A New Neural Search and Insights Platform for Navigating and Organizing AI Research

Marzieh Fadaee[*] **Olga Gureenkova**[*] **Fernando Rejon Barrera**[*]
Carsten Schnober[*] **Wouter Weerkamp**[*] **Jakub Zavrel**[*]
Zeta Alpha
`{lastname}@zeta-alpha.com`

Abstract

To provide AI researchers with modern tools for dealing with the explosive growth of the research literature in their field, we introduce a new platform, AI Research Navigator, that combines classical keyword search with neural retrieval to discover and organize relevant literature. The system provides search at multiple levels of textual granularity, from sentences to aggregations across documents, both in natural language and through navigation in a domain specific Knowledge Graph. We give an overview of the overall architecture of the system and of the components for document analysis, question answering, search, analytics, expert search, and recommendations.

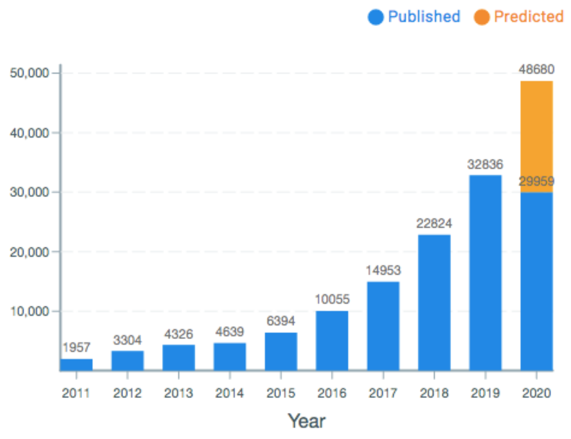


Figure 1: Growth of AI related documents on arXiv.

1 Introduction

The growth of publications in AI has been explosive in recent years. A big portion of this growth is happening on platforms outside of traditional publishing venues, for instance arXiv e-print archive (see Figure 1) and blogs. Although this encourages broad access to AI expertise and technology, it makes efficient and effective search, monitoring, and discovery in the AI field increasingly difficult. Most general-purpose academic search engines lack a specialization on AI content and practical know-how, because they focus on classical bibliographic information across all scientific disciplines. At the same time, academic search engines often do not make use of the latest AI technologies in search, as well as natural language processing (NLP) and insights capabilities. The main reason that limits them is the need to operate at a much larger scale and cover a large amount of knowledge.

Recent developments in various NLP tasks are showing fast progress towards an almost human level of language understanding (Devlin et al., 2019; Brown et al., 2020). Applying these new technologies to the processing of research and engineering literature bears the promise of accelerating scientific discovery. In addition, providing efficient tools to automate some of the drudgery of human scholarly work by machine understanding of scientific knowledge is extremely valuable. Similar directions are being explored in recent studies (Ammar et al., 2018; Kardas et al., 2020; Zhao and Lee, 2020).

Our system, AI Research Navigator[†], aims to help AI researchers with a simple-to-use semantic search for documents (§4.1), the answering of detailed factual questions (§4.3), the generation of insights via visual analytics (§4.4), combined with recommendations to filter the constant flood of new information in their field (§4.5). These technologies are combined in the platform with both project and task-oriented tools to support a more effective and efficient organization of a researcher's work on multiple projects and topics (§5). This paper presents a short outline of the system.

[*] All authors contributed equally.

[†] `search.zeta-alpha.com`

2 Document Analysis

A key ingredient to an AI insights platform is the content available to users. We ingest, process, and store documents from a variety of sources, aiming to get broad coverage, as well as detailed views on theoretical and applied AI. At the moment our platform contains approximately 140 thousand scientific papers, collected from `arXiv.org`[*] and `OpenReview.net`, and about 24 thousand posts from data science blogs. Our goal for the near future is to expand this set of sources to include different types of content (e.g., source code, news, tweets). We ingest new content from our sources on a daily basis, offering our users the latest insights. Newly ingested or updated content is fed to our back-end storage system (Section 3) via a distributed processing pipeline that takes the documents through a number of processing and information extraction steps and generates embeddings that capture the intent and meaning of the text. We also extract images from each document to serve as an illustration for the paper in the search engine. We manage the state of each document, as it traverses the pipeline, with a messaging queue platform (Apache Pulsar). This allows us to scale our processing throughput, while keeping certain processing guarantees.

