Language Models as Context-sensitive Word Search Engines

Matti Wiegmann and Michael Völske and Benno Stein
Bauhaus-Universität Weimar
{matti.wiegmann,michael.voelske,benno.stein}@uni-weimar.de

Martin Potthast
Leipzig University
martin.potthast@uni-leipzig.de

Abstract
Context-sensitive word search engines are writing assistants that support word choice, phrasing, and idiomatic language use by indexing large-scale n-gram collections and implementing a wildcard search. However, search results become unreliable with increasing context size (e.g., $n \geq 5$), when observations become sparse. This paper proposes two strategies for word search with larger $n$, based on masked and conditional language modeling. We build such search engines using BERT and BART and compare their capabilities in answering English context queries with those of the $n$-gram-based word search engine Netspeak. Our proposed strategies score within 5 percentage points MRR of $n$-gram collections while answering up to 5 times as many queries.\(^1\)

1 Introduction
A wide range of computer tools has been developed to support the writing process, including both active and passive ones. Active tools automatically paraphrase a text as it is written, if the text is highly likely to be incorrect or stylistically inappropriate. Passive tools suggest either spelling, grammar, and style corrections or how to continue a sentence. Passive tools that are less integrated into word processors are context-free and context-sensitive word search engines. Context-free search engines include searchable dictionaries, thesauri, and collections of idioms in which queries are made about a known word or phrase for which alternatives are sought. In the absence of context, their search results are usually sorted alphabetically. Context-sensitive word search engines allow their users to formulate cloze-style queries to search for an unknown word or phrase, ranking the search results according to their frequency of use.

A conventional context-sensitive word search engine, as shown in Figure 1, answers a cloze query $q = \text{the * fox}$ asking for words or phrases commonly written between ‘the’ and ‘fox’ by retrieving the appropriate subset $D_q \subseteq D$ from a collection of $n$-grams $D$. Formally, the index $\mu : Q \to \mathcal{P}(D)$ maps the set of cloze queries $Q$ to the power set $\mathcal{P}(D)$, which is implemented as wildcard retrieval, and the results $\mu(q) = D_q$ are ordered by their occurrence frequency in a large text corpus, which approximates the frequency of use. Assuming a sufficiently large text corpus is available such that each $n$-gram matching a given cloze query $q$ has been observed sufficiently often, ranking these $n$-grams by their frequency satisfies the probability ranking principle (Robertson, 1977). In other words, if one asks a sufficiently large number of people to answer a cloze query, the frequency distribution of the answers would correlate with that of the $n$-grams found. The main limitations of this approach are, (1) that the number of context words in each cloze query is limited by $n$, with more context reducing the size of the cloze accordingly, and, (2) that the size of the text corpus required to observe $q$ sufficiently often increases exponentially with $n$, so that in practice $n < 10$.

In this work, these two limitations are addressed by using transformer-based language models to predict phrases corresponding to a query, rather than retrieving them from an $n$-gram index. In particular, we propose a masked language model and an autoregressive model for conditional generation to answer cloze queries (Section 3). These models are compared to Netspeak, a state-of-the-art

---

\(^1\)Our code is available at Github and our data is available at Zenodo.
Table 1: Selected context queries with the <original substitution (McCarthy and Navigli, 2007; Lee et al., 2021), word sense disambiguation, paraphrasing, and phrase-level substitution (Madnani and Dorr, 2010), although these tasks usually require a known word or phrase.

Expression matching and corpus-based statistics form the basis for writing assistants, while language models, mostly based on the transformer architecture (Vaswani et al., 2017), often take on the heavy lifting (Alikaniotis et al., 2019). Transformer-encoder models, like BERT (Devlin et al., 2019), are often pre-trained by masked language modeling, which is highly similar to wildcard word search but knows only one correct target. Encoder models are frequently applied to solve cloze tests (Gonçalo Oliveira, 2021) and its related foundational tasks. Autoregressive language models, like GPT (Radford et al., 2019), are used for infilling (Donahue et al., 2019), which is similar to mask prediction but generates arbitrary-length sequences. Conditional language models (autoencoders) are used in phrase-level substitution tasks like denoising (Lewis et al., 2019).

