SChuBERT: Scholarly Document Chunks with BERT-encoding boost Citation Count Prediction

Thomas van Dongen, Gideon Maillette de Buy Wenniger, and Lambert Schomaker

Bernoulli Institute for Mathematics,Computer Science and Artificial Intelligence
University of Groningen, Groningen, The Netherlands
t.a.van.dongen AT student.rug.nl
gemdbw AT gmail.com l.r.b.schomaker AT rug.nl

Abstract

Predicting the number of citations of scholarly documents is an upcoming task in scholarly document processing. Besides the intrinsic merit of this information, it also has a wider use as an imperfect proxy for quality which has the advantage of being cheaply available for large volumes of scholarly documents. Previous work has dealt with number of citations prediction with relatively small training data sets, or larger datasets but with short, incomplete input text. In this work we leverage the open access ACL Anthology collection in combination with the Semantic Scholar bibliometric database to create a large corpus of scholarly documents with associated citation information and we propose a new citation prediction model called SChuBERT. In our experiments we compare SChuBERT with several state-of-the-art citation prediction models and show that it outperforms previous methods by a large margin. We also show the merit of using more training data and longer input for number of citations prediction.

1 Introduction

Predicting the quality of scientific articles is a novel task in the field of deep learning. There are many indicators of quality such as whether a paper was accepted or rejected, meta-information such as the author's h-index(es), and the number of citations. The number of citations, while not a perfect indicator of quality, is available for any paper which makes it suitable for constructing a large dataset. In this work we propose ACL-BiblioMetry, a new dataset consisting of 30000 papers with citation information. We also test several state-of-the deep learning models and propose a new model called SChuBERT which outperforms all other methods.

Using the full text of scholarly documents has the potential to substantially improve the performance of the citation count prediction task. But prohibitive memory costs of applying advanced deep learning models on the full text can be a roadblock. In particular, BERT (Devlin et al., 2018) and its variants have been very successful as building blocks for state-of-the-art natural language processing models for many tasks. Citation count prediction for scholarly documents is a task where BERT has clear potential as well. However, scholarly documents are particularly long texts in general. Since BERT has a time complexity that is quadratic with respect to the input length, it is limited to 512 tokens by default, a limit which can not be increased by much without causing prohibitive computational cost.

Recent models including the Reformer (Kitaev et al., 2020) and Longformer (Beltagy et al., 2020) have sought to overcome the quadratic computational cost of the Transformer model (Vaswani et al., 2017) underlying BERT. While these models are very promising, they do not offer the unsupervised pre-training on large amounts of data that makes BERT so powerful as of yet. Although in principle these models could be applied as a drop-in replacement for BERT, it requires more research to show if and how unsupervised pre-training as done in BERT can be made to work well with very long context. For these reasons, in this work we use BERT as our base building block and find effective ways to overcome its input length limit, leaving experimentation with the aforementioned models for future research.

For dealing with large amounts of training examples containing very long input text we need an approach that: 1) Is able to fit the encoding of the long text into memory, 2) can efficiently process the large amount of training examples when training over many epochs. Both requirements can be fulfilled by chunking the long input text of our examples into parts, and pre-computing BERT embeddings for each of these parts using a pre-trained

BERT model. The core of the final model is a sequence-model, in particular a gated recurrent unit (GRU) (Cho et al., 2014), which directly uses the pre-computed chunk embeddings as inputs. This approach simultaneously overcomes the memory problems associated with dealing with very long input texts, as well as achieves high computational efficiency by performing the expensive step of computing BERT embeddings for chunks only once.

While the task of citation count prediction using the contents of a scholarly document is not new, and goes back at least to the work of Fu and Aliferis (2008), work up until now has been limited in: a) the size of the training data, b) the size of the input text. Table 1 gives an overview of data used in earlier work, note that most are restricted by using only the title + abstract as well as a small number of examples, while (Maillette de Buy Wenniger et al., 2020) substantially increase the number of examples but still use only a limited part of body text available from S2ORC (Lo et al., 2019). In this work, we show that both these factors have a large influence on the accuracy of models predicting citation counts. Essentially, state-of-the-art methods cannot be adequately evaluated with too small training data. Therefore, apart from providing state-of-the art results for citation-count-prediction on a data set currently unmatched in terms of number of examples with full length input text, we also provide the code for other researchers to rebuild our dataset and the methodology of citation count prediction using the semantic scholar database to label new collections of scholarly documents.

