Finding Related ArXiv Papers with BERT

CS224U Project - Experiment Protocol

Junshen Kevin Chen
Stanford University
jkc1@stanford.edu

1 Introduction

In this project, I propose to design a system that evaluates the level of relation between two academic papers (classification), and suggest possible related work given a text abstract as query (retrieval) using semantic information, in hopes that it would augment the current keyword-matching-based academic search engines such as Google Scholar.

I propose to use a pre-trained BERT (Devlin et al., 2018) (and possibly other pre-trained deep transfomer models if time permits) for this task, fine-tuning and building a model that encodes the text of each paper in the corpus into a vector representation. Then, given an unseen query, use the same model to encode the query text, and retrieve document(s) related to the query based on vector distances.

2 Hypothesis

I hypothesize that a pre-trained BERT model can be fine-tuned to encode "paper relation" using a corpus of ArXiv paper abstracts, and this model is able to retrieve related articles in the corpus to some accuracy. I further hypothesize the resulting system would have a run time complexity in the seconds, such that it has practical uses in augmenting existing keyword-matching-based academic search engines.

3 Data

3.1 Paper abstracts and citation graph

I use Semantic Scholar's Open Research Corpus (s2-corpus) (Ammar et al., 2018) as my main source of data. This corpus encompasses numerous popular sources of academic research, including ArXiv, IEEE, NeurIPS, etc. It also contains paper from various regions and in various languages. There are a totoal of 184,000,000 items in this corpus.

Each entry in this data set is an academic paper uniquely identified by an ID, and includes metadata information such as the title, fields of study, authors, year of publication, source conference, etc.

The s2-corpus includes the citation graph in the form of the ID's of in-citations and out-citations for each paper. From my investigation, this citation graph is significantly more complete in comparison to the ArXiv citation graph produced by (Clement et al., 2019).

I use the paper abstract text from this corpus, which is the main input of the BERT-based model.

3.2 Paper full text

Because Semantic Scholar is a search engine that only points to contents in other hosts and conferences, the s2-corpus provides no full-text of papers. To augment the models with full texts, I use the ArXiv dataset by (Clement et al., 2019).

3.3 Preprocessing

Before using the aforementioned two corpora, I first run them through a data pipeline to perform the following preprocessing steps:

1. Filtering out articles not in English (based on abstract and title), since BERT is pre-trained in English.
2. Filtering out articles not in ArXiv, since we do not have the full text otherwise.
3. Optionally keeping only articles in Computer Science. I will train a smaller model with CS articles only for a proof-of-concept, then use the full ArXiv corpus to train a bigger model.
4. Removing in-citation or out-citation pointers of articles that are not kept after the previous steps

5. Tokenizing and encoding each abstract text with pre-trained BERT tokenizer, prepending the `[CLS]` token, then trimming to the max BERT length of 512

3.4 Train-dev-test split

After the aforementioned filtering steps, the resultant CS-only, English, ArXiv articles corpus has 157,227 entries. I randomly split this corpus to train, dev, test sets in the proportion of $90 : 5 : 5$.

3.5 Sampling

For the specific models I propose (see section 5), we need training and testing samples in the form of triplets:

$(anchor, positive\_sample, negative\_sample)$

Where a positive sample is an article that either cites or is cited by the anchor, and a negative sample is an article neither cites nor cited by the anchor.

I employ algorithm 1 to sample the triplets:

Algorithm 1: Sample Triplets

```
Input  : $S$, set of all articles
Input  : $A$, set of all articles in the split
Output : $D$, set of article triplets
$D \leftarrow \{\}$
for $a \in A$ do
    for $p \in Citations(a)$ do
        repeat
            $n \leftarrow RandomChoice(S)$
        until $n \notin Citations(a)$;
        $D \leftarrow D \cup (a, p, n)$
    end
end
```

The algorithm is performed on each of train, dev, test set.

4 Metrics

Accuracy We will not use the classical definition of accuracy in a binary classification task as a metric, because the notion of accuracy is not relevant in this task for two reasons:

1. The dataset is sampled not according to real proportions of related / unrelated articles. Since we are sampling at equal proportion, therefore the constructed dev / test set does not reflect the real distribution. Setting a threshold on classifying related / unrelated when given two articles is irrelevant.
2. Suggesting related articles is a document retrieval task. Therefore we care less about the notion of whether two articles are related in the absolute sense, but care more about whether one article is more related to the anchor than another.

However, we can tweak this notion slightly to fit the "document comparison task". Given an anchor sample, a positive candidate and a negative candidata, a trained predictor would output which candidate relates to the anchor more than the other, such that predicting the positive candidate to be more related becomes a correct prediction, and incorrect otherwise. Then, accuracy is defined as:

$$Accuracy = \frac{\sum_{(a,p,n)\in D} \mathbf{1}[rel(a,p) > rel(a,n)]}{\sum_{(a,p,n)\in D} 1}$$

Mean vector distance difference One meaningful metric to compare for different models is vector distance. Since the goal of the model is the reduce vector distance for positive (related) samples, and increase vector distance for negative (unrelated) samples, then the difference between the two across the set is meaningful. We define mean vector distance difference (MVDD) as:

$$MVDD = \sum_{\vec{a}\in\vec{A}} \left(\frac{\sum_{\vec{n}} dist(\vec{a},\vec{n})}{\sum_{\vec{n}} 1} - \frac{\sum_{\vec{p}} dist(\vec{a},\vec{p})}{\sum_{\vec{p}} 1} \right)$$

Where $\vec{a}$ is an encoded vector representation of an article in the split, $\vec{n}$ is an encoded vector representation of an unrelated article to a, $\vec{p}$ is an encoded vector representation of a related article to a. The higher this number the better the model performs.

Where $dist(\vec{u}, \vec{v})$ is a vector distance function we use for the ranking problem. See section 5.

Precision, Recall, F-score @ k Since document retrieval is a recommender system task, we will use precision, recall, and f-score @ k as an evaluation metric. It is defined such that given a recommendation budget k, which is set arbitrarily depending on the application, we retrieve top-k documents related to the anchor a, as a set P (predicted positive documents). Then, we may define the following:

$$Precision@k = \frac{\sum_{\hat{p}\in P} \mathbf{1}[\hat{p} \in Citations(a)]}{\sum_{\hat{p}\in P} 1}$$

$$Recall@k = \frac{\sum_{\hat{p}\in P} \mathbf{1}[\hat{p} \in Citations(a)]}{\sum_{p\in Citations(a)} 1}$$

We may experiment with different k for meaningful results. Then, we use the classical definition of f-score:

$$F_\beta@k = (1+\beta^2)\frac{Precision@k \cdot Recall@k}{\beta^2 Precision@k + Recall@k}$$

Where β is a constant defining ”how many times recall is as important as precision”.

5 Models and General Reasoning

5.1 Baseline Random Predictor

To understand the performance of any trained BERT models, we need to first define some baseline models against which to compare, to get a frame of reference in the metrics.

Intuitively, we can use a random predictor for each of our two tasks, to generate a frame of reference against which to compare the trained model.

For the ”document comparison task”, given an anchor and two candidates, the random predictor would return either candidate randomly.

For the ”document retrieval task”, given an anchor candidate, and a recommendation budget k, the random predictor would randomly sample k documents from the set of all articles S.

5.2 InferSent

InferSent (Conneau et al., 2017) is a semantic sentence encoder developed by researchers at Facebook AI Research. It is pre-trained with GloVe and fastText word embeddings, and encodes sentences of variable length to a fixed sized encoded vector. It is able to achieve high performance in various NLP and NLI tasks.

Intuitively, InferSent encodes sentences such that sentences with similar semantic meanings encodes to vectors with low distance and vice versa. However, article relevance is not strightly higher when they are semantically similar.

I will attempt to use the vanilla InferSent model, and also fine-tune it with triplet loss, an use various vector distance functions to compare their performance. See section 5.4.

5.3 BERT

Finally, I will fine-tune a pre-trained BERT-small model to perform a similar text encoding task, such that the encoded vector of related articles are low in distance and vice versa.

Intuitively, BERT (and any transformer model, for that matter), attends to all positions in the input tokens and therefore is able to deeply extract semantic information among texts. However, BERT is not by-default trained for this specific task of ”document relevance” and therefore needs fine-tuning.

Since BERT outputs embedding vector for each token in the input, I will experiment with the following methods for comparing between articles:

1. Fine-tuning the model and taking the embedding of BERT’s `[CLS]` token.
2. Average-pooling across the word dimension.
3. Max-pooling across the word dimension.

5.4 Vector Distances

For each of the above models (less the baseline), I will experiment with the following vector distance functions to compare their performance:

Euclidean distance squared

$$EuclSq(u,v) = \sum_i (u_i - v_i)^2$$

Cosine distance

$$CosDist(u,v) = \frac{u^T v}{|u||v|}$$

Dot product

$$Dot(u,v) = u^T v$$

The lower the distance, the more similar the articles represented by u and v.

6 Summary of Progress

At the time of writing this document, I have acquired and pre-processed both data sets, and performed all pre-processing with the aforementioned methods. I have also built a dummy classfier to use both as a superclass for classifiers as well as the random predictor baseline.

References

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the literature graph in semantic scholar. In *NAACL*.

Colin B. Clement, Matthew Bierbaum, Kevin P. O'Keeffe, and Alexander A. Alemi. 2019. On the use of arxiv as a dataset.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.