An Ensemble Approach to Data Mining for Real-time Information Retrieval

WWT Artificial Intelligence Research & Development

SEPTEMBER | 2020

By: Vishal Jain, Vinay Garg and Patrick McDermott

World Wide Technology
wwt.com

Table of Contents

Abstract

Information retrieval (IR), locating relevant documents in a document collection based on a user's query, is a common problem in text analysis. Traditional keyword-based IR engines are good at finding relevant information, but struggle to provide semantic and contextual results for complex queries. We propose an ensemble approach, a combination of multiple Natural Language Processing (NLP) models, which transform documents into vectors, followed by scoring and ranking documents based on their relevance to the user search query. To provide an effective search mechanism over a large document corpus, we incorporate Elasticsearch in our solution. It is a popular distributed search engine which allows indexing and searching of documents in near real-time. With the retrieved documents, we generate multi-sentence summaries using an extractive text summarizer to make it easier for users to glean the relevant content. All these components are packaged into an end-to-end solution encapsulated by a Python Flask UI where the users can enter a search query and get the relevant results in near real-time. A major challenge faced in evaluating the performance of NLP-based IR models is the absence of relevance labels or scores for document-query pairs. To benchmark the performance of our ensemble approach to this problem, we use titles of the documents as queries to calculate the Mean Reciprocal Rank (MRR) as a validation metric for individual techniques and their ensemble.

Business Justification

The volume and variety of textual data being generated in business organizations and online continue to expand exponentially. Looking to improve business processes, knowledge-driven organizations are increasingly turning to data mining using Natural Language Processing (NLP) techniques. This helps them leverage the large collections of unstructured text at their disposal to discover new information or answer specific research questions. Locating relevant documents in a document collection (or corpus) based on a user's query, also known as information retrieval (IR) (Baeza-Yates & Ribeiro-Neto 1999, Cambridge 2009), is a common problem in text analysis. Popular applications of IR include web search, recommender systems and question answering. IR tools can prove quite useful to businesses dealing with logbook data. For example, when a technician or analyst needs to find historical issues or actions related to a custom query, manual search can be tedious and time-consuming depending on the size of the dataset. Another use case where IR tools can be incorporated is in automatic service chatbots which can produce relevant articles and solutions related to an issue raised by a user.

IR is particularly useful for researchers who need to find articles and papers relevant to a search query from a large database of research articles. With the ever-increasing volume of research journals, it can be difficult for scientists and practitioners to keep up with the rapid pace of new publications. As a result of the urgent call to action around the coronavirus pandemic, we decided to work on the COVID-19 Open Research Dataset (CORD19). It is a resource of approximately 200,000 scholarly articles about COVID-19, SARS-CoV-2 and related coronaviruses with the aim to aid researchers who seek complex answers to high-priority scientific questions.

Our solution is based on an ensemble approach that includes varied models, from simple (TF-IDF) to complex (BERT), taking into consideration both pre-trained and fine-tuned models. This combination of multiple NLP models is used to transform documents into vectors, followed by scoring and ranking documents based on their relevance to the search query. This makes our solution more robust than traditional keyword search-based systems by providing users with more contextual results based on semantic similarity.

With the aim to provide near real-time search (within one second) to complex user queries, we wanted an efficient search mechanism in place. Elasticsearch, a popular distributed search engine based on Apache Lucene, was used to search over the multi-dimensional vector space. Along with the retrieved documents, we generated multi-sentence summaries using an extractive text summarizer to make it easier for the users to glean the relevant content. Different modules of our analytical engine are tied together as an end-to-end solution encapsulated into a minimalistic and neat Flask UI, which abstracts the technicalities from the business users.

Methodology

For effective retrieval of relevant documents from a large corpus, the documents are typically transformed into a suitable mathematical representation for comparison. Here, we used algebraic models which represent documents and queries as vectors. The similarity of the query vector and document vector is represented as a numeric score for how well each object in the database matches the query and ranks the objects according to this score. The top scoring documents are returned as the results relevant to the query.