3 Language Modeling for Word Search

In this work, we formulate context-sensitive word search with language models as learning a distribution \( p(w_q | q) \), where \( q = q_l \) is \( q_r \) consists of left and right side contexts \( q_l \) and \( q_r \) and a wildcard token \( \_ \). Either \( q_l \) or \( q_r \) can be empty. The result set \( D_q \) consists of all \( n \)-grams \( q_l w_{q,i} q_r \) for all \( w_{q,i} \in w_q \), in

context-sensitive word search engine based on an index of Google \( n \)-grams (Section 4). Based on the cloze test corpus CLOTH (Xie et al., 2018) and Wikitext (Merity et al., 2016), both of our proposed language models achieve an MRR near their theoretical maximum, falling short of Netspeak’s only between 0.03–0.07, and they exceed a mean nDCG index of Google search with language models as learning a distribution \( p(w_q | q) \), where \( q = q_l \) is \( q_r \) consists of left and right side contexts \( q_l \) and \( q_r \) and a wildcard token \( \_ \). Either \( q_l \) or \( q_r \) can be empty. The result set \( D_q \) consists of all \( n \)-grams \( q_l w_{q,i} q_r \) for all \( w_{q,i} \in w_q \), in
Table 2: The original token (OT) dataset consists of n-gram queries extracted from Wikitext-103 and CLOTH and lists the original token as the single answer. The ranked answers (RA) dataset is extracted from OT by replacing the answer with the ranked results retrieved from Netspeak, discarding all unanswered queries.

Table 3: The average MRR of the original token for all queries in the OT test datasets, split by source and query length. NA ⊆ OT only considers queries that Netspeak could answer and Ratio indicates the subset size. Time indicates the average response time for one query.

4 Experimental Setup

We implemented both strategies of learning context-sensitive word search using the Huggingface (Wolf et al., 2020) implementation of DistilBERT for MLM and BART for CDLM. We evaluate the pre-trained and the fine-tuned models against the two datasets with word search queries shown in Table 2.

Data We constructed two datasets with word search queries. The original token (OT) dataset offers as the single answer the token chosen by the author of the source text. The ranked answers (RA) dataset offers multiple, ordered answers with relevance judgments for each query.

The original token dataset consists of queries extracted from Wikitext-103 (Merity et al., 2016), which consists of good or featured English Wikipedia articles, and CLOTH (Xie et al., 2018), which consists of middle and high school learner’s English cloze-tests. For Wikitext, we constructed queries for each 3-to-9-gram by replacing the token at each position in the n-gram with a wildcard and adding the original token as the answer. We discarded all newlines, headlines starting with a =, n-grams with non-letter tokens to not cross sentence boundaries or quotations, and queries with proper nouns as answers. For CLOTH, we constructed a query for each 3 and 5-gram that overlapped with a cloze-gap in the dataset and added the teacher’s preferred answer as the original token answer. We discarded all n-grams with non-letter tokens and proper nouns as answers. Each wildcard was assigned one of 5 word classes based on Spacy’s POS annotations of the source sentences: verbs and auxiliaries, nouns, determiners and pronouns, adjectives and adverbs, and conjunctions and particles. Verb and noun classes were marked
if the query contains another verb or noun, respectively. As the training set, we selected the first 10 million queries from the training split of Wikitext. As the dev set, we selected all queries extracted from Wikitext’s dev split. As the test set, we used all 3 and 5-gram queries from Wikitext’s test split and all CLOTH splits.

The ranked answers datasets consist of the queries from the original token dataset, but all answers were replaced by the top 30 results retrieved from Netspeak, which is equivalent to the most frequent observations in Google n-grams. We assigned a relevance score to each result based on its absolute frequency: above 100K we assigned a high (3) score, above 10K a medium (2) score, with any occurrence a low (1), and otherwise a zero (0) relevance score. We discarded all queries with an empty result set. We determined the splits analogously to the original token dataset.

Model Configuration For the MLM strategy, we fine-tune Huggingface’s implementation of DistilBERTForMaskedLM on the original token dataset, using the pre-trained distilbert-base-uncased checkpoint. We only exposed one n-gram as input at a time. We train the model using the standard training routine with default parameters, although we doubled the masking probability to 30 %, twice the rate used for BERT (Devlin et al., 2019), and adapted the initial learning rate to 2e-5 and the weight decay to 0.01. We evaluate the performance once with the pre-trained checkpoint as dBERT and once after fine-tuning as dBERT_ft.

For the CDLM strategy, we fine-tune Huggingface’s implementation of BARTFor-ConditionalGeneration for infilling on the ranked answers dataset using the pre-trained facebook/bart-base checkpoint. We only exposed one n-gram as input at a time and used the same hyperparameters as with the MLM strategy, except that masking was done manually. We evaluate the performance with the pre-trained checkpoint as BART and after fine-tuning as BART_ft.

