BM25-WEIGHTED BERT-BASED EMBEDDING OF PUBMED

Luca Cappelletti **Tommaso Fontana** **Justin Reese**

ABSTRACT

Knowledge graphs provide a natural representation of biomedical data and have been applied to use cases such as graph machine learning to identify possible drug repurposing candidates and genes related to human disease. For these use cases, it is helpful to incorporate data from scientific literature, which contains rich information about entities (e.g. genes, diseases) in these knowledge graphs. However, it is challenging to integrate this information into knowledge graphs. In this work, we propose a procedure based on Hugging Face Tokenizer, BERT-based embeddings and BM25 to convert articles into vector features. We share the BERT, SciBERT and Specter embedding and textual informations from Internet Archive for 33M articles and provide a Python package, pubmed_embedding, to query them easily.

***K*eywords** Transformers · BM25 · PubMed

1 Introduction

Knowledge graphs (KGs) are a powerful tool for representing and organizing large amounts of data. They have been applied in various domains, including the biomedical field, where they have been used to identify potential drug repurposing candidates and genes related to human disease [1]. One of the key challenges in building KGs is incorporating information from scientific literature, which contains rich textual details about entities such as genes and diseases.

This paper proposes a method for computing meaningful embeddings of large corpora of scientific articles, and a procedure to incorporate this information into KGs. Our approach is based on natural language processing (NLP) techniques, including the Hugging Face Tokenizer, BERT-based embeddings, and BM25. We use these tools to convert articles into vector features, which can then be used as node features for many tasks. To make our results widely accessible, we share the BERT [2], SciBERT [3], and Specter [4] embedding and textual information for $33,348,342$ articles. We also provide a Python package, pubmed_embedding, which allows users to quickly query these embeddings and incorporate the information they contain into their own KGs.

Our method provides a convenient and effective way to incorporate information from scientific literature into KGs, enabling a more accurate and comprehensive representation of biomedical data.

2 Procedure for generating embedding

We describe here a procedure for generating BM25-weighted BERT-based embeddings for text data from PubMed articles. This process involves retrieving the PubMed documents in XML format, training a BERT tokenizer model using the GRAPE [5] library, and using BM25 to compute a weighted average of the embedding for each token. The resulting embeddings are represented using half-precision floats that tally up to around 50 GB. The embeddings are split into chunks of 50000 articles to allow the loading of just the used chunks on computers with limited RAM. A Python package called pubmed_embedding is available on Pypi for retrieving embeddings for any given PubMed ID.

Data retrieval We start by retrieving the PubMed articles and metadata in XML format. We have implemented a Rust parser to convert the XML documents into a tab-separated document with the PubMed ID, its title and abstract, and other available textual information.

$$\textit{BM25}(D,q) = \ln\left(1+\frac{N-n_q+0.5}{n_q+0.5}\right)\cdot\frac{f(q,D)\cdot(k_1+1)}{f(q,D)+k_1\cdot\left(1-b+b\frac{|D|}{\textit{average document length}}\right)}$$

Figure 1: **BM25 score:** Given a document $D \in \mathbb{D}$ and a word in the vocabulary q, the weighting schema uses the number of documents where the word q appears n_q, the number of times the word q appears in the document D, $f(q, D)$, a free parameter b, generally equal to $3/4$ and another free parameter k_1, which is usually selected in range $[1.2, 2.0]$. In out case $b = 0.75$ and $k_1 = 1.5$. These parameters may be fine tuned using optimization schemas such as Bayesian Optimization.

Tokenization We proceed using the pipeline provided by the GRAPE library [5] to train a BERT tokenizer model [2] for the BERT-based model. A BERT transformer model is a deep learning model trained to perform natural language processing tasks such as language translation and text classification. BERT stands for "Bidirectional Encoder Representations from Transformers," which refers to the model that uses a transformer architecture to process input text in a bidirectional manner, taking into account the context of words in a sentence. This allows BERT to perform better on many natural language processing tasks than previous models that only considered the context of words in a single direction. We have employed SciBERT [3], Specter [4] and the original BERT [6]. SciBERT and Specter are possibly the best-suited pre-trained language models for embedding the articles' textual information, as the authors have trained them primarily on a sizeable multi-domain corpus of scientific publications. Still, the procedure can be readily re-executed with any BERT-based language model and generalized for any model-producing textual embedding.