The rest of the paper is organized as follows: in section 2 we discuss related work, in section 3 we describe the models used for citation count prediction, in section 4 we discuss the dataset construction, in section 5 we present our experiments, in section 6 we show our results and in section 7 we end with conclusions.

2 Related Work

Recently, multiple datasets have been released which are useful for the scientific quality prediction problem. The S2ORC dataset (Lo et al., 2019) has abstract information for 81.1M papers and full-text for 8.1M papers, both with citations.

Other large datasets exist such as unarXive and PubMed Central Open Access Subset, but these datasets span various domains. Given the difficulty of the citation prediction task, we made a new dataset for just the computational linguistics and natural language processing domain, to be used as a benchmark for citation prediction models.

The PeerRead dataset (Kang et al., 2018) is another useful dataset that has accept/reject decisions for 14.7K full-text papers. This is a useful dataset on which more research has been performed (Shen et al., 2019), but the amount of papers in it is fairly limited. For this reason, we propose our new dataset which contains full-text and citations for a large number of papers.
A number of methods have been proposed for the citation prediction problem. (Brody et al., 2006) try to predict future citations of a paper by using web usage statistics, e.g. the number of times the paper was downloaded. (Abrishami and Aliakbary, 2019) use deep learning techniques to predict long-term citations using short-term citations. (Bai et al., 2019) use a measure called Paper Potential Index (PPI) which is based on a combination of features such as the impact of the authors and early citations. The problem with these methods is that information such as short-term citations and web usage statistics are only available after the paper is published. Furthermore, these methods disregard any of the actual papers' content. Because of this reason, our work focuses on predicting the citations using only the textual content. Limited research is available on this topic. One of the first papers which focused on predicting citation count by only using information available at publication is by (Fu and Aliferis, 2008). They use the paper title, abstract and keywords as well as bibliometric information as input data for an SVM. They then predict a binary label (positive or negative) based on whether the paper received at least a set number of citations within 10 years. This work was expanded upon by (Ibáñez et al., 2009). They predict a discrete value (few, some or many citations) using multiple classification models, outperforming the baseline set by (Fu and Aliferis, 2008) using both naive Bayes as well as logistic regression. Both papers use a fairly small dataset (3788 papers for (Fu and Aliferis, 2008) and 2246 papers for (Ibáñez et al., 2009)).

3 Models

In this section we briefly describe our two baseline models: BiLSTM and hierarchical attention networks (HANs). This is followed by a description of the BERT-based SChuBERT model, to the best of our knowledge first applied to the task of citation count prediction in this work.

3.1 BiLSTM Based Prediction

Our BiLSTM baseline model, is a re-implementation of the BiLSTM model introduced in (Shen et al., 2017) . This model was initially used for the task of Wikipedia text quality prediction and applied also in (Shen et al., 2019) for the task of accept/reject prediction on the PeerRead dataset, and finally in (Maillette de Buy Wenniger et al., 2020) for the task of citation count prediction. The name "BiLSTM" is somewhat deceptive as the model contains several other layers in addition to a plain BiLSTM to improve performance:

1. The sentence embeddings in the input are fed to an average pooling layer, to combine them to a single representation per input sentence.

2. Following the BiLSTM is a max-pooling layer followed by a rectified linear hidden layer. These additional layers are added to further improve performance.

The simplicity of the sentence encoding employed by this model yields relatively high computational efficiency, lower memory usage and scalability to longer input text. This makes the model competitive in settings where the amount of training material is limited, such as PeerRead accept/reject prediction (Shen et al., 2019; Maillette de Buy Wenniger et al., 2020). However, as we will show later in this work, there is a clear advantage to using the more advanced SChuBERT model given enough training data is available.

3.2 Hierarchical Attention Networks

The HAN model (Yang et al., 2016), see Figure 1, used in this work is a PyTorch re-implementation of the original model.[1] It is in some ways similar to the BiLSTM model discussed earlier, but creates more advanced sentence-level representations by applying a BiLSTM with attention for encoding these as well as employing a BiLSTM with attention for for converting the sentence-level representations to document-level representations.

We next discuss the more advanced BERT-based model.

