)^{-1} \times Rel(c_{sentence}\,,\ q) \quad (2)$$

Both consistent and central aggregations measure the distance between count representations using Eq. 4, which is a weighted average of the cosine distance between the sentences and the absolute order of magnitude difference between the counts (Eq. 3).

$$D_{order}(c, c') = \log_{10}\left(\frac{max(c, c')}{min(c, c')}\right) \quad (3)$$

$$D(c, c') = \alpha D_{sentence}(c, c') + (1 - \alpha)min(1, D_{order}(c, c')) \quad (4)$$

[1]We use *multi-qa-mpnet-base-cos-v1*.

[2]https://github.com/nielstron/quantulum3

Question Reformulation. We avoid a lookup on popular taxonomies such as WordNet or Wikipedia categories. While the WordNet taxonomy is very restrictive due to canonicalization, leading to poor matches between surface forms, the Wikipedia categories are very extensive, especially for large classes. In this case, harnessing LLM's ability to form semantic associations can be very helpful in returning a handful of most relevant reformulations. We use ChatGPT (version `gpt-3.5-turbo-0613`.) to reformulate a count question, when a named-entity is present, asking the LLM to replace the entity with other ones comparable in terms of the answer type. For example, reformulations for the number of *films produced by Warner Bros* are film produced by Walt Disney Pictures, Sony Pictures, and the 20th Century Studios.

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

LLM-Enhancements. LLMs generate better results when asked to generate reasoning steps [24]. However, evaluating such generations is challenging [6]. We apply LLM generations in CardiO in a more controlled and structured format and observe the changes in the performance metrics. In addition to (i) generating *reformulations* for peer calibration as explained in Sec. 3, we prompt LLMs to (ii) tag sentences with counts relevant to the user question and (iii) extract structured count representations. The goal is to improve precision by removing non-matching cardinal mentions early on. For this, we use LLMs to tag sentences having counts related to the question. The model is prompted with the question, a snippet name and the corresponding snippet and asked if a given sentence from the snippet has the answer to the question. Further, we extract structured count representations by prompting LLMs with the question, a snippet, and a sentence containing one or more cardinals to return a structured representation comprising the text span, the quantity, the type and, a bound (lower, upper, equal, approximate).

4 CARDINALITY BENCHMARKS

Few question-answering benchmarks specifically identify count questions. Their answers are reading-comprehension style as text spans. We use the **CoQuAD** test set comprising 312 count questions [7]. These are short entity-specific count questions extracted from search-engine query logs. We report on the 84 count questions from **Natural Questions (NQ)** [13] extracted by [7]. We create the **Cardinality Questions (CQ)** benchmark comprising 500 count questions, where we can control the type of counts that can be encountered in the Web. We build our data set on the 90 classes identified in [8]. Ground truth annotations are accompanied by additional labels, which describe whether we have the exact ground truth or an estimate, whether the ground truths are directly available on websites or aggregated manually, whether the entity set is popular or not, and whether the entity set is specific or not.

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

- **Exact Precision (EP)** For a given question, *EP* measures if the predicted count matches the ground truth count exactly.
- **Order of Magnitude Precision (OMP)** For a given question, it measures if the predicted count is within 1 order of magnitude of the ground truth and by how much (Eq. 5). We calculate the order difference first using Eq. 3. If the prediction is within one order difference, it returns the degree to which the prediction deviates from the ground truth between [0, 1], else 0.

$$OMP_q = 1.0 - min(1, D_{order}(c_{prediction}, c_{ground\ truth}) \tag{5}$$

- **Order of Magnitude Recall (OMR)** It penalizes empty predictions in the presence of ground-truth answers by returning 0. Non-empty predictions are measured using Eq. 5.

5 EVALUATION

Baselines. We evaluate CardiO (vanilla), and subsequently report the effect of LLM enhancements, and rich aggregations. We compare CardiO with CoQEx [7], open and semi-open LLMs like Llama2 (70B, 7B-chat, 70B-chat) [23], and proprietary LLMs like GPT3.5[3]. The LLMs are tested with 0-shot and snippet-augmented 0-shot prompts. In both cases, they are prompted to return a number and a one-line explanation. We then apply our count extractor to get the integer value. We also tested for answers returned in a specified JSON format to avoid the count extractor. Since this did not always help, we report only on the performance of the free-form answers.