BM25-weighted averages BERT-based models produce embedding for any vocabulary token, i.e. word fragments. Each PubMed article contained a variable number of tokens. We can average the embeddings for the tokens to obtain a single embedding for each article. However, if we employ a simple mean, recurrent terms such as "disease" or "gene" may overwhelm the embedding of less frequent terms that characterize each specific article. To address this issue, we employ BM25 [7] which is a scoring function used in information retrieval to calculate the relevance of a document to a query. It is based on the frequentist probability that an article contains a query term, the length of the article, and the average length of articles in the collection. TFIDF, or term frequency-inverse document frequency, is a weighting scheme used to calculate the importance of a term in an article compared to a collection of articles. It considers the frequency of a term in the article and inverse document frequency, which is the logarithm of the ratio of the total number of documents to the number of documents that contain the term.

The BM25 weighting scheme can be used to effectively weight queries of large sets of articles, allowing for more relevant results to be returned to the user. These approaches are effectively a bag-of-words retrieval function that ranks a set of articles based on the query terms appearing in each, regardless of their proximity within the article. We report the formula of BM25 in figure 1, illustrating how to compute the score of a token q relative to a document D in a collection of documents.

We compute the frequency of each token across PubMed, and for each token and article, we obtain a BM25 score, representing how well each token characterizes that article. We use these scores to compute a weighted average, thus obtaining an embedding that better captures the topics of each article.

Half-precision and data chunks We provide a Python package called pubmed_embedding, readily available from PyPi, that allows querying and retrieval of the provided embeddings. Since we have chunked the embedding in smaller pieces of 50K articles each and 74MB, in most cases, a user will only have to download a subset of the embeddings when querying a small portion of the overall dataset.

3 Software and datasets

The procedure described above n provides textual information (title and abstract) and pre-computed embeddings for over 33M PubMed articles. In addition, we offer BM25-weighted embeddings for each article based on pre-trained BERT, SciBERT, and Specter models. These embeddings are available from our Python package, "pubmed_embedding", which allows users to easily query the dataset for a subset of these articles. The embeddings are provided in small chunks to make it more manageable for users to download and query. Example use of the package is shown in figure 2.

Textual features We provide the textual information associated with $33,348,342$ PubMed articles in TSV format, as made available from the PubMed XML data, which generally include title and abstract.

```python
from pubmed_embedding import get_pubmed_embedding_from_curies

pubmed_ids = ["PMID:24774509", "PMID:15170967", "PMID:7850793"]

get_pubmed_embedding_from_curies(
    curies=pubmed_ids,
    version="pubmed_bert_30_11_2022"
)
```

	0	1	2	3	4	5	6	
PMID:24774509	-0.000191	-0.000306	-0.000297	-0.000329	-0.000150	-0.000201	-0.000268	-0.00026
PMID:15170967	-0.000482	-0.000739	-0.000701	-0.000910	-0.000242	-0.000342	-0.000647	-0.00042
PMID:7850793	-0.000182	-0.000324	-0.000217	-0.000406	-0.000027	-0.000204	-0.000286	-0.00018

3 rows × 768 columns

```python
get_pubmed_embedding_from_curies(
    curies=pubmed_ids,
    version="pubmed_scibert_30_11_2022"
)
```

	0	1	2	3	4	5	6	
PMID:24774509	0.000020	-0.000057	0.000023	-0.000041	-0.000081	-0.000276	0.000031	-0.00005
PMID:15170967	-0.000143	-0.000093	0.000045	-0.000017	-0.000063	-0.000215	-0.000217	-0.00031
PMID:7850793	-0.000035	-0.000132	0.000071	-0.000051	0.000061	0.000004	-0.000060	-0.00005

