Document-level Representation Learning using Citation-informed Transformers

Arman Cohan[†*] **Sergey Feldman**[†*] **Iz Beltagy**[†] **Doug Downey**[†] **Daniel S. Weld**[†,‡]

[†]Allen Institute for Artificial Intelligence
[‡]Paul G. Allen School of Computer Science & Engineering, University of Washington
{armanc,sergey,beltagy,dougd,danw}@allenai.org

Abstract

Representation learning is a critical ingredient for natural language processing systems. Recent Transformer language models like BERT learn powerful textual representations, but these models are targeted towards token- and sentence-level training objectives and do not leverage information on inter-document relatedness, which limits their document-level representation power. For applications on scientific documents, such as classification and recommendation, the embeddings power strong performance on end tasks. We propose SPECTER, a new method to generate document-level embedding of scientific documents based on pretraining a Transformer language model on a powerful signal of document-level relatedness: the citation graph. Unlike existing pretrained language models, SPECTER can be easily applied to downstream applications without task-specific fine-tuning. Additionally, to encourage further research on document-level models, we introduce SCIDOCS, a new evaluation benchmark consisting of seven document-level tasks ranging from citation prediction, to document classification and recommendation. We show that SPECTER outperforms a variety of competitive baselines on the benchmark.[1]

1 Introduction

As the pace of scientific publication continues to increase, Natural Language Processing (NLP) tools that help users to search, discover and understand the scientific literature have become critical. In recent years, substantial improvements in NLP tools have been brought about by pretrained neural language models (LMs) (Radford et al., 2018; Devlin et al., 2019; Yang et al., 2019). While such models are widely used for representing individual words or sentences, extensions to whole-document embeddings are relatively underexplored. Likewise, methods that do use inter-document signals to produce whole-document embeddings (Tu et al., 2017; Chen et al., 2019) have yet to incorporate state-of-the-art pretrained LMs. Here, we study how to leverage the power of pretrained language models to learn embeddings for scientific documents.

A paper's title and abstract provide rich semantic content about the paper, but, as we show in this work, simply passing these textual fields to an "off-the-shelf" pretrained language model—even a state-of-the-art model tailored to scientific text like the recent SciBERT (Beltagy et al., 2019)—does *not* result in accurate paper representations. The language modeling objectives used to pretrain the model do not lead it to output representations that are helpful for document-level tasks such as topic classification or recommendation.

In this paper, we introduce a new method for learning general-purpose vector representations of scientific documents. Our system, SPECTER,[2] incorporates inter-document context into the Transformer (Vaswani et al., 2017) language models (e.g., SciBERT (Beltagy et al., 2019)) to learn document representations that are effective across a wide-variety of downstream tasks, without the need for any task-specific fine-tuning of the pretrained language model. We specifically use citations as a naturally occurring, inter-document incidental supervision signal indicating which documents are most related and formulate the signal into a triplet-loss pretraining objective. Unlike many prior works, at inference time, our model does not require any citation information. This is critical for embedding new papers that have not yet been cited. In experiments, we show that SPECTER's representations substantially outperform the state-

[*] Equal contribution
[1] https://github.com/allenai/specter
[2] SPECTER: Scientific Paper Embeddings using Citation-informed TransformERs

of-the-art on a variety of document-level tasks, including topic classification, citation prediction, and recommendation.

As an additional contribution of this work, we introduce and release SciDocs, a novel collection of data sets and an evaluation suite for document-level embeddings in the scientific domain. SciDocs covers seven tasks, and includes tens of thousands of examples of anonymized user signals of document relatedness. We also release our training set (hundreds of thousands of paper titles, abstracts and citations), along with our trained embedding model and its associated code base.

2 Model

2.1 Overview

Our goal is to learn task-independent representations of academic papers. Inspired by the recent success of pretrained Transformer language models across various NLP tasks, we use the Transformer model architecture as basis of encoding the input paper. Existing LMs such as BERT, however, are primarily based on masked language modeling objective, only considering intra-document context and do not use any inter-document information. This limits their ability to learn optimal document representations. To learn high-quality document-level representations we propose using citations as an inter-document relatedness signal and formulate it as a triplet loss learning objective. We then pretrain the model on a large corpus of citations using this objective, encouraging it to output representations that are more similar for papers that share a citation link than for those that do not. We call our model SPECTER, which learns Scientific Paper Embeddings using Citation-informed TransformERs. With respect to the terminology used by Devlin et al. (2019), unlike most existing LMs that are "fine-tuning based", our approach results in embeddings that can be applied to downstream tasks in a "feature-based" fashion, meaning the learned paper embeddings can be easily used as features, with no need for further task-specific fine-tuning. In the following, as background information, we briefly describe how pretrained LMs can be applied for document representation and then discuss the details of SPECTER.