3.3 SChuBERT

Our SChuBERT model, shown in fig 2, consists of two parts: a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model (Devlin et al., 2018) to extract features and a deep learning model to learn from the features and predict. The main difference between this model and the other models is the use of contextualized word embeddings instead of context-independent word embeddings such as the Glove embeddings used in the HAN model. These offer a much richer context by not just encoding a word using a static embedding but encoding it based on the context it appears in.

One limitation of transformer-based models such as BERT is that they have a time complexity of $\mathcal{O}(N^2)$ with respect to the input length. For this reason, most of these models are pre-trained on sequences of a maximum length of 512. Since we are dealing with very long sequences, we have to work around this limit. The simplest approach is to truncate the documents to a length of 512 as proposed in (Xie et al., 2019). However, since our documents are so long, this would remove a lot of information. For this reason, we adopt the technique proposed in (Joshi et al., 2019). We split each input into chunks of 512 with an overlap of 50 tokens each to preserve a relation between the chunks.

For pre-trained BERT models for feature extraction, there are two considerations to make. Firstly, since BERT generates an embedding of length 768 for each token in our chunks of (max) 512 tokens, we need to pool over these embeddings to get embeddings of equal length. Note that the CLS (classification) token, which is normally used for classification tasks, is not a good representation without fine-tuning since it only holds useful information for the pre-training tasks when no fine-tuning is performed on the target domain. For this reason, we use mean pooling over our embeddings. Secondly, the different layers in BERT hold different information. The earlier layers are closer to the original word embeddings, which in the case of BERT are WordPiece embeddings (Wu et al., 2016), while the later layers are closer to the pre-training targets. Intuitively, it would make sense that the last layers are too close to the pre-training targets and are thus biased. However, our findings correspond with (Pe-

[1]Adapted from https://github.com/cedias/Hierarchical-Sentiment

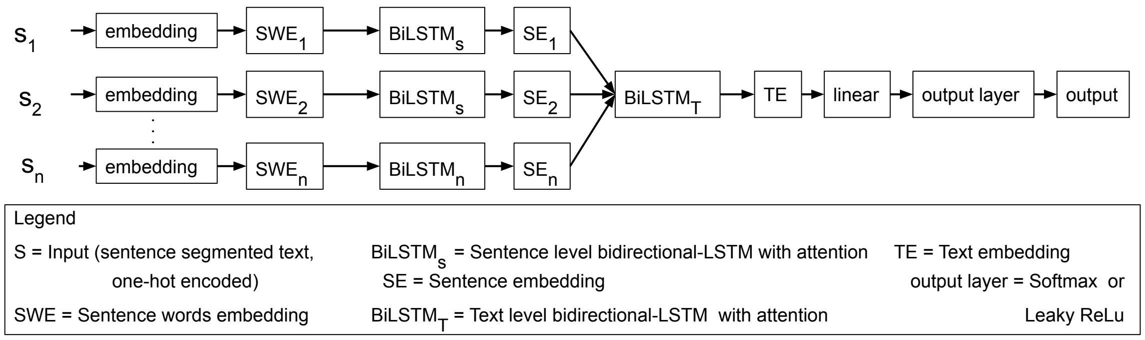
S1
S2
Sn
embedding
SWE1
SWE2
SWEn
BiLSTMs
BiLSTMn
SE1
SE2
SEn
BiLSTMT
TE
linear
output layer
output
Legend
S = Input (sentence segmented text, one-hot encoded)
SWE = Sentence words embedding
BiLSTMs = Sentence level bidirectional-LSTM with attention
SE = Sentence embedding
BiLSTMT = Text level bidirectional-LSTM with attention
TE = Text embedding
output layer = Softmax or Leaky ReLu

Figure 1: The HAN baseline model used in this work. Adapted from (Maillette de Buy Wenniger et al., 2020) with permission of the authors.