3 rows × 768 columns

```python
get_pubmed_embedding_from_curies(
    curies=pubmed_ids,
    version="pubmed_specter_30_11_2022"
)
```

	0	1	2	3	4	5	6	
PMID:24774509	-0.000147	-0.000009	-0.000097	-0.000159	-0.000156	-0.000171	0.000060	-0.00008
PMID:15170967	-0.000325	-0.000001	-0.000156	-0.000164	-0.000439	-0.000202	-0.000022	-0.00020
PMID:7850793	-0.000185	0.000166	0.000060	-0.000046	-0.000166	-0.000165	0.000108	-0.00021

3 rows × 768 columns

Figure 2: Examples of using the pubmed_embedding Python package to query the pre-computed BM25-weighted embeddings for PubMed articles based on the pre-trained BERT, SciBERT, and Specter models. The package can be used to easily access the 768-dimensional article embeddings, which are provided in small chunks for efficiently querying and downloading the data of interest.

Embedding Using the procedure above, we have computed and share BM25-weighted embedding based on the pre-trained BERT, SciBERT and Specter models. We provide 768-dimensional embeddings for 33M articles in Numpy format.

Python Package We provide a Python package called pubmed_embedding that allows to query the provided embedding. Since we have chunked the embedding in smaller pieces of 50K articles each and 74MB, in most cases, a user should only have to download part of the embedding when querying a small portion of the overall dataset. The Python package is readily available from PyPi. Usage examples are shown in figure 2.

4 Applications

This section briefly describes the process of connecting nodes in a knowledge graph (KG) to relevant articles and computing their features. These approaches will enable us to more accurately connect KG nodes to relevant articles and improve the overall quality of the KG.

Connecting nodes to relevant articles Whenever there exists even minimal textual information associated with nodes in a KG, it is possible to compute BM25-weighted BERT-based embedding for those nodes by using the same tooling we provided used to calculate the PubMed embeddings. Once the node textual embedding is computed, one can identify related PubMed articles by employing similarity metrics such as the cosine similarity (our package provides a parallel Rust pairwise version with Python binding to facilitate this calculation). Alternatively, more complex similarity measures may be employed. We provide examples for the computation of these embedding in the project GitHub repository.

5 Conclusions

This paper presents a new method for incorporating information from scientific articles into knowledge graphs. Our approach is based on natural language processing techniques, including the Hugging Face Tokenizer, BERT-based embeddings, and BM25. We applied our method to a dataset of 33 million PubMed articles and showed that it could effectively convert articles into vector features that can be easily integrated into knowledge graphs. We also provide a Python package, pubmed_embedding, that allows users to easily query our BERT, SciBERT, and Specter embeddings and textual information from the articles. Our method can be used to improve the quality and comprehensiveness of knowledge graphs in the biomedical domain and potentially other fields.

In future works, we will apply and evaluate the effectiveness of these features in a range of KG-related tasks, and provide embedding for other large textual databases.

References

[1] Justin T. Reese, Deepak Unni, Tiffany J. Callahan, Luca Cappelletti, Vida Ravanmehr, Seth Carbon, Kent A. Shefchek, Benjamin M. Good, James P. Balhoff, Tommaso Fontana, Hannah Blau, Nicolas Matentzoglu, Nomi L. Harris, Monica C. Munoz-Torres, Melissa A. Haendel, Peter N. Robinson, Marcin P. Joachimiak, and Christopher J. Mungall. Kg-covid-19: A framework to produce customized knowledge graphs for covid-19 response. *Patterns*, 2(1):100155, 2021.

[2] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[3] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[4] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. Specter: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180*, 2020.

[5] Luca Cappelletti, Tommaso Fontana, Elena Casiraghi, Vida Ravanmehr, Tiffany J. Callahan, Marcin P. Joachimiak, Christopher J. Mungall, Peter N. Robinson, Justin Reese, and Giorgio Valentini. Grape: fast and scalable graph processing and embedding, 2021.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. Gaithersburg, MD: NIST, January 1995.