2.2 Background: Pretrained Transformers

Recently, pretrained Transformer networks have demonstrated success on various NLP tasks (Radford et al., 2018; Devlin et al., 2019; Yang et al., 2019; Liu et al., 2019); we use these models as the foundation for SPECTER. Specifically, we use SciBERT (Beltagy et al., 2019) which is an adaptation of the original BERT (Devlin et al., 2019) architecture to the scientific domain. The BERT model architecture (Devlin et al., 2019) uses multiple layers of Transformers (Vaswani et al., 2017) to encode the tokens in a given input sequence. Each layer consists of a self-attention sublayer followed by a feedforward sublayer. The final hidden state associated with the special `[CLS]` token is usually called the "pooled output", and is commonly used as an aggregate representation of the sequence.

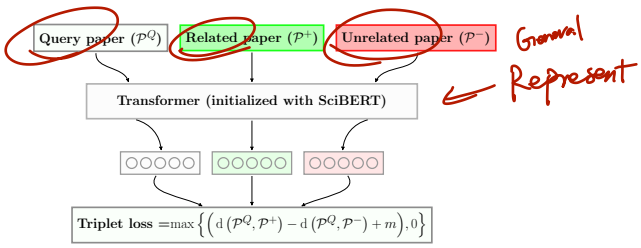


Figure 1: Overview of SPECTER.

Document Representation Our goal is to represent a given paper $\mathcal{P}$ as a dense vector $\mathbf{v}$ that best represents the paper and can be used in downstream tasks. SPECTER builds embeddings from the title and abstract of a paper. Intuitively, we would expect these fields to be sufficient to produce accurate embeddings, since they are written to provide a succinct and comprehensive summary of the paper.[3] As such, we encode the concatenated title and abstract using a Transformer LM (e.g., SciBERT) and take the final representation of the `[CLS]` token as the output representation of the paper:[4]

$$\mathbf{v} = \texttt{Transformer(input)}_{\texttt{[CLS]}}, \quad (1)$$

where `Transformer` is the Transformer's forward function, and `input` is the concatenation of the `[CLS]` token and WordPieces (Wu et al., 2016) of the title and abstract of a paper, separated by the `[SEP]` token. We use SciBERT as our model initialization as it is optimized for scientific text, though our formulation is general and any Transformer language model instead of SciBERT. Using

[3] We also experimented with additional fields such as venues and authors but did not find any empirical advantage in using those (see §6). See §7 for a discussion of using the full text of the paper as input.

[4] It is also possible to encode title and abstracts individually and then concatenate or combine them to get the final embedding. However, in our experiments this resulted in sub-optimal performance.

the above method with an "off-the-shelf" SciBERT does not take global inter-document information into account. This is because SciBERT, like other pretrained language models, is trained via language modeling objectives, which only predict words or sentences given their in-document, nearby textual context. In contrast, we propose to incorporate citations into the model as a signal of inter-document relatedness, while still leveraging the model's existing strength in modeling language.

2.3 Citation-Based Pretraining Objective

A citation from one document to another suggests that the documents are related. To encode this relatedness signal into our representations, we design a loss function that trains the Transformer model to learn closer representations for papers when one cites the other, and more distant representations otherwise. The high-level overview of the model is shown in Figure 1.

In particular, each training instance is a triplet of papers: a query paper $\mathcal{P}^Q$, a positive paper $\mathcal{P}^+$ and a negative paper $\mathcal{P}^-$. The positive paper is a paper that the query paper cites, and the negative paper is a paper that is not cited by the query paper (but that may be cited by $\mathcal{P}^+$). We then train the model using the following triplet margin loss function:

$$\mathcal{L} = \max\left\{\left(\mathrm{d}\left(\mathcal{P}^Q, \mathcal{P}^+\right) - \mathrm{d}\left(\mathcal{P}^Q, \mathcal{P}^-\right) + m\right), 0\right\} \quad (2)$$

where d is a distance function and m is the loss margin hyperparameter (we empirically choose $m = 1$). Here, we use the L2 norm distance:

$$\mathrm{d}(\mathcal{P}^A, \mathcal{P}^B) = \|\mathbf{v}_A - \mathbf{v}_B\|_2,$$

where $\mathbf{v}_A$ is the vector corresponding to the pooled output of the Transformer run on paper A (Equation 1).[5] Starting from the trained SciBERT model, we pretrain the Transformer parameters on the citation objective to learn paper representations that capture document relatedness.