Parameter setting. We perform a 5-fold cross-validation and determine the best value for the parameters $\alpha \in [0, 1]$ in Eq. 4, $\beta \in [0, 1]$ in Eq. 1 and the number of neighbors k to consider for R_{cons}, which are then used in other benchmarks. We report the standard deviation of the *central* and *consistent* aggregation methods in Table 1.

Baseline comparison. In Table 1, we notice a gap in the OMP scores of the generative and other lightweight models. Even CoQEx, which uses a supervised masked LM, is almost 0.18 points behind GPT3.5 when augmented with snippets. Our unsupervised CardiO (vanilla) (i) beats the 0-shot models in most settings, (ii) is better than the snippet-augmented Llama2-7B-chat model, and (iii) has a higher OMR than the larger Llama2-70B model. The difference between 0-shot and snippet-augmented LLMs is much higher than that between the pretrained and finetuned Llama models, emphasizing the importance of the snippets. Even so, the snippet-augmented models lack traceability to source snippet, characterizing the count type, providing bound and peer sets provided by CardiO.

Lack of traceability baselines. In the 0-shot mode, the LLMs return one-line claims, which require additional verification. When asked *"how many uninhabited islands in Sweden?"*, GPT3.5 returns: ***0** - Sweden does not have any uninhabited islands*, and Llama2 models return: ***221 uninhabited islands** in Sweden (Source: Swedish Agency for Marine and Water Management)*[4]. With snippets, Llama2-7B-chat just returns a number ***267,570***, GPT3.5 returns: *There are over **267,570 islands** in Sweden, with fewer than 1000 of them being*

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

[4]Note that the citation is itself generated, and can, but need not be correct.

Table 1: Performance on cardinality benchmarks, grouped by answer traceability.

Answer Traceability	Method	CQ Benchmark (n=500)			NQ (n=84) [13]			CoQuAD (n=312) [7]		
		EP	OMP	OMR	EP	OMP	OMR	EP	OMP	OMR
Not possible	0-shot Llama2-7B-chat	0.056	0.578	0.560	0.250	0.695	0.662	0.137	0.584	0.547
	0-shot Llama2-70B	0.113	0.670	0.521	0.288	0.725	0.630	0.190	0.692	0.548
	0-shot Llama2-70B-chat	0.079	0.653	0.631	0.325	0.775	0.738	0.220	0.646	0.621
	0-shot GPT3.5	0.133	0.716	0.689	0.438	0.797	0.759	0.273	0.724	0.689
Medium	Snippets + Llama2-7B-chat	0.213	0.640	0.553	0.289	0.674	0.610	0.224	0.635	0.572
	Snippets + Llama2-70B	0.242	0.723	0.567	0.338	0.723	0.611	0.320	0.743	0.581
	Snippets + Llama2-70B-chat	0.278	0.743	0.728	0.494	0.838	0.808	0.302	0.749	0.715
	Snippets + GPT3.5	0.303	0.825	0.751	0.548	0.834	0.724	0.381	0.815	0.726
High	CoQEx [7]	0.175	0.631	0.577	0.329	0.665	0.626	0.266	0.696	0.611
	CardiO (vanilla)	0.192	0.659	0.659	0.298	0.661	0.661	0.228	0.618	0.618
	+ LLM Sentence filter	0.191	0.663	0.653	0.293	0.657	0.634	0.229	0.657	0.634
	+ LLM Count extraction	0.174	0.571	0.571	0.298	0.661	0.661	0.205	0.601	0.601
	+ Peer calibration	0.190	0.657	0.657	0.310	0.654	0.654	0.234	0.623	0.623
	+ Central aggregation	0.196 (±0.029)	0.630 (±0.013)	0.630 (±0.013)	0.298	0.634	0.634	0.253	0.649	0.649
	+ Consistent aggregation	0.186 (±0.064)	0.653 (±0.023)	0.653 (±0.023)	0.274	0.660	0.660	0.240	0.642	0.642

inhabited, which can but need not come from a snippet. CardiO provides the total number of islands, the exact sentence, the snippet, and additional snippets all with numbers in the 200,000s, confirming the answer's order of magnitude. As Sweden's peers CardiO returns Finland, Norway, Indonesia, Canada & Australia. For the number of *musicians who have won a Nobel Prize*, where the snippets do not have the count, CardiO uses peers to predict 5 (7 being the ground truth), while LLM-only models return names. Some of these names appear in snippets[5]: GPT3.5 returns Bob Dylan in both settings; Llama2-70B-chat additionally returns Rabindranath Tagore with snippets and in 0-shot mode returns three other names, which on further inspection were not found in the snippets.