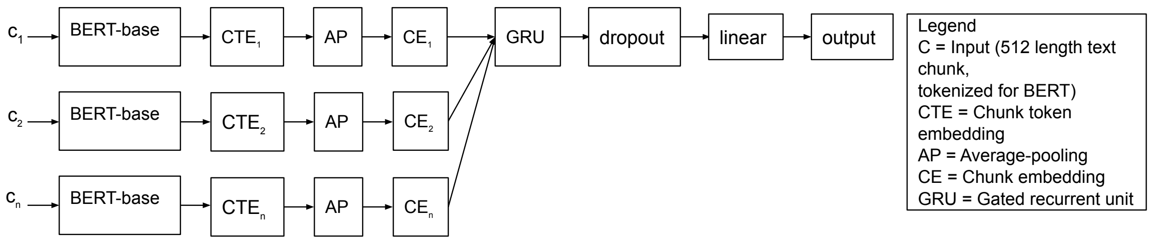
c1
c2
cn
BERT-base
CTE1
CTE2
CTEn
AP
CE1
CE2
CEn
GRU
dropout
linear
output
Legend
C = Input (512 length text chunk, tokenized for BERT)
CTE = Chunk token embedding
AP = Average-pooling
CE = Chunk embedding
GRU = Gated recurrent unit

Figure 2: The SChuBERT model proposed in this work.

ters et al., 2019) in which the last layer (layer 12) is found to be the most useful for feature extraction which is why we use this layer.

After extracting the embeddings, they are passed through a fairly simple model to do predictions. We use a GRU, followed by a single dropout layer and a linear layer. We use a simple model since the embeddings already hold a lot of information and are prone to over-fitting when a more complex model is used.

4 Dataset construction

Citation count prediction relies on sufficiently large labeled data, of good quality and preferably with full document text. To obtain such data, we need:

1. A large set of good quality scholarly documents, preferably in the same domain, or a way to collect such a set from the internet.
2. A scalable way to obtain citation counts for papers , and a way to restrict the citation counting to a fixed number of years after a paper's publication, in order to get comparable counts for papers that are published in different years.

To accomplish these, we first discuss a method to collect papers from the ACL Anthology database, yielding a relatively large set of full text documents, of notably good quality and relatively controlled length in comparison to other alternatives such as the arXiv repository which we also considered. The resulting dataset is called ACL-BiblioMetry.[2] We next discuss a method to collect the required citation counts.

4.1 Scraping ACL

For retrieving the data from the ACL Anthology database, we use the method described in Algorithm 1. First, all relevant links are extracted from the ACL Anthology main page. This includes links to all listed venues from all years. The venue as well as the year is saved for each entry as they are used as names for the saved PDF and bib files. Then, for each link, the page source is retrieved. In the page source, all relevant links to PDFs are found. PDF links that correspond to posters, presentations, supplementary materials and notes are ignored. After this, the bib link corresponding to the PDF link is extracted and both are saved using the venue name and year.

4.2 Citations Retrieval

To retrieve the set of papers that cites a given paper, we use the Semantic Scholar database.[3] The

[2]Link to scraper code and citation information data: https://github.com/Pringled/ACL-BiblioMetry

[3]The Scholar Database source files are available from https://api.semanticscholar.org/corpus/download/ .

{"entities":[],"journalVolume":"","journalPages":"97-115","pmid":"","fieldsOfStudy":["Computer Science"],
"year":2019,**"outCitations": ["c91f19447f7a72afe58ecf7281033df276b20497",
"bd59f9543127f56074aa2e6adb259099eb333912", "acbd8a36a59b7e27ddf24b64133b6b9cf4c6990c",
"d8c1b48ae4d6e4676d060c06087bb6b1ac81a005", . . . ,"2671a510c47b7fbe117fa07051829914cd1b4c98"]**,
"s2Url":"https://semanticscholar.org/
paper/3958cfb18ce6f32e90bd6ef5473be7ddd5a4e464", "s2PdfUrl":"", **"id":"3958cfb18ce6f32e90bd6ef5473be7ddd5a4e464"**,
"authors":[{"name":"Tim van de Kamp","ids":["7401984"]},{"name":"David Stritzl","ids":["146553639"]},{"name":"Willem
Jonker","ids":["6235263"]},{"name":"Andreas Peter","ids":["144253636"]}],"journalName":"",
"paperAbstract":"We propose several functional encryption schemes for set intersection and variants on two or multiple sets.
. . .
"inCitations":["f1a2ab3038bedbdfabd35f8d41103b99f51d0ec7"], "title":"Two-Client and Multi-client Functional Encryption for Set Intersection","doi":"10.1007/978-3-030-21548-4_6","sources":["DBLP"],"doiUrl":"https://doi.org/10.1007/978-3-030-21548-4_6","venue":"ACISP"}