2.4 Selecting Negative Distractors

The choice of negative example papers $\mathcal{P}-$ is important when training the model. We consider two sets of negative examples: the first set simply consists of randomly selected papers from the corpus. Given a query paper, intuitively we would expect the model to be able to distinguish between cited papers, and uncited papers sampled randomly from the entire corpus. This inductive bias has been also found to be effective in content-based citation recommendation applications (Bhagavatula et al., 2018). But, random negatives may be easy for the model to distinguish from the positives. To provide a more nuanced training signal, we augment the randomly drawn negatives with a more challenging second set of negative examples. We denote as "hard negatives" the papers that are not cited by the query paper, but *are* cited by a paper cited by the query paper, i.e. if $\mathcal{P}_1 \xrightarrow{cite} \mathcal{P}_2$ and $\mathcal{P}_2 \xrightarrow{cite} \mathcal{P}_3$ but $\mathcal{P}_1 \not\xrightarrow{cite} \mathcal{P}_3$, then $\mathcal{P}_3$ is a candidate hard negative example for $\mathcal{P}_1$. We expect the hard negatives to be somewhat related to the query paper, but typically less related than the cited papers. As we show in our experiments (§6), including hard negatives results in more accurate embeddings compared to using random negatives alone.

2.5 Inference

At inference time, the model receives one paper, $\mathcal{P}$, and it outputs the SPECTER's Transfomer pooled output activation as the paper representation for $\mathcal{P}$ (Equation 1). We note that for inference, SPECTER requires only the title and abstract of the given input paper; the model does not need any citation information about the input paper. This means that SPECTER can produce embeddings even for new papers that have yet to be cited, which is critical for applications that target recent scientific papers.

3 SciDocs Evaluation Framework

Previous evaluations of scientific document representations in the literature tend to focus on small datasets over a limited set of tasks, and extremely high (99%+) AUC scores are already possible on these data for English documents (Chen et al., 2019; Wang et al., 2019). New, larger and more diverse benchmark datasets are necessary. Here, we introduce a new comprehensive evaluation framework to measure the effectiveness of scientific paper embeddings, which we call SciDocs. The framework consists of diverse tasks, ranging from citation prediction, to prediction of user activity, to document classification and paper recommendation. Note that SPECTER will not be further fine-tuned on any of the tasks; we simply plug in the embeddings as features for each task. Below, we describe each of the tasks in detail and the evaluation data associated with it. In addition to our training data, we release all the datasets associated with the evaluation tasks.

[5] We also experimented with other distance functions (e..g, normalized cosine), but they underperformed the L2 loss.

3.1 Document Classification

An important test of a document-level embedding is whether it is predictive of the class of the document. Here, we consider two classification tasks in the scientific domain:

MeSH Classification In this task, the goals is to classify scientific papers according to their Medical Subject Headings (MeSH) (Lipscomb, 2000).[6] We construct a dataset consisting of 23K academic medical papers, where each paper is assigned one of 11 top-level disease classes such as cardiovascular diseases, diabetes, digestive diseases derived from the MeSH vocabulary. The most populated category is Neoplasms (cancer) with 5.4K instances (23.3% of the total dataset) while the category with least number of samples is Hepatitis (1.7% of the total dataset). We follow the approach of Feldman et al. (2019) in mapping the MeSH vocabulary to the disease classes.

Paper Topic Classification This task is predicting the topic associated with a paper using the predefined topic categories of the Microsoft Academic Graph (MAG) (Sinha et al., 2015)[7]. MAG provides a database of papers, each tagged with a list of topics. The topics are organized in a hierarchy of 5 levels, where level 1 is the most general and level 5 is the most specific. For our evaluation, we derive a document classification dataset from the level 1 topics, where a paper is labeled by its corresponding level 1 MAG topic. We construct a dataset of 25K papers, almost evenly split over the 19 different classes of level 1 categories in MAG.