LLM enhancements do not improve overall performance. The LLM sentence filter and count extraction methods increase the OMP for 7% and 11% of the questions, respectively, and decrease the OMP for 11% to 27% of the questions. Peer calibration, when used with the default path, affects 8 questions, of which 5 gain in OMP. CardiO achieves 0.28 in both OMP and OMR in 331 queries, using only peer calibration, indicating high variance among peers.

Aggregation strategies. Supporting evidence-based central and consistent aggregation methods boost CardiO (vanilla)'s performance in the CoQuAD dataset, and the central aggregation method affects the exact precision. As the number of supporting evidence changes with the questions, these metrics may perform better with thresholding the distance metric than taking top-k neighbors.

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

6 CONCLUSION

We propose the CardiO framework to answer count questions on the Web with rich representation. CardiO's simple framework beats small pure LLM-based Llama models. We observe a performance gap between monolithic LLM-only systems and more transparent and traceable systems. Moreover, the answers from LLM-only methods are not directly traceable to the source. These numbers and claims need further verification, thus creating many exciting avenues for future work.

[5]Bob Dylan was the most frequent in the snippets, followed by Rabindranath Tagore

REFERENCES

[1] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.
[2] Paolo Boldi et al. 2011. Query reformulation mining: models, patterns, and applications. *Information retrieval* (2011).
[3] Valeriia Bolotova et al. 2022. A non-factoid question-answering taxonomy. In *SIGIR*.
[4] Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: Is more always better?. In *SIGIR*. 291–298.
[5] Luis Galárraga, Simon Razniewski, Antoine Amarilli, and Fabian M. Suchanek. 2017. Predicting Completeness in Knowledge Bases. In *WSDM*.
[6] Sebastian Gehrmann et al. 2023. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *JAIR* (2023).
[7] Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. 2022. Answering Count Queries with Explanatory Evidence. In *SIGIR*.
[8] Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. 2023. Class Cardinality Comparison as a Fermi Problem. In *WWW Companion*.
[9] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. In *ICML*.
[10] Vinh Thinh Ho et al. 2021. Extracting contextualized quantity facts from web tables. In *WWW*.
[11] Mandar Joshi et al. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL*.
[12] Vladimir Karpukhin et al. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *EMNLP*.
[13] Tom Kwiatkowski et al. 2019. Natural questions: a benchmark for question answering research. *TACL* (2019).
[14] Michael Luggen et al. 2019. Non-parametric class completeness estimators for collaborative knowledge graphs—the case of wikidata. In *ISWC*.
[15] Paramita Mirza, Simon Razniewski, Fariz Darari, and Gerhard Weikum. 2018. Enriching Knowledge Bases with Counting Quantifiers. In *ISWC*.
[16] Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. 2002. Probabilistic question answering on the Web. In *WWW*.
[17] Simon Razniewski et al. 2024. Completeness, Recall, and Negation in Open-World Knowledge Bases: A Survey. *Comput. Surveys* (2024).
[18] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*.
[19] Soo Young Rieh and Hong Xie. 2006. Analysis of multiple query reformulations on the web. *Information Processing and Management* (2006).
[20] Swarnadeep Saha and Harinder Pal. 2017. Bootstrapping for Numerical Open IE. In *ACL*.
[21] Arnaud Soulet et al. 2018. Representativeness of Knowledge Bases with the Generalized Benford's Law. In *ISWC*.
[22] Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro A. Szekely. 2021. Representing Numbers in NLP: a Survey and a Vision. In *NAACL-HLT*.
[23] Hugo Touvron et al. 2023. Llama 2: Open foundation and fine-tuned chat models. (2023). arXiv:2307.09288
[24] Jason Wei et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS* (2022).
[25] Ralph Weischedel et al. 2012. *OntoNotes Release 5.0. Linguistic Data Consortium.* Technical Report.