5 Evaluation

We quantitatively evaluate our proposed methods using the mean reciprocal rank (MRR) and the normalized discounted cumulative gain (nDCG) (Järvelin and Kekäläinen, 2002).

System Performance We evaluate the system performance using the MRR of the author’s chosen word, shown in Table 3, assuming that the author’s chosen word in the source text is also a good answer to the cloze query. Therefore, the better word search engine should rank the author’s choice higher on average over many queries. Table 3 shows the MRR for the four models compared to Netspeak, once over all queries in the test datasets, and once for the shared subset of queries where Netspeak returned non-empty results.

The MRR results allow three conclusions. First, our proposed fine-tuning strategy improves the pre-trained baseline’s performance consistently for BART and on queries from Wikitext for dBERT. Second, on queries from RA, the best models already perform close to Netspeak. Third, both fine-tuned models can answer 4-5 times as many queries than Netspeak, which can be observed from the ratio between RA and OT datasets. Since the OT dataset contains up to 82% uncommon queries, which have no support in the Google n-grams indexed by Netspeak, the language models...
Figure 4: The MRR by word class (left) and wildcard position (center and right) of Netspeak and the four Models on the Original Token test dataset. Queries that Netspeak could not answer were ignored. The gray bars indicate the relative frequency.

score up to 9 percentage points lower than on RA. The MRR increases with increasing context size since additional context can only reduce the set of potentially matching answers.

**Ranking** We evaluate the ranking of the results using the nDCG as shown in Figure 3. Consistent with the MRR results, the fine-tuned models outperform their pre-trained counterpart, dBERT profits more from fine-tuning and performs best. Most of the relevant results are in the top ranks since the nDCG scores only marginally improve past rank 10.

**Position and Word Class** We evaluate further query attributes besides size and genre, wildcard position, and wildcard word class, using the MRR as shown in Figure 4. These results show that a large part of the performance gain when fine-tuning can be attributed to gains in the closed-class words. The MRR is lower for open-class words since there are more plausible options for each query and the original token is on a lower rank more often. Fine-tuning has only a marginal impact on open-class words. dBERT scores the lowest when the wildcard is either at the beginning or at the end of the query, while BART scores the lowest for wildcards at the beginning. Fine-tuning significantly improves the performance in these cases, with only marginal improving queries with wildcards in the center positions.

The performance difference between closed and open-class words also partially explains the substantially lower MRR and nDCG scores over CLOTH queries for all models: The answers to cloth-queries more often belong to lower scoring open classes, the answers to Wikitext-queries more frequently belong to the high scoring closed classes.

**Runtime** We compare the runtime performance by measuring the average time to answer a query (see Table 3) over all queries in the ranked answers test dataset. Netspeak and dBERT are equally fast with 5 ms per query, while BART takes twice as long. In practice, both language models are fast enough for context-sensitive word search. We measured the performance of the language models with sequential, non-batched queries on GPU. We measured the performance of Netspeak with a local Netspeak instance and a local index, queried through Netspeak’s GRPC API. All systems were tested in identical containers with 4 AMD EPYC 7F72 CPU cores, 32 GB of RAM, and one A100 GPU.

6 **Conclusion**

This paper investigates whether state-of-the-art language models can mitigate the shortcomings of n-gram indices in context-sensitive word search engines. We present strategies to fine-tune masked and conditional language models so that they can answer word search queries. Our evaluation shows that our proposed methods can answer short queries (3 tokens) nearly as well as by observing actual n-gram frequencies in a large text corpus. Further-
more, our fine-tuned models perform well when supporting observations are scarce so that $n$-gram indices provide no results. Since this already is the dominant case for $n = 5$, we can conclude that language models, fine-tuned for word search queries, are a suitable extension to context-sensitive word search engines.

Impact Statement

Context-sensitive word search engines provide easier access to language resources and our work extends this to data from language models. This implies an increased risk of leaking sensible data contained in the source data. We avoided training models to predict proper nouns to avoid that a model can be used to search for personal information.

We use and combine data from Wikitext (i.e. Wikipedia), CLOTH, and the Google Web and Books $n$-grams, obtained from publicly available and appropriately acknowledged sources and according to their terms and conditions. Our derived systems and evaluation procedure may be susceptible to biases inherent in the data we used. We took no extra steps to de-bias the models or data used.

References