3.2 Citation Prediction

As argued above, citations are a key signal of relatedness between papers. We test how well different paper representations can reproduce this signal through citation prediction tasks. In particular, we focus on two sub-tasks: predicting *direct citations*, and predicting *co-citations*. We frame these as ranking tasks and evaluate performance using MAP and nDCG, standard ranking metrics.

Direct Citations In this task, the model is asked to predict which papers are cited by a given query paper from a given set of candidate papers. The evaluation dataset includes approximately 30K total papers from a held-out pool of papers, consisting of 1K query papers and a candidate set of up to 5 cited papers and 25 (randomly selected) uncited papers. The task is to rank the cited papers higher than the uncited papers. For each embedding method, we require only comparing the L2 distance between the raw embeddings of the query and the candidates, without any additional trainable parameters.

Co-Citations This task is similar to the *direct citations* but instead of predicting a cited paper, the goal is to predict a highly co-cited paper with a given paper. Intuitively, if papers A and B are cited frequently together by several papers, this shows that the papers are likely highly related and a good paper representation model should be able to identify these papers from a given candidate set. The dataset consists of 30K total papers and is constructed similar to the *direct citations* task.

3.3 User Activity

The embeddings for similar papers should be close to each other; we use user activity as a proxy for identifying similar papers and test the model's ability to recover this information. Multiple users consuming the same items as one another is a classic relatedness signal and forms the foundation for recommender systems and other applications (Schafer et al., 2007). In our case, we would expect that when users look for academic papers, the papers they view in a single browsing session tend to be related. Thus, accurate paper embeddings should, all else being equal, be relatively more similar for papers that are frequently viewed in the same session than for other papers. To build benchmark datasets to test embeddings on user activity, we obtained logs of user sessions from a major academic search engine. We define the following two tasks on which we build benchmark datasets to test embeddings:

Co-Views Our co-views dataset consists of approximately 30K papers. To construct it, we take 1K random papers that are not in our train or development set and associate with each one up to 5 frequently co-viewed papers and 25 randomly selected papers (similar to the approach for citations). Then, we require the embedding model to rank the co-viewed papers higher than the random papers by comparing the L2 distances of raw embeddings. We evaluate performance using standard ranking metrics, nDCG and MAP.

[6] https://www.nlm.nih.gov/mesh/meshhome.html

[7] https://academic.microsoft.com/

Co-Reads If the user clicks to access the PDF of a paper from the paper description page, this is a potentially stronger sign of interest in the paper. In such a case we assume the user will read at least parts of the paper and refer to this as a "read" action. Accordingly, we define a "co-reads" task and dataset analogous to the co-views dataset described above. This dataset is also approximately 30K papers.

3.4 Recommendation

In the recommendation task, we evaluate the ability of paper embeddings to boost performance in a production recommendation system. Our recommendation task aims to help users navigate the scientific literature by ranking a set of "similar papers" for a given paper. We use a dataset of user clickthrough data for this task which consists of 22K clickthrough events from a public scholarly search engine. We partitioned the examples temporally into train (20K examples), validation (1K), and test (1K) sets. As is typical in clickthrough data on ranked lists, the clicks are biased toward the top of original ranking presented to the user. To counteract this effect, we computed propensity scores using a swap experiment (Agarwal et al., 2019). The propensity scores give, for each position in the ranked list, the relative frequency that the position is over-represented in the data due to exposure bias. We can then compute de-biased evaluation metrics by dividing the score for each test example by the propensity score for the clicked position. We report propensity-adjusted versions of the standard ranking metrics Precision@1 ($\hat{\text{P}}$@1) and Normalized Discounted Cumulative Gain ($\hat{\text{nDCG}}$).

We test different embeddings on the recommendation task by including cosine embedding distance[8] as a feature within an existing recommendation system that includes several other informative features (title/author similarity, reference and citation overlap, etc.). Thus, the recommendation experiments measure whether the embeddings can boost the performance of a strong baseline system on an end task. For SPECTER, we also perform an online A/B test to measure whether its advantages on the offline dataset translate into improvements on the online recommendation task (§5).

[8]Embeddings are L2 normalized and in this case cosine distance is equivalent to L2 distance.