Figure 3: Example of a JSON paper entry in the Semantic Scholar database source files. The paper id, outCitations, and inCitations are shown in bold, for clarity.

```
scrapeACL
Data: None.
Result: A folder of PDF and corresponding bib files.
relevant_links ← extractLinks(page)
for link ∈ relevant_links do
    title ← getVenueAndYear(link);
    page_source ← getPageContent(link);
    for line ∈ page_source do
        if in_line("pdf") and not (in_line("poster")
          or in_line("presentation") or
          in_line("supplementary") or
          in_line("notes")) then
            pdf_link ← extractPdfLink(line)
            bib_link ← extractBibLink(line)
            downloadPdf(pdf_link)
            downloadBib(bib_link)
return None
```

Algorithm 1: Algorithm for scraping the ACL anthology database.

```
findCitationsForArticleFromDatabase
Data: ⟨authors_table, articles_table⟩, ⟨title: String, authors: list⟩.
Result: A dictionary of ⟨year, citation_ids_list⟩ entries.
for author ∈ authors do
    article_ids ←
      selectIDsWithAuthor(authors_table, author);
    for article_id ∈ article_ids do
        article ←
          selectArticleWithID(articles_table, article_id);
        if article.title = title then
            return computeYearGroupedCitations(article)
return None
```

Algorithm 2: Algorithm for matching an article title and authors list to the database, returning the citations information for the first found article that matches the title and one of the authors.

database in its provided form consists of a collection of JSON objects, one per line. Figure 3 shows an example of an entry from the database source-files. Each entry has an *id*, a list of paper *outCitations*: IDs of the papers that the entry paper cites, as well as a list of paper *inCitations*: IDs of papers that are citing the entry paper. For our purposes in this work we are mainly interested in the *inCitations* information. Naively, the raw semantic scholar database entries already provide us with the information of how often a paper is cited. However, in practice this is not very useful, since papers are published in different years. Consequently, more recent papers will have had much less time to "collect" citations. To correct for this, and get comparable citation counts, we need to count only citations within a fixed window of time from each paper's data of publication. The latter task is slightly more involved to solve, noting that every source file is 1.6 Gigabytes, with 185 source files for a total of 283 Gigabytes of text data at the time of writing.

4.3 SQL database for efficient retrieval

By creating an SQL database that contains all the information in a structured way with proper indices, the task becomes manageable. Specifically we create a database consisting of two tables:

1. Authors table. Fields: [article_id (text, PRIMARY KEY), author_name (text).]
   An index is added to author_name, to facilitate fast lookup of papers that have a certain

```
computeYearGroupedCitations
Data: ⟨authors_table, articles_table⟩,
      article.
Result: A dictionary of ⟨year,
        citation_ids_list⟩ entries.
result_dict $\leftarrow$ dict([]);
for citing_article_id $\in$ article.in_citations
 do
    citing_article $\leftarrow$
      selectArticleWithID(articles_table,
      citing_article_id);
    if not(citing_article.year $\in$ result_dict)
     then
        result_dict[year] $\leftarrow$ list([]);
    result_dict[year].
      append(citing_article.article_id);
return result_dict
```

Algorithm 3: Algorithm for generating a dictionary of ids of citing articles, collected in sub-lists indexed by year.

author.

2. Articles table. Fields: [article_id (text. PRIMARY KEY), title (text), pages (text), year (text), volume (text), journal (text), inbound_citations (text), outbound_citations (text), doi (text)].
The fields in the articles table are kept quite minimal, omitting some unnecessary information from the original semantic scholar source files. An index is added to article_id for fast lookup of a paper with a given article_id.

4.3.1 Database creation

After creating the database the two tables are filled by simply looping over the semantic scholar source files and adding a corresponding entry to the articles table for each article entry in the source files. The authors table in addition is filled with an entry for each author of the article entry. The aim of this is that an article can be retrieved based on each of the author's names separately, increasing recall. To further increase recall all the author names are lowercased (in the created database and during retrieval).

4.3.2 Number of citations retrieval

Given an article, the citations of the article are retrieved from the database based on the authors list and title of the paper. This is done in two stages, shown also in Algorithm 2:

1. Paper retrieval: One by one, for each of the authors, all paper ids are retrieved. From these, matching article entries are found from the articles table. The first paper by any of the authors that matches the query title is returned as a positive match.[4] Just as the case for author names, titles are also lowercased to further increase recall.

2. Once the correct article entry is retrieved, the list of paper IDs of inbound citations, can be obtained from this entry. For each of these IDs an article entry is obtained and from that entry the publication year of that article. Finally, the IDs of the citing papers are grouped in a dictionary indexed by year (see Algorithm 3).

4.3.3 Citation scores and year-range uniformity

In our work we follow Maillette de Buy Wenniger et al. (2020) in using citation scores defined as

$$\text{citation\_score} = log(\text{number\_of\_citations} + 1) \tag{1}$$

When computing these (or other) scores, it is critical to use uniform year-ranges, that is a uniform MAX_YEARS: the maximum years after the publication of an article for collecting citations. A secondary question is: what are good values for MAX_YEARS? We believe in principle higher values will reduce the effects of randomness in the scores, and therefore it seems reasonable to allow at least a few years (e.g. setting MAX_YEARS > 3). Taking this into account, we believe that whereas enforcing MAX_YEARS uniformly is important, the value chosen for it is less important: all large enough values will give citation_score distributions such that the citation_score can be used (to some extent) to reflect the relative quality or impact of articles. Therefore, we leave finding an optimal value for future work. Even so, the advantage of the way we collect the citation information is that it is straightforward to experiment with different settings. One practical reason however for not choosing the parameter too large is that it disallows more recent publications to be included. For example, at the time of writing (August 2020) setting MAX_YEARS to 3 means that articles

[4]We require only one author name to match, because this significantly increases recall, while the chance of false positives given the full title and one fully matching author name is negligible.

Table 1: Properties of datasets for citation count prediction applied in earlier work.

paper source	# papers (train + validation + test)	# reviews	paper text type
(Fu and Aliferis, 2008)	3788	N/A	title+abstract
(Li et al., 2019)	1739, 384	7171, 1119	title + abstract
(Plank and van Dale, 2019)	3427	12260	title + abstract
(Maillette de Buy Wenniger et al., 2020)	78894 + 4383 + 4382	N/A	title + abstract + partial body
ACL-BiblioMetry dataset (this work)	27853 + 1548 + 1549	N/A	title + abstract + full body

Table 2: Character count per example statistics ACL dataset different settings.

systems	BiLSTM, HAN, SChuBERT	BiLSTM, HAN	SChuBERT		
setting	title + abstract	title + abstract + body text (max 200000 chars)	title + abstract + body text		
			max 5 chunks	max 6 chunks	no limit
#characters (avg, max)	975 , 20000	17293 , 20000	12019 , 19064	14061 , 22643	23787 , 1261656

Table 3: Hyperparameters used in the experiments.

	BiLSTM and HAN	SChuBERT
vocabulary size	10000	
weight initialization		
general	Xavier uniform	
lstm	Xavier normal	
bias	zero	
optimizer, learning rate	Adam, 0.005	Adam, 0.001
epochs	160	30
maximum input characters	20000	no limit
word embeddings	GloVe	N/A
loss function	MAE	MAE
dropout probability	0.5	0.3
BiLSTM/GRU hidden size	192	512
batch size	4, 16	12
word embedding size	50	N/A
BERT sentence embedding size	N/A	768

published up to 2016 can be included, as they have 3 complete years after 2016 (i.e. 2017, 2018, 2019) to "collect" citations. Papers published after 2016 cannot be included with this setting. We used this setting in our experiments, as we believe it to be large enough to give reliable citation_score values, while small enough to allow inclusion of a large number of articles in the data.

Computation

Once MAX_YEARS is chosen, for an article a, and and associated citations dictionary $a\_citations\_dict$ computed by Algorithm 3 selecting included citations is easy. Simply concatenate the lists of year-indexed citations sublists with $year(sublist) \leq a.year + MAX\_YEARS$. Based on the final list of included citations, the citation score or other metrics can then be easily computed.

5 Experiments

In our experiments, we want to assess the usability of the ACL data for number of citations prediction, and generally larger training data for citation prediction, as enabled by the automatic number of citation labeling framework contributed in this work. We also want to test two hypotheses:

1. Longer input text improves performance: using then entire paper text (title + abstract + body text) is substantially better than using only the paper title + abstract.

2. Larger training data substantially improves performance. More specifically, when using training data for number of citation prediction that is n times larger than what has been used for the related task of accept/reject prediction on the PeerRead CL dataset (computation and language domain) yields substantially better results than when using a training set of size comparable to PeerRead CL.

To test these two hypothesis, we perform the following comparisons:

1. Full text input in comparison to abstract only.

2. Full data input in comparison to 50% data input and to 10% data input.

To test our second hypothesis, we take 10% of our dataset to get a dataset which is approximately

Table 4: Results on the full data and with full input.

	BiLSTM	HAN	SChuBERT (5 chunk)	SChuBERT (6 chunk)	SChuBERT
R^2 score	0.319 ± 0.013	0.339 ± 0.013	0.369 ± 0.009	0.380 ± 0.004	**0.398 ± 0.006**
MSE	1.110 ± 0.021	1.080 ± 0.021	1.032 ± 0.015	1.013 ± 0.006	**0.985 ± 0.010**
MAE	0.824 ± 0.009	0.820 ± 0.009	0.805 ± 0.005	0.798 ± 0.005	**0.789 ± 0.005**

Table 5: Results on the full data and with abstract text only.

	BiLSTM	HAN	SChuBERT
R^2 score	0.158 ± 0.006	0.248 ± 0.014	**0.249 ± 0.002**
MSE	1.377 ± 0.010	**1.230 ± 0.023**	**1.230 ± 0.004**
MAE	0.933 ± 0.002	0.885 ± 0.008	**0.884 ± 0.002**

Table 6: Results for SChuBERT on a subset of the data and with full input.

	SChuBERT 50% data	SChuBERT 10% data
R^2 score	0.327 ± 0.007	0.205 ± 0.026
MSE	1.058 ± 0.011	1.473 ± 0.048
MAE	0.809 ± 0.005	0.923 ± 0.027

Table 7: Number of trainable parameters for used hidden sizes.

Hidden size	BiLSTM	HAN	SchuBERT
192	1170949	2059525	N/A
256	N/A	N/A	788225
512	N/A	N/A	969665

Table 8: Training time per epoch in seconds.

	BiLSTM	HAN	SchuBERT
Time in seconds	1048	1921	12

Table 9: Results for SChuBERT with hidden size 256 (with full data).

MSE	MAE	R2
0.994 ± 0.013	0.788 ± 0.006	0.392 ± 0.008

the size of PeerRead CL (3000 papers). We then compare this to half our data and full data to show the importance of larger datasets. Lastly, we also test SChuBERT on a portion of the chunks to ensure a fair comparison with BiLSTM and HAN which were capped at 20k characters. Statistics about the number of characters per example in the different settings are shown in Table 2.

5.1 Experimental Settings

Table 3 shows the hyperparameters used for training the models in our experiments. As evaluation metrics, we report the standard metrics of mean squared error (MS) and mean average error (MAE), which are commonly used for regression evaluation, as well as the R^2 score. We repeat each experiment three times to counter false conclusions due to optimizer instability, and report average and standard deviation for each of the metrics.

6 Results

Our results show that SChuBERT is able to outperform both BiLSTM as well as HAN for the citation prediction problem by a significant margin. Table 4 shows a comparison of the three models for full data input and full-text input. While BiLSTM and HAN have a comparable R^2 score, SChuBERT has an R^2 score of almost 0.06 higher, showing the power of contextualized word embeddings. SChuBERT also performs better with 5 chunks (which equates to less total input used than BiLSTM and HAN which were capped at 20k characters) and 6 chunks (which equates to slightly more input used than BiLSTM and HAN). For reference, the average number of chunks was 7.6. In practice, capping the chunks mostly results in extremely long papers being cut off, just like in BiLSTM and HAN.

We also compared how well the different models performed on abstract-only inputs, shown in Table 5. These results show that HAN and SChuBERT have comparable results, which shows that SChuBERT benefits more from longer inputs. In general, the performance of all models is substantially better on full-text inputs when compared to abstract only.

In Table 6 we show the performance of SChuBERT on less data. As expected, the performance decreases substantially with less data, showing the benefit of larger datasets such as the one proposed in this paper. Due to time constraints, we did not test BiLSTM and HAN on less data.

As a final comparison of the systems, we show the number of trainable parameters in Table 7 and the training time in seconds per epoch in Table 8. As can be seen, SChuBERT has a smaller number of trainable parameters even with a larger hidden size for the GRU. Additionally, in Table 9 we show results for SChuBERT when we half the hidden size to 256, which turns out to only give a small drop in performance. The training time in seconds per epoch is also much lower for SChuBERT, which trains approximately 87 faster than BiLSTM and 160 times faster than HAN. However, this is after the embeddings have been generated, which takes relatively long (a little over 7 hours for the full dataset) but only has to be done once. Even when taking this into consideration, training SChuBERT is still much faster given that it converges about 4 times faster than the other systems.

7 Conclusion

In this work, we showed the importance of larger and better curated data for the citation prediction problem. We proposed ACL-BiblioMetry, a new large dataset created with the algorithms we provide in this work. We also proposed SChuBERT, a new model for the citation prediction problem which can deal with large inputs and gets significantly better results than several state-of-the-art models. The model shows the strength of modern language models and contextualized word embeddings and their appliance to the citation prediction problem. Our results showed that both the length of the input as well as the amount of data are important for achieving better results. The current work takes a step forward by using a larger training set of full text examples and leveraging this data with stronger models, in particular the SChuBERT model, without considering the historical publication context and other factors. We leave experimentation with further extended context for the predictive models, as well as other language models and even larger datasets for future work.

Acknowledgments

The Peregrine high performance computing cluster, at the Center for Information Technology of the University (CIT) of Groningen, was used for running part of the experiments in this study. We would like to thank the people at the CIT for their support and access to the cluster. We would also like to thank Charles-Emmanuel Dias for sharing his HAN implementation, which proved to be a solid foundation for the HAN models used in this work.

References

Ali Abrishami and Sadegh Aliakbary. 2019. Predicting citation counts based on deep neural network learning techniques. *Journal of Informetrics*, 13:485–499.

Xiaomei Bai, Fuli Zhang, and Ivan Lee. 2019. Predicting the citations of scholarly paper. *Journal of Informetrics*, 13(1):407 – 418.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Tim Brody, Stevan Harnad, and Les Carr. 2006. Earlier web usage statistics as predictors of later citation impact. *Journal of the American Association for Information Science and Technology (JASIST)*, 57(8):1060–1072. DOI: 10.1002/asi.20373.

Gideon Maillette de Buy Wenniger, Thomas van Dongen, Eleri Aedmaa, Herbert Teun Kruitbosch, Edwin A. Valentijn, and Lambert Schomaker. 2020. Structure-tags improve text classification for scholarly document quality prediction. In *Proceedings of the the 1st Workshop on Scholarly Document Processing (SDP 2020)*. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

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

Lawrence Fu and Constantin Aliferis. 2008. Models for predicting and explaining citation count of biomedical articles. *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, 6:222–6.

Alfonso Ibáñez, Pedro Larrañaga, and Concha Bielza. 2009. Predicting citation count of Bioinformatics papers within four years of publication. *Bioinformatics*, 25(24):3303–3309.

Mandar Joshi, Omer Levy, Daniel S. Weld, and Luke Zettlemoyer. 2019. Bert for coreference resolution: Baselines and analysis.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. 2018. A dataset of peer reviews (peerread): Collection, insights and nlp applications.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer.

Siqing Li, Wayne Xin Zhao, Eddy Jing Yin, and Ji-Rong Wen. 2019. A neural citation count prediction model based on peer review 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 4914–4924, Hong Kong, China. Association for Computational Linguistics.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Dan S. Weld. 2019. S2orc: The semantic scholar open research corpus.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pre-trained representations to diverse tasks.

Barbara Plank and Reinard van Dale. 2019. Cite-tracked: A longitudinal dataset ofpeer reviews and citations. In *Proceedings of the 4th Joint Workshop on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL 2019)*.

Aili Shen, Jianzhong Qi, and Timothy Baldwin. 2017. A hybrid model for quality assessment of wikipedia articles. In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pages 43–52.

Aili Shen, Bahar Salehi, Timothy Baldwin, and Jianzhong Qi. 2019. A joint model for multimodal document quality assessment.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation for consistency training.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.