In this section, we describe the major components of our solution, as shown in Figure 1:

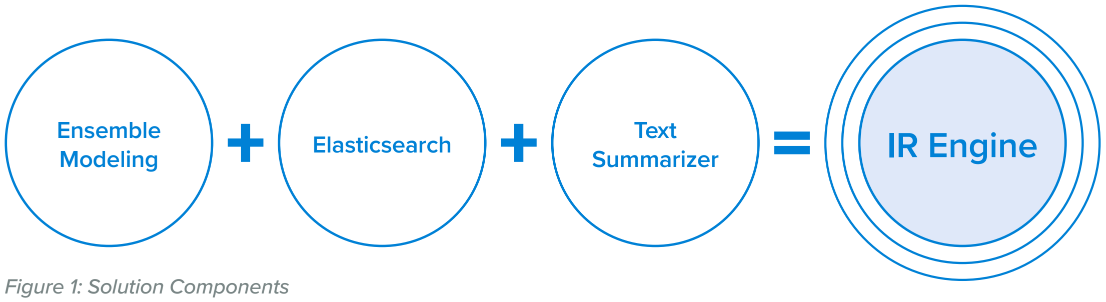


Figure 1: Solution Components

ENSEMBLE MODELING

We used an ensemble of six different NLP models of varying complexity, including both pre-trained and fine-tuned models, thus yielding more contextual results. The cosine similarity between the query and the document vectors is calculated for each of these six models, and the sum of the obtained cosine similarities is the final score for the selection criterion for relevant documents.

- TF-IDF (Term Frequency – Inverse Document Frequency) - Determines the importance of an individual word relative to a document.
- Word2vec (pre-trained model from Google) - Gives word embeddings for individual words.
- BERT (Bidirectional Encoder Representations from Transformers) - Pre-trains deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context.
- SciBERT - Pre-trains BERT model on medical journal articles.
- Topic Modeling (Latent Semantic Indexing) - Assigns particular words to clusters (or word clouds) that contain similar terms.
- Doc2vec - Gives document-level vectors (an enhanced version of Word2vec algorithm).

Due to the lack of labels and overall understanding of the coronavirus literature, we selected an ensemble approach instead of relying on a single IR methodology. We observed that all six methods gave contextually similar results yet provided different documents for the same query and thus justified the use of multiple models. An ensemble made the solution more robust if one or more techniques did not perform well for a query.

ELASTICSEARCH

To implement near real-time search over the document corpus, we incorporated Elasticsearch, a distributed, open source search and analytics engine for all types of data. Elasticsearch uses a data structure called an inverted index which is designed for very fast full-text searches. During the indexing process, Elasticsearch stores documents (and their corresponding vectors generated from different techniques) in JavaScript Object Notation (JSON) format and builds an inverted index to make the document data searchable in near real-time. The latency from the time a document is indexed until it becomes searchable is very short, typically within one second.

TEXT SUMMARIZER

To generate a concise summary of the top results from the ranked documents, we used extractive text summarization which chooses representative sentences in documents and automatically generates a shorter version of a document or a collection of documents (Figure 2).

We used TextRank, a graph-based algorithm (Mihalcea & Tarau 2004), which applies a variation of PageRank over a graph constructed specifically for the task of summarization. It produces a ranking of the elements in the graph. The most important elements are the ones that better describe the text. This approach allows TextRank to build summaries without a training corpus or labeling and allows the use of the algorithm with different languages.