4 Experiments

Training Data To train our model, we use a subset of the Semantic Scholar corpus (Ammar et al., 2018) consisting of about 146K query papers (around 26.7M tokens) with their corresponding outgoing citations, and we use an additional 32K papers for validation. For each query paper we construct up to 5 training triples comprised of a query, a positive, and a negative paper. The positive papers are sampled from the direct citations of the query, while negative papers are chosen either randomly or from citations of citations (as discussed in §2.4). We empirically found it helpful to use 2 hard negatives (citations of citations) and 3 easy negatives (randomly selected papers) for each query paper. This process results in about 684K training triples and 145K validation triples.

Training and Implementation We implement our model in AllenNLP (Gardner et al., 2018). We initialize the model from SciBERT pretrained weights (Beltagy et al., 2019) since it is the state-of-the-art pretrained language model on scientific text. We continue training all model parameters on our training objective (Equation 2). We perform minimal tuning of our model's hyperparameters based on the performance on the validation set, while baselines are extensively tuned. Based on initial experiments, we use a margin m=1 for the triplet loss. For training, we use the Adam optimizer (Kingma and Ba, 2014) following the suggested hyperparameters in Devlin et al. (2019) (LR: 2e-5, Slanted Triangular LR scheduler[9] (Howard and Ruder, 2018) with number of train steps equal to training instances and cut fraction of 0.1). We train the model on a single Titan V GPU (12G memory) for 2 epochs, with batch size of 4 (the maximum that fit in our GPU memory) and use gradient accumulation for an effective batch size of 32. Each training epoch takes approximately 1-2 days to complete on the full dataset. We release our code and data to facilitate reproducibility. [10]

Task-Specific Model Details For the classification tasks, we used a linear SVM where embedding vectors were the only features. The C hyperparameter was tuned via a held-out validation set. For the recommendation tasks, we use a feed-forward ranking neural network that takes as input

[9]Learning rate linear warmup followed by linear decay.

[10]https://github.com/allenai/specter

ten features designed to capture the similarity between each query and candidate paper, including the cosine similarity between the query and candidate embeddings and manually-designed features computed from the papers' citations, titles, authors, and publication dates.

Baseline Methods Our work falls into the intersection of textual representation, citation mining, and graph learning, and we evaluate against state-of-the-art baselines from each of these areas. We compare with several strong textual models: SIF (Arora et al., 2017), a method for learning document representations by removing the first principal component of aggregated word-level embeddings which we pretrain on scientific text; SciBERT (Beltagy et al., 2019) a state-of-the-art pretrained Transformer LM for scientific text; and Sent-BERT (Reimers and Gurevych, 2019), a model that uses negative sampling to tune BERT for producing optimal sentence embeddings. We also compare with Citeomatic (Bhagavatula et al., 2018), a closely related paper representation model for citation prediction which trains content-based representations with citation graph information via dynamically sampled triplets, and SGC (Wu et al., 2019a), a state-of-the-art graph-convolutional approach. For completeness, additional baselines are also included; due to space constraints we refer to Appendix A for detailed discussion of all baselines. We tune hyperparameters of baselines to maximize performance on a separate validation set.

5 Results

Table 1 presents the main results corresponding to our evaluation tasks (described in §3). Overall, we observe substantial improvements across all tasks with average performance of 80.2 across all metrics on all tasks which is a 3.3 point absolute improvement over the next-best baseline. We now discuss the results in detail.

For document classification, we report macro F1, a standard classification metric. We observe that the classifier performance when trained on our representations is better than when trained on any other baseline. Particularly, on the MeSH (MAG) dataset, we obtain an 86.3 (82.6) F1 score which is about a $\Delta = +2.2$ ($+2.1$) point absolute increase over the best baseline on each dataset respectively. Our evaluation of the learned representations on predicting user activity is shown in the "User activity" columns of Table 1. SPECTER achieves a MAP score of 83.8 on the co-view task, and 84.5 on co-read, improving over the best baseline (Citeomatic in this case) by 2.7 and 4.0 points, respectively. We observe similar trends for the "citation" and "co-citation" tasks, with our model outperforming virtually all other baselines except for SGC, which has access to the citation graph at training and test time.[11] Note that methods like SGC cannot be used in real-world setting to embed new papers that are not cited yet. On the other hand, on co-citation data our method is able to achieve the best results with nDCG of 94.7, improving over SGC with 2.2 points. Citeomatic also performs well on the citation tasks, as expected given that its primary design goal was citation prediction. Nevertheless, our method slightly outperforms Citeomatic on the direct citation task, while substantially outperforming it on co-citations (+1.9 nDCG).