2.1 Parsing and Linking Documents

Scientific publications consist of sections that have varying degrees of informativeness. As an example, the bibliography of a paper is interesting for the citation graph, but does not contain actual new content. In order to process and index only relevant and informative sections, we parse the document structure and extract candidate citation records using ParsCit (Councill et al., 2008). We then sanitize these candidates using a set of heuristics, and link them to our Knowledge Graph (KG, Section 3) using fast approximate string matching.

We make use of domain-specific concepts and their relations to improve the effectiveness of components like Question Answering (QA, Section 4.3), KG population, analytics processing (Section 4.4), and semantic search (Section 4.1). We train a statistical named entity recognizer (NER) using a small manually curated seed set of around a thousand AI related concepts, and run this NER on every document that we process. We then link the recognized concepts to concepts in our KG using a weighted combination of string and contextual embedding similarity. Finally, those entities that were not linked because they fell below a similarity threshold are considered as new candidates to further populate the KG. Figure 2 shows the domain-specific concept types that are currently in our KG.

[*]We include all papers from the following AI related categories: cs.AI, cs.LG, cs.CV, cs.IR, CS.NE, cs.CL, and stat.ML.

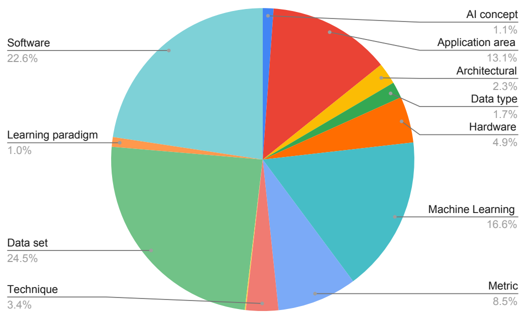


Figure 2: Distribution of KG concept types.

3 Storage Systems

To provide access to the information we obtained from our processing pipeline, we currently store our information in three core systems.

Knowledge Graph. We store people, content, and concepts in a knowledge graph (stored in Dgraph). The node identifiers are used in the search frontend for navigating content and building queries.

Search Index. For fast access to documents, we use an open-source search engine (Elasticsearch) in combination with HNSW (Malkov and Yashunin, 2016) for nearest neighbor search. We use three separate indices, one for sentences (31M), one for chunks (approx. 10 consecutive sentences) (4.2M), and one for full documents (160K) and citation records (740K).

Document Representations. As a result of each processing step, we store new document representations. Most representations are in the form of standoff annotations, linking labels (e.g., a concept ID or vector) to a particular span of characters in the source document.

4 Accessing Information

Processing and storing information is only useful when we can provide meaningful access to it. Our platform allows information access in a variety of ways. In this section, we discuss content (§4.1) and expert (§4.2) search, QA system (§4.3), analytics component (§4.4), and recommendations system (§4.5).

4.1 Content Search

One of the main methods to access information on our platform is search. We currently support traditional keyword-based search and vector-based (nearest neighbor) search. Both search systems offer valuable information. While keyword-based search is useful in finding documents directly related to the query, vector-based search offers a range of query interpretations and more diversity.

Keyword-based search. Our keyword-based search functionality scores documents for a given user query based on several heuristics. (1) We borrow from Metzler and Croft (2005) the notion of sequential dependencies between query terms, construct term n-grams from the user query, and treat each n-gram as a phrase query. (2) A match of a longer n-gram is more important than the match of a shorter n-gram, which is implemented as a dynamic boost per n-gram query. (3) We combine evidence from multiple document fields (Ogilvie and Callan, 2003) and assign higher weights to metadata fields like author name, title, and abstract, while limiting the weight of the full text field. (4) We use a `dismax` query over fields to determine whether an n-gram refers to an author or to content. (5) Given the limited text length of the metadata fields, we only rely on term presence, and assign a constant field-dependent score. (6) Finally, we assume that highly cited and recent documents are more important to users.

Vector-based. In many cases, keyword searches are hard to use when exploring a new domain. To allow a more meaning-based exploration, fully neural retrieval models can be beneficial. Recent advances in neural language modeling as unsupervised pretraining have achieved significant improvements in a wide variety of NLP tasks (Devlin et al., 2019). However, incorporating them in retrieval systems presents some challenges. Using pretrained language models to jointly encode queries and documents is often not computationally feasible for large-scale retrieval. Recent studies propose various methods to benefit from large neural models. Luan et al. (2020) propose a hybrid method for combining sparse and dense representations that outperforms baselines in open retrieval. Chang et al. (2020) use a siamese network, initialized with BERT, to encode query and document individually. They propose three self-supervised tasks that capture different aspects of query-document relations.

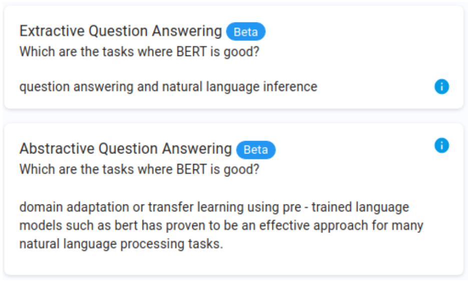
Inspired by these studies, we use the Sentence-BERT model proposed by Reimers and Gurevych (2019) to generate sentence embeddings for each document and, additionally, we also encode all words in context with SciBERT (Beltagy et al., 2019) embeddings. Finally, we encode sentences, chunks, and full documents into representative vectors, fine-tuned on self-supervised training tasks similar to Chang et al. (2020). In our platform, we encode the query as a vector at search time, and retrieve its N (approximate) nearest neighbor documents in the vector space. This requires the documents and the queries to be encoded in a similar way using the same embedding space. By using HSNW and loading its full graph in memory, we are able to serve nearest neighbor search results in a highly efficient manner.

4.2 Expert Search

In addition to navigating knowledge via natural language search and domain-specific topics from our KG, we also aim to improve navigation by connecting searchers to experts. For this, we follow a document-centric approach to expertise retrieval, along the lines sketched in Balog et al. (2009) and Husain et al. (2019). In the expert search component, we embed user queries and documents using the Sentence-BERT model similar to Section 4.1. This allows the system to retrieve papers that are related to the query. We then derive the experts from the sets of authors of these papers using an approach where each retrieved paper contributes an exponentially weighted vote for an author, with a factor that reduces the bias towards highly prolific authors. Our experiments, described in detail in (Berger et al., 2020) show that these modern Transformer-based contextualized embeddings outperform TF-IDF and LSI-based document representations on this task.

4.3 Question Answering

Our QA module provides an answer to either a concrete user question (e.g., *How many TPUs are needed to train BERT?*) or, alternatively, to a question related to the user's query, which we automatically generate (e.g., *What is a knowledge graph?* for the query *knowledge graph*). To distinguish


Extractive Question Answering Beta
Which are the tasks where BERT is good?

question answering and natural language inference

Abstractive Question Answering Beta
Which are the tasks where BERT is good?

domain adaptation or transfer learning using pre - trained language models such as bert has proven to be an effective approach for many natural language processing tasks.

Figure 3: Extractive and abstractive QA components.

between questions and other types of queries, we use a Naive Bayes classifier trained on the NPS Chat Corpus (Forsyth et al., 2006).

We deploy two types of QA deep learning models: extractive QA and abstractive QA (see example in Figure 3). Both models take as input a set of relevant documents, and provide the user with one or more answers. Since an answer is always part of a particular context, we also present this context as a source of explanation of the answer to the user.

Extractive QA. Our extractive QA model is built with an existing BERT-based question answering model from the DeepPavlov library (Burtsev et al., 2018). The model takes as input a pair of question and context and rate their relatedness. At query time, we chunk the input documents and send multiple question-context pairs to the model. We obtain the best answer by filtering the candidates according to the confidence of the model.

Abstractive QA. For the abstractive answers we use a model based on the approach proposed in Nishida et al. (2019), trained on the MS-MARCO data set (Bajaj et al., 2016). Our model and its evaluation in the AI domain are described in detail in Tsiamas (2020). Since this architecture has its own neural retrieval component, at query time the model has access to the question and all input documents. Unlike extractive QA, this model is also capable of answering yes/no questions.

The two QA models complement each other in the types of answers they provide. Although these systems are still experimental (approximately 70% of answers to a benchmark set of in-domain questions were relevant), together with sentence and paragraph retrieval they show potential for discovering interesting information that goes beyond what surface-level single-document-based systems provide.

4.4 Analytics

Rather than reviewing a long list of documents or reading a short answer in response to a query, sometimes users can get to an insight faster by observing a tabular, a summary, or a graph overview over the entire set of relevant documents.

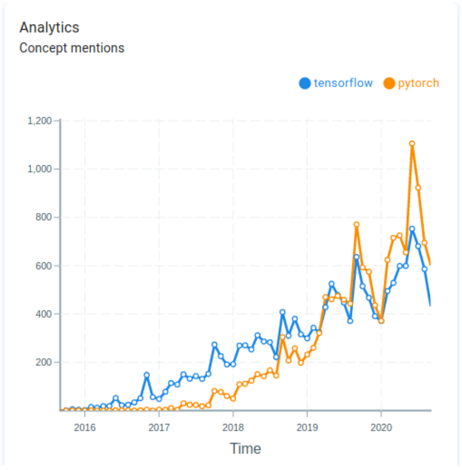


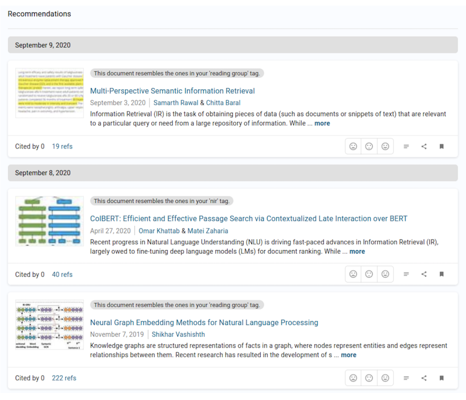
Figure 4: Contrastive popularity analytics.

Our analytics module aims to give users this quick and easy-to-grasp overview over a result set when a query sufficiently matches some predefined analytics query templates. For instance, when we detect AI concepts from the KG in a query, or a reference to an abstract concept (e.g., *"Which datasets are used for image classification?"*), we show the contrastive popularity plots for the specific concepts as identified in documents using the NER and linker module. Figure 4 provides an example of a contrastive popularity graph. The graphs provide a global overview and are also clickable so the users can use them to quickly identify patterns and discover specific papers relevant to their interests.

4.5 Recommendations

With the amount of new information available on a daily basis, a recommender system is inevitable to filter and keep track of relevant publications. Users of our platform receive recommendations in notification emails and in the recommendations view (Figure 5).

The relevance of a publication can be decomposed into several factors. We implemented a modular system architecture which allows us to weigh relevance factors on a per-user basis, and to add, remove, modify, and evaluate modules individually. Each module generates recommendations


Recommendations

September 9, 2020

This document resembles the ones in your 'reading group' tag.

Multi-Perspective Semantic Information Retrieval

September 3, 2020 | Samarth Rawal & Chitta Baral

Information Retrieval (IR) is the task of obtaining pieces of data (such as documents or snippets of text) that are relevant to a particular query or need from a large repository of information. While ... more

Cited by 0 19 refs

September 8, 2020

This document resembles the ones in your 'nir' tag.

ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT

April 27, 2020 | Omar Khattab & Matei Zaharia

Recent progress in Natural Language Understanding (NLU) is driving fast-paced advances in Information Retrieval (IR), largely owed to fine-tuning deep language models (LMs) for document ranking. While ... more

Cited by 0 40 refs

This document resembles the ones in your 'reading group' tag.

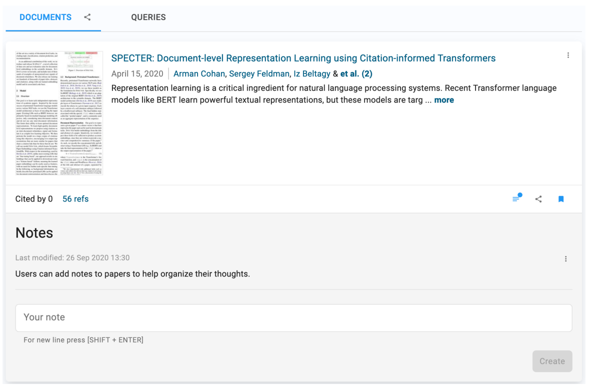
Neural Graph Embedding Methods for Natural Language Processing

November 7, 2019 | Shikhar Vashishth

Knowledge graphs are structured representations of facts in a graph, where nodes represent entities and edges represent relationships between them. Recent research has resulted in the development of s ... more

Cited by 0 222 refs

Figure 5: Recommendations on our platform.



SPECTER: Document-level Representation Learning using Citation-informed Transformers

April 15, 2020 | Arman Cohan, Sergey Feldman, Iz Beltagy & et al. (2)

Representation learning is a critical ingredient for natural language processing systems. Recent Transformer language models like BERT learn powerful textual representations, but these models are targ ... more

Cited by 0 56 refs

Notes

Last modified: 26 Sep 2020 13:30

Users can add notes to papers to help organize their thoughts.

Your note

For new line press [SHIFT + ENTER]

Figure 6: Adding tags and notes to a document.

with (normalized) scores, which are aggregated by the core recommender system, allowing for both global and personalized weights for each module.

Our recommendation architecture is based on hybrid recommender systems (Gomez-Uribe and Hunt, 2016), combining content-based and collaborative filtering. However, since virtually all recommendations are of new papers, we suffer from the cold-start problem and we mostly rely on content-based recommendations. The content-based module generates recommendations based on user-tagged documents: when a user tags a document, it triggers an initial search for related documents. From this point on, recommendations are only generated from the most recent documents.

Our current basic content-based modules are based on similarity metrics derived from our document representations, as described in section 4.1, with score normalization being provided by leave-one-out tuning on the set of documents in a tag.

Apart from these content-based similarity recommendations, we are also experimenting with additional modules that provide similarity scores. **Citation-based** recommendations are based on (in-direct) citations to documents stored by the user. **Author-based** recommendations are (co-)authored by authors which are frequently tagged by a user. **Popularity** recommendations are globally "popular" documents, for instance based on the number of views, citation counts, or tag counts.

5 Productivity Tools

Discovering relevant information in an effective way is key to researchers, knowledge workers, and decision makers. Even though an AI-enabled platform like the one described in this paper can be helpful for this purpose, it is only the first step in researching a topic. Organizing and accessing this information is a necessary feature. Users of our platform are also supported to organize information and knowledge without having to rely on external tools for reading lists and notes. Having found a relevant piece of information in our system, users can save this into their own specific project and topic tags. They can also directly write their notes on the papers and projects they are working on in the tool. Tags serve to organize lists of documents, as well as the queries used, and notes taken while working on a project (see Figure 6). The tagging system can also be used to track the status and priority of work. Tag-based lists can easily be shared with others within the platform, on social media, and exported into other tools. As described above, these tags are also the starting point that allows users to be alerted about new results relevant to their interests.

6 Discussion and Next Steps

Having introduced a new platform to discover and organize knowledge for AI researchers, we foresee considerable future research to reach real machine understanding of scientific literature, such as extraction of complex entity relations and more advanced use of neural embeddings to reduce the dependency on manual KG curation. We leverage a mix of state-of-the-art AI components to give researchers transparent access to a body of knowledge from a large volume of heterogeneous and non-reviewed content. As a result, it raises the concern of dealing with fairness, factuality, conflicting opinions, and out-of-date information, which requires deeper investigation. Finally, we are interested to further explore how the productivity tools in our platform can contribute to better collaboration in teams and improving knowledge sharing and discovery.

Acknowledgements

We would like to acknowledge the joint effort from Zeta Alpha's team including Shamil Mammadov and Victor Zuanazzi to make this work possible.

References

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, et al. 2018. Construction of the literature graph in semantic scholar. *arXiv preprint arXiv:1805.02262*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. Ms marco: A human generated machine reading comprehension dataset.

Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. 2009. A language modeling framework for expert finding. *Information Processing and Management, 45(1)*, page 1–19.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Mark Berger, Jakub Zavrel, and Paul Groth. 2020. Effective distributed representations for academic expert search. In *First Workshop on Scholarly Document Processing*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, Alexey Litinsky, Varvara Logacheva, Alexey Lymar, Valentin Malykh, Maxim Petrov, Vadim Polulyakh, Leonid Pugachev, Alexey Sorokin, Maria Vikhreva, and Marat Zaynutdinov. 2018. DeepPavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127, Melbourne, Australia. Association for Computational Linguistics.

Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. In *International Conference on Learning Representations*.

Isaac Councill, C. Lee Giles, and Min-Yen Kan. 2008. ParsCit: an open-source CRF reference string parsing package. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Eric Forsyth, Jane Lin, and Craig Martell. 2006. The NPS Chat Corpus. Data retrieved from Linguistic Data Consortium, `https://catalog.ldc.upenn.edu/LDC2010T05`.

Carlos A. Gomez-Uribe and Neil Hunt. 2016. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4):1–19.

Omayma Husain, Naomie Salim, Rose Alinda Alias, Samah Abdelsalam, and Alzubair Hassan. 2019. Expert finding systems: A systematic review. *Applied Sciences (Switzerland)*, 9(20):1–32.

Marcin Kardas, Piotr Czapla, Pontus Stenetorp, Sebastian Ruder, Sebastian Riedel, Ross Taylor, and Robert Stojnic. 2020. Axcell: Automatic extraction of results from machine learning papers. In *2004.14356*.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, dense, and attentional representations for text retrieval.

Yury A. Malkov and D. A. Yashunin. 2016. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *CoRR*, abs/1603.09320.

Donald Metzler and W. Bruce Croft. 2005. A Markov Random Field Model for Term Dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 472–479. ACM.

Kyosuke Nishida, Itsumi Saito, Kosuke Nishida, Kazutoshi Shinoda, Atsushi Otsuka, Hisako Asano, and Junji Tomita. 2019. Multi-style generative reading comprehension.

Paul Ogilvie and Jamie Callan. 2003. Combining Document Representations for Known-Item Search. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 143–150. ACM.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Ioannis Tsiamas. 2020. Complex question answering by pairwise passage ranking and answer style transfer. In *Masters Thesis, University of Amsterdam*.

Tiancheng Zhao and Kyusong Lee. 2020. Talk to papers: Bringing neural question answering to academic search. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 30–36, Online. Association for Computational Linguistics.