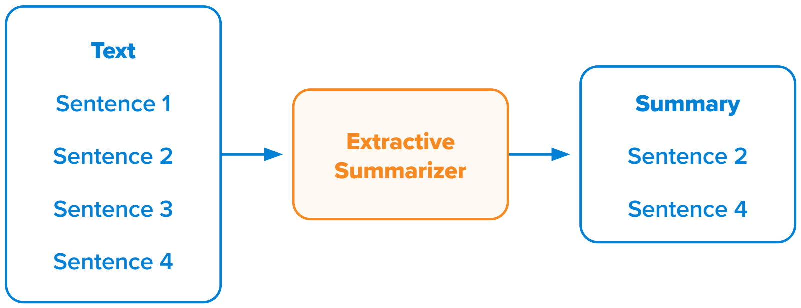


Figure 2: Extractive Summarization

Implementation and Results

EXPERIMENTAL SET-UP

The solution was built in the WWT Advanced Technology Center (ATC) environment on a Linux machine equipped with a 2x Intel Xeon CPU E5-2640 v2 in a containerized fashion. The development was done on Jupyter Notebook inside a docker container. We used the following software and Python packages for our development:

- Python 3
- Docker
- Elasticsearch
- Jupyter Notebook
- Pandas
- nltk
- gensim
- flask
- sentence-transformers
- Pytorch

DATA

Our dataset is the COVID-19 Open Research Dataset (CORD-19), which is a resource of about 200,000 periodically updated scholarly articles about COVID-19, SARS-CoV-2 and related coronaviruses. It is hosted on Kaggle as part of a research challenge to apply text and data mining approaches to find answers to questions within and connect insights across this content in support of the ongoing COVID-19 response efforts worldwide. For the initial preprocessing, we dropped duplicates by both ”title” and ”abstract” and removed entries with null values in these fields. Additionally, we performed fuzzy matching, removing entries with a title almost identical with another, to avoid getting quasi-duplicate results. We then extracted the required fields from the data, namely cord_uid, title, and abstract, and stored this separately as a list of 103,842 valid documents in JSON format.

EMBEDDINGS GENERATION USING ENSEMBLE

Once we had our document corpus (list of valid abstracts), we generated vector embeddings for every abstract using different models from our ensemble. Since the generation of embeddings for the entire document corpus every time would be redundant, we stored the generated embeddings and their requisite trained models as pickle files. They could later be loaded directly. To generate embeddings for the search query, we loaded the saved models to transform the query into a vector of the same size as that of the documents. By saving the models and entire corpus vectors, we avoided having to train our models every time for a new query and saved significant time. Some of these models required the documents to be “cleaned” before they were passed as input. This involved removing everything except alphabetic characters, removing stop words and making all text lowercase. The graphic below describes the methods and classes used to implement each model in Python.

TF-IDF

- Termwise tf-idf weights using TfidfModel
- LsiModel to transform BOW format to a 300-dimensional vector

Topic Modeling

- TfidfVectorizer to get the document-term matrix
- TruncatedSVD to reduce the dimensionality of vectors to number of topics

BERT

- Pre-trained model from Google
- SentenceTransformer to load model and encode docs

Word2vec

- Pre-trained model on Google news loaded using KeyedVectors
- Word Centroid Similarity method (Galke et al. 2017) for document-level embeddings

Doc2vec

- Doc2Vec model trained on corpus
- Document embeddings obtained from *docvecs* method

SciBERT

- Pre-trained model (scibert_scivocab_uncased)
- Pytorch Hugging Face library to load the model and generate embeddings

Figure 3: Methods and Classes

INDEXING AND DOCUMENT SCORING IN ELASTICSEARCH

The next step was to store all documents of the corpus and their corresponding vector embeddings (from all models) as an index in an Elasticsearch cluster. Since every document is stored in a JSON format, we needed to specify the document schema in a separate file (index.json). The schema file describes the fields that every document will contain, including cord_uid, title, abstract, and the separate fields to store vector representation of that document from each technique (field type of *dense_vector* in Elasticsearch and the value of attribute “dims” matches the vector size for the corresponding technique). The documents were indexed in a batch-wise manner. This also helped us keep track of how many documents had been indexed. Once indexing was done, the corpus was ready to be searched.

When the user provides the search query as an input, the vector embeddings for the query are generated from each model in the ensemble. A scoring function in Elasticsearch (*script_score*) was defined which decides how the overall score for ranking the documents based on their similarity to the query would be calculated. In our ensemble model, we used the sum of cosine similarities between the vector embeddings of the document and the query from each model. A parameter called SEARCH_SIZE determined the number of top relevant documents that would be returned in the search response.

GENERATING SUMMARIES USING TEXT SUMMARIZER

To get a summary of the top relevant results, we concatenated the returned abstracts and passed it as input to the gensim summarizer package, which took in a "word_count" parameter to control the length of the summary. We provided a simple Python Flask-based UI, where the user could enter their query in the search which outputs top n relevant abstracts (along with their titles) and concise summary displayed back to the user. Figure 4 shows the entire architecture of our solution, and a snapshot of our Flask UI is shown in Figure 5.

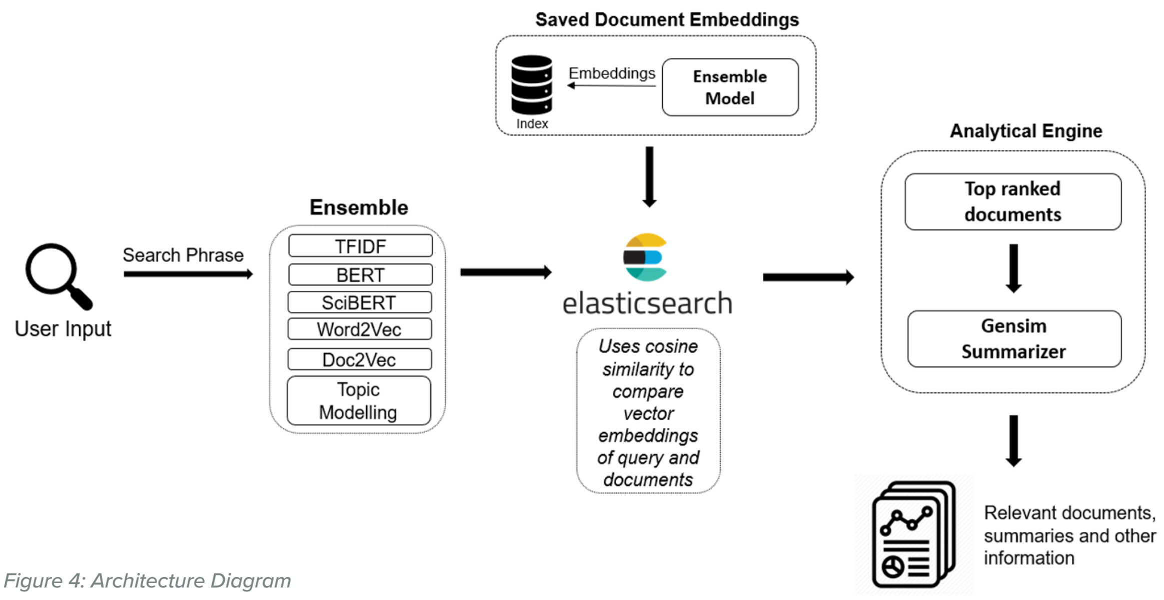
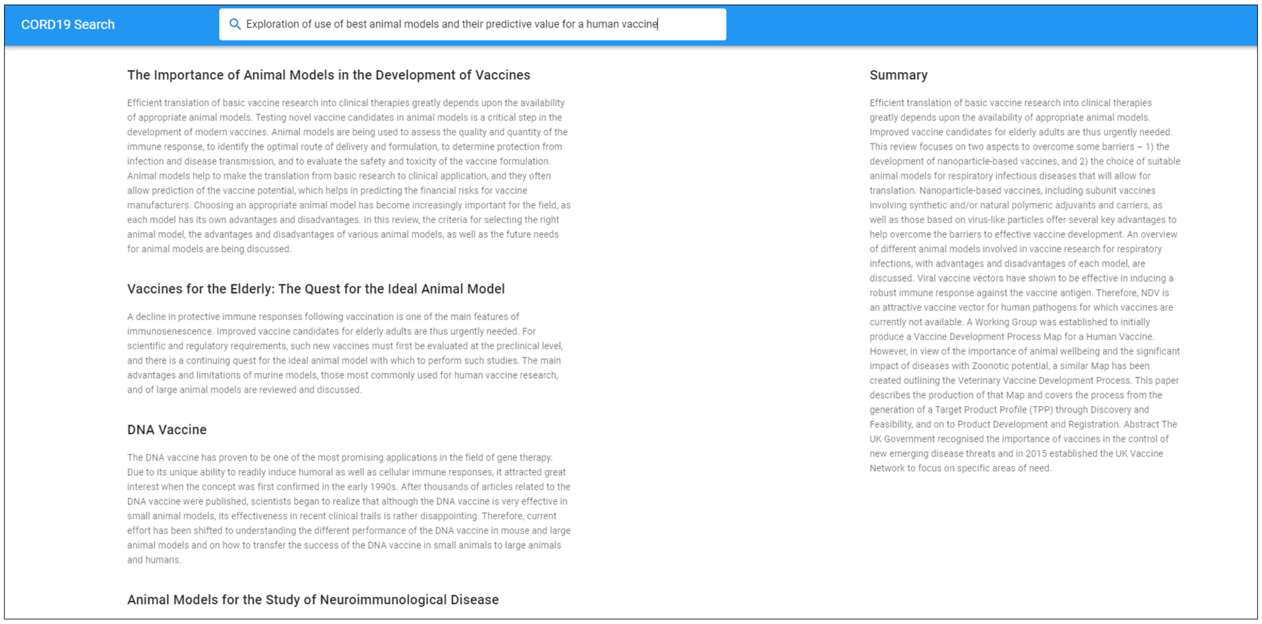


Figure 4: Architecture Diagram


CORD19 Search
Exploration of use of best animal models and their predictive value for a human vaccine

The Importance of Animal Models in the Development of Vaccines

Efficient translation of basic vaccine research into clinical therapies greatly depends upon the availability of appropriate animal models. Testing novel vaccine candidates in animal models is a critical step in the development of modern vaccines. Animal models are being used to assess the quality and quantity of the immune response, to identify the optimal route of delivery and formulation, to determine protection from infection and disease transmission, and to evaluate the safety and toxicity of the vaccine formulation. Animal models help to make the translation from basic research to clinical application, and they often allow prediction of the vaccine potential, which helps in predicting the financial risks for vaccine manufacturers. Choosing an appropriate animal model has become increasingly important for the field, as each model has its own advantages and disadvantages. In this review, the criteria for selecting the right animal model, the advantages and disadvantages of various animal models, as well as the future needs for animal models are being discussed.

Vaccines for the Elderly: The Quest for the Ideal Animal Model

A decline in protective immune responses following vaccination is one of the main features of immunosenescence. Improved vaccine candidates for elderly adults are thus urgently needed. For scientific and regulatory requirements, such new vaccines must first be evaluated at the preclinical level, and there is a continuing quest for the ideal animal model with which to perform such studies. The main advantages and limitations of murine models, those most commonly used for human vaccine research, and of large animal models are reviewed and discussed.

DNA Vaccine

The DNA vaccine has proven to be one of the most promising applications in the field of gene therapy. Due to its unique ability to readily induce humoral as well as cellular immune responses, it attracted great interest when the concept was first confirmed in the early 1990s. After thousands of articles related to the DNA vaccine were published, scientists began to realize that although the DNA vaccine is very effective in small animal models, its effectiveness in recent clinical trails is rather disappointing. Therefore, current effort has been shifted to understanding the different performance of the DNA vaccine in mouse and large animal models and on how to transfer the success of the DNA vaccine in small animals to large animals and humans.

Animal Models for the Study of Neuroimmunological Disease

Summary

Efficient translation of basic vaccine research into clinical therapies greatly depends upon the availability of appropriate animal models. Improved vaccine candidates for elderly adults are thus urgently needed. This review focuses on two aspects to overcome some barriers – 1) the development of nanoparticle-based vaccines, and 2) the choice of suitable animal models for respiratory infectious diseases that will allow for translation. Nanoparticle-based vaccines, including subunit vaccines involving synthetic and/or natural polymeric adjuvants and carriers, as well as those based on virus-like particles offer several key advantages to help overcome the barriers to effective vaccine development. An overview of different animal models involved in vaccine research for respiratory infections, with advantages and disadvantages of each model, are discussed. Viral vaccine vectors have shown to be effective in inducing a robust immune response against the vaccine antigen. Therefore, NDV is an attractive vaccine vector for human pathogens for which vaccines are currently not available. A Working Group was established to initially produce a Vaccine Development Process Map for a Human Vaccine. However, in view of the importance of animal wellbeing and the significant impact of diseases with Zoonotic potential, a similar Map has been created outlining the Veterinary Vaccine Development Process. This paper describes the production of that Map and covers the process from the generation of a Target Product Profile (TPP) through Discovery and Feasibility, and on to Product Development and Registration. Abstract The UK Government recognised the importance of vaccines in the control of new emerging disease threats and in 2015 established the UK Vaccine Network to focus on specific areas of need.

Figure 5: Snapshot of IR Engine

VALIDATION OF RETRIEVED DOCUMENTS

Each of the models in the ensemble would prioritize retrieved documents differently. So, a question is: how would we know which model is performing better? Or, how could we be sure that the final ensemble model gave more reasonable results than the individual models? A challenge that we faced here is the absence of ground truth, i.e., labels related to the relevance of the results of a given query. Since the search query could be virtually anything, it was impossible to have relevant scores/labels for each document-query pair (even if one possessed the proper domain knowledge). This made using standard evaluation metrics utilized in IR impossible here. A novel workaround for this was using the title of a document as the search query and finding the rank of that specific document in the top n (e.g., 20) results. This approach offered an empirical proxy to the ground truth. We randomly selected Q documents, and their titles, from the corpus to serve as our "validation set." The size Q was chosen to be large enough to cover the most typical queries. Each of those Q titles now became a fiducial query phrase. Since we were retrieving 20 documents, we chose Q to be around 1,000. The titles of those Q randomly selected documents served as good proxies of eventual queries that users might enter in production. With that, we could calculate Mean Reciprocal Rank (MRR), which is the average of the reciprocal ranks (1 for first place, 1⁄2 for second place, 1⁄3 for third place and so on) of results for the set of Q queries. Note that if the specific document did not appear in the top n results, the corresponding reciprocal rank would be 0. See figure 6 below.

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{rank_i}$$

Figure 6: Mean Reciprocal Rank formula

MRR could therefore serve as a KPI (Key Performance Indicator), much like AUC (Area Under the Curve), R2 (Coefficient of Determination), or MAPE (Mean Average Percentage Error) that are commonly used to measure model goodness. The higher the value of MRR, the better the model.

RESULTS AND DISCUSSION

Using a query set of 1,000 randomly chosen titles, we tabulated the following values from the individual models and their combination as an ensemble:

- MRR values from the top 20 results obtained.
- Average time taken to generate vector embeddings for the query.
- Average time taken to search for the top 20 relevant documents for the query using cosine similarity in Elasticsearch.

Table 1: Model Evaluation

Models	Mean Reciprocal Rank (MRR)	Avg. query embedding time (milliseconds)	Avg. search time (milliseconds)
Topic Modeling	0.073	7.50	44.67
TF-IDF	0.175	6.73	67.64
BERT	0.110	58.61	122.61
Word2Vec	0.120	3.88	67.69
SciBERT	0.050	56.21	121.82
Doc2Vec	0.116	4.62	65.93
Ensemble *(all of the above)*	**0.299**	**126.76**	**325.82**
Best Combination *(TF-IDF+BERT+ Word2vec+ Doc2vec)*	**0.327**	**73.04**	**226.44**

Below are our observations:

- The ensemble showed a significant improvement in MRR over the individual models with a trivial expense of increased query embedding generation and search time. Most of the individual models had an effective MRR of around 0.1. Whereas, the ensemble showed an MRR close to 0.3. Thus, the ensemble model provided a large predictive lift and more contextual results compared to the individual models.
- Excluding the low MRR valued models (SciBERT and Topic Modeling) from our ensemble, we got a slight improvement in the MRR value and reduction in the query embeddings and search time.

Conclusion

We have presented an ensemble approach to the IR problem using a combination of multiple NLP models to score and rank documents in a corpus based on their relevance to a search query. We used a variety of models, from simple (TF-IDF) to complex (BERT), and included both pre-trained and fine-tuned models. Therefore, we allowed for more contextual results and made the solution more robust. In the absence of relevance labels/scores for document-query pairs, we used MRR as a validation metric using the titles of the documents as queries to search. We showed that our ensemble performs better than all individual models, with comparable vector embedding generation and search times.

We engineered an optimized end-to-end solution by storing corpus embeddings (one-time process) to eliminate the redundancy, incorporating Elasticsearch, which helped us to boost the search response time significantly, and adding a summarizer module to provide a succinct summary of the top relevant results. All these components were condensed together and abstracted as a Python Flask UI where the user could enter a search query and get the relevant results and the summary displayed in near real-time.

In future work, this ensemble model can be tested on a dataset that contains relevance labels/scores for a set of document-query pairs, and standard validation metrics like Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) can be used to evaluate the performance of the ensemble vis-à-vis the individual models. This can also help overcome a possible limitation of calculating MRR using titles as queries, where some abstracts have vaguely worded or very short titles. Other summarizer models, like BertSum, can be explored, which can be fine-tuned on the corpus to give more contextual summaries (Liu 2019).

References

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval* (Vol. 463). New York: ACM press.

Cambridge, U. P. (2009). Introduction to information retrieval.

Hoi, S. C., & Jin, R. (2008). Semi-supervised ensemble ranking.

Wang, Y., Choi, I. C., & Liu, H. (2015). Generalized ensemble model for documentranking in information retrieval. *arXiv preprint arXiv:1507.08586.*

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188-1196).

Beltagy, I., Lo, K., & Cohan, A. (2019). SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676.*

Kusner, M., Sun, Y., Kolkin, N., & Weinberger, K. (2015, June). From word embeddings to document distances. In *International conference on machine learning* (pp. 957-966).

Galke, L., Saleh, A., & Scherp, A. (2017). Word embeddings for practical information retrieval. *INFORMATIK 2017.*

Mihalcea, R., & Tarau, P. (2004, July). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing* (pp. 404-411).

Liu, Y. (2019). Fine-tune BERT for extractive summarization. *arXiv preprint arXiv:1903.10318.*

World Wide Technology

ARTIFICIAL INTELLIGENCE
RESEARCH & DEVELOPMENT

Learn more about WWT's Artificial Intelligence Research & Development at:
https://www.wwt.com/artificial-intelligence-research-and-development

ABOUT WWT ARTIFICIAL INTELLIGENCE RESEARCH & DEVELOPMENT

The Artificial Intelligence Research & Development program at WWT is an applied research initiative focused on investigating the one to three-year horizon of the artificial intelligence space. The AI R&D program conducts internal projects grounded in WWT's deep understanding of industry use cases and produces reusable components and white papers to share with customers and the AI community.

Through strategic partnerships, cutting-edge data science and ML Ops skills, and ability to test and learn in the Advanced Technology Center and public cloud, the AI R&D team advances WWT's knowledge of the AI and ML space, thus allowing WWT to be on the bleeding edge and remain strategic advisors to businesses in their AI needs.

ABOUT WWT

Founded in 1990, WWT has grown to become a global technology solution provider with $12 billion in annual revenue. With thousands of IT engineers, hundreds of application developers and unmatched labs for testing and deploying technology at scale, WWT helps customers bridge the gap between IT and business. By bridging leading technology companies together in a physical yet virtualized environment through its Advanced Technology Center, WWT integrates individually impressive technologies to produce game-changing solutions.

Based in St. Louis, WWT employs more than 6,000 employees and operates more than 4 million square feet of warehousing, distribution and integration space in more than 20 facilities throughout the world.