Finally, for recommendation task, we observe that SPECTER outperforms all other models on this task as well, with nDCG of 54.0. On the recommendations task, as opposed to previous experiments, the differences in method scores are generally smaller. This is because for this task the embeddings are used along with several other informative features in the ranking model (described under task-specific models in §4), meaning that embedding variants have less opportunity for impact on overall performance.

We also performed an online study to evaluate whether SPECTER embeddings offer similar advantages in a live application. We performed an online A/B test comparing our SPECTER-based recommender to an existing production recommender system for similar papers that ranks papers by a textual similarity measure. In a dataset of 4,113 clicks, we found that SPECTER ranker improved clickthrough rate over the baseline by 46.5%, demonstrating its superiority.

We emphasize that our citation-based pretraining objective is critical for the performance of SPECTER; removing this and using a vanilla SciBERT results in decreased performance on all tasks.

6 Analysis

In this section, we analyze several design decisions in SPECTER, provide a visualization of its embedding space, and experimentally compare

[11] For SGC, we remove development and test set citations and co-citations during training We also remove incoming citations from development and test set queries as these would not be available at test time in production.

Task →	Classification		User activity prediction				Citation prediction				Recomm.		Avg.
Subtask →	MAG	MeSH	Co-View		Co-Read		Cite		Co-Cite				
Model ↓ / Metric →	F1	F1	MAP	nDCG	MAP	nDCG	MAP	nDCG	MAP	nDCG	nD$\hat{C}$G	P@1	
Random	4.8	9.4	25.2	51.6	25.6	51.9	25.1	51.5	24.9	51.4	51.3	16.8	32.5
Doc2vec (2014)	66.2	69.2	67.8	82.9	64.9	81.6	65.3	82.2	67.1	83.4	51.7	16.9	66.6
Fasttext-sum (2017)	78.1	84.1	76.5	87.9	75.3	87.4	74.6	88.1	77.8	89.6	52.5	18.0	74.1
SIF (2017)	78.4	81.4	79.4	89.4	78.2	88.9	79.4	90.5	80.8	90.9	53.4	19.5	75.9
ELMo (2018)	77.0	75.7	70.3	84.3	67.4	82.6	65.8	82.6	68.5	83.8	52.5	18.2	69.0
Citeomatic (2018)	67.1	75.7	81.1	90.2	80.5	90.2	86.3	94.1	84.4	92.8	52.5	17.3	76.0
SGC (2019a)	76.8	82.7	77.2	88.0	75.7	87.5	**91.6**	**96.2**	84.1	92.5	52.7	18.2	76.9
SciBERT (2019)	79.7	80.7	50.7	73.1	47.7	71.1	48.3	71.7	49.7	72.6	52.1	17.9	59.6
Sent-BERT (2019)	80.5	69.1	68.2	83.3	64.8	81.3	63.5	81.6	66.4	82.8	51.6	17.1	67.5
SPECTER (Ours)	**82.6**	**86.3**	**83.8**	**91.8**	**84.5**	**92.7**	88.4	94.9	**87.9**	**94.7**	**54.0**	**20.4**	**80.2**

Table 1: Results on the SciDocs evaluation suite consisting of 7 tasks.

SPECTER's use of fixed embeddings against a fine-tuning approach.

Ablation Study We start by analyzing how adding or removing metadata fields from the input to SPECTER alters performance. The results are shown in the top four rows of Table 2 (for brevity, here we only report the average of the metrics from each task). We observe that removing the abstract from the textual input and relying only on the title results in a substantial decrease in performance. More surprisingly, adding authors as an input (along with title and abstract) hurts performance.[12] One possible explanation is that author names are sparse in the corpus, making it difficult for the model to infer document-level relatedness from them. As another possible reason of this behavior, tokenization using Wordpieces might be suboptimal for author names. Many author names are out-of-vocabulary for SciBERT and thus, they might be split into sub-words and shared across names that are not semantically related, leading to noisy correlation. Finally, we find that adding venues slightly decreases performance,[13] except on document classification (which makes sense, as we would expect venues to have high correlation with paper topics). The fact that SPECTER does not require inputs like authors or venues makes it applicable in situations where this metadata is not available, such as matching reviewers with anonymized submissions, or performing recommendations of anonymized preprints (e.g., on OpenReview).

	CLS	USR	CITE	REC	Avg.
SPECTER	84.4	**88.2**	**91.5**	**37.2**	**80.2**
− abstract	82.2	72.2	73.6	34.5	68.1
+ venue	**84.5**	88.0	91.2	36.7	79.9
+ author	82.7	72.3	71.0	34.6	67.3
No hard negatives	82.4	85.8	89.8	36.8	78.4
Start w/ BERT-Large	81.7	85.9	87.8	36.1	77.5

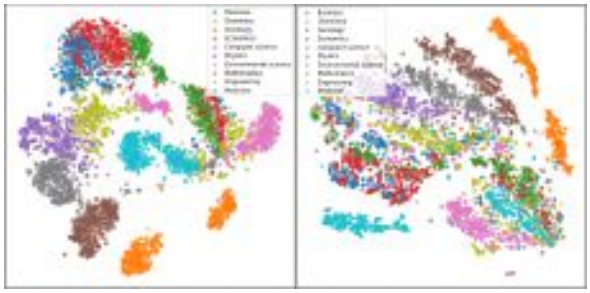
Table 2: Ablations: Numbers are averages of metrics for each evaluation task: CLS: classification, USR: User activity, CITE: Citation prediction, REC: Recommendation, Avg. average over all tasks & metrics.

One design decision in SPECTER is to use a set of hard negative distractors in the citation-based fine-tuning objective. The fifth row of Table 2 shows that this is important—using only easy negatives reduces performance on all tasks. While there could be other potential ways to include hard negatives in the model, our simple approach of including citations of citations is effective. The sixth row of the table shows that using a strong *general-domain* language model (BERT-Large) instead of SciBERT in SPECTER reduces performance considerably. This is reasonable because unlike BERT-Large, SciBERT is pretrained on scientific text.

Visualization Figure 2 shows t-SNE (van der Maaten, 2014) projections of our embeddings (SPECTER) compared with the SciBERT baseline for a random set of papers. When comparing SPECTER embeddings with SciBERT, we observe that our embeddings are better at encoding topical information, as the clusters seem to be more compact. Further, we see some examples of cross-

[12] We experimented with both concatenating authors with the title and abstract and also considering them as an additional field. Neither were helpful.

[13] Venue information in our data came directly from publisher provided metadata and thus was not normalized. Venue normalization could help improve results.



(a) SPECTER (b) SciBERT

Figure 2: t-SNE visualization of paper embeddings and their corresponding MAG topics.

topic relatedness reflected in the embedding space (e.g., Engineering, Mathematics and Computer Science are close to each other, while Business and Economics are also close to each other). To quantify the comparison of visualized embeddings in Figure 2, we use the DBScan clustering algorithm (Ester et al., 1996) on this 2D projection. We use the completeness and homogeneity clustering quality measures introduced by Rosenberg and Hirschberg (2007). For the points corresponding to Figure 2, the homogeneity and completeness values for SPECTER are respectively 0.41 and 0.72 compared with SciBERT's 0.19 and 0.63, a clear improvement on separating topics using the projected embeddings.

Comparison with Task Specific Fine-Tuning While the fact that SPECTER does not require fine-tuning makes its paper embeddings less costly to use, often the best performance from pretrained Transformers is obtained when the models are fine-tuned directly on each end task. We experiment with fine-tuning SciBERT on our tasks, and find this to be generally inferior to using our fixed representations from SPECTER. Specifically, we fine-tune SciBERT directly on task-specific signals instead of citations. To fine-tune on task-specific data (e.g., user activity), we used a dataset of co-views with 65K query papers, co-reads with 14K query papers, and co-citations (instead of direct citations) with 83K query papers. As the end tasks are ranking tasks, for all datasets we construct up to 5 triplets and fine-tune the model using triplet ranking loss. The positive papers are sampled from the most co-viewed (co-read, or co-cited) papers corresponding to the query paper. We also include both easy and hard distractors as when training SPECTER (for hard negatives we choose the least non-zero co-viewed (co-read, or co-cited) papers).

Training signal	CLS	USR	CITE	REC	All
SPECTER	**84.4**	**88.2**	**91.5**	**37.2**	**80.2**
SciBERT fine-tune on co-view	83.0	84.2	84.1	36.4	76.0
SciBERT fine-tune on co-read	82.3	85.4	86.7	36.3	77.1
SciBERT fine-tune on co-citation	82.9	84.3	85.2	36.6	76.4
SciBERT fine-tune on multitask	83.3	86.1	88.2	36.0	78.0

Table 3: Comparison with task-specific fine-tuning.

We also consider training jointly on all task-specific training data sources in a multitask training process, where the model samples training triplets from a distribution over the sources. As illustrated in Table 3, without any additional final task-specific fine-tuning, SPECTER still outperforms a SciBERT model fine-tuned on the end tasks as well as their multitask combination, further demonstrating the effectiveness and versatility of SPECTER embeddings.

7 Related Work

Recent representation learning methods in NLP rely on training large neural language models on unsupervised data (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019; Beltagy et al., 2019; Liu et al., 2019). While successful at many sentence- and token-level tasks, our focus is on using the models for document-level representation learning, which has remained relatively under-explored.

There have been other efforts in document representation learning such as extensions of word vectors to documents (Le and Mikolov, 2014; Ganesh et al., 2016; Liu et al., 2017; Wu et al., 2018; Gysel et al., 2017), convolution-based methods (Liu et al., 2018; Zamani et al., 2018), and variational autoencoders (Holmer and Marfurt, 2018; Wang et al., 2019). Relevant to document embedding, sentence embedding is a relatively well-studied area of research. Successful approaches include seq2seq models (Kiros et al., 2015), BiLSTM Siamese networks (Williams et al., 2018), leveraging supervised data from other corpora (Conneau et al., 2017), and using discourse relations (Nie et al., 2019), and BERT-based methods (Reimers and Gurevych, 2019). Unlike our proposed method, the majority of these approaches do not consider any notion of inter-document relatedness when embedding documents.

Other relevant work combines textual features with network structure (Tu et al., 2017; Zhang et al., 2018; Bhagavatula et al., 2018; Shen et al., 2018; Chen et al., 2019; Wang et al., 2019). These works

typically do not leverage the recent pretrained contextual representations and with a few exceptions such as the recent work by Wang et al. (2019), they cannot generalize to unseen documents like our SPECTER approach. Context-based citation recommendation is another related application where models rely on citation contexts (Jeong et al., 2019) to make predictions. These works are orthogonal to ours as the input to our model is just paper title and abstract. Another related line of work is graph-based representation learning methods (Bruna et al., 2014; Kipf and Welling, 2017; Hamilton et al., 2017a,b; Wu et al., 2019a,b). Here, we compare to a graph representation learning model, SGC (Simple Graph Convolution) (Wu et al., 2019a), which is a state-of-the-art graph convolution approach for representation learning. SPECTER uses pretrained language models in combination with graph-based citation signals, which enables it to outperform the graph-based approaches in our experiments.

SPECTER embeddings are based on only the title and abstract of the paper. Adding the full text of the paper would provide a more complete picture of the paper's content and could improve accuracy (Cohen et al., 2010; Lin, 2008; Schuemie et al., 2004). However, the full text of many academic papers is not freely available. Further, modern language models have strict memory limits on input size, which means new techniques would be required in order to leverage the entirety of the paper within the models. Exploring how to use the full paper text within SPECTER is an item of future work.

Finally, one pain point in academic paper recommendation research has been a lack of publicly available datasets (Chen and Lee, 2018; Kanakia et al., 2019). To address this challenge, we release SCIDOCS, our evaluation benchmark which includes an anonymized clickthrough dataset from an online recommendations system.

8 Conclusions and Future Work

We present SPECTER, a model for learning representations of scientific papers, based on a Transformer language model that is pretrained on citations. We achieve substantial improvements over the strongest of a wide variety of baselines, demonstrating the effectiveness of our model. We additionally introduce SCIDOCS, a new evaluation suite consisting of seven document-level tasks and release the corresponding datasets to foster further research in this area.

The landscape of Transformer language models is rapidly changing and newer and larger models are frequently introduced. It would be interesting to initialize our model weights from more recent Transformer models to investigate if additional gains are possible. Another item of future work is to develop better multitask approaches to leverage multiple signals of relatedness information during training. We used citations to build triplets for our loss function, however there are other metrics that have good support from the bibliometrics literature (Klavans and Boyack, 2006) that warrant exploring as a way to create relatedness graphs. Including other information such as outgoing citations as additional input to the model would be yet another area to explore in future.

Acknowledgements

We thank Kyle Lo and Daniel King for helpful research discussions, Russel Reas for setting up the public API, Field Cady for help in initial data collection and the anonymous reviewers (especially Reviewer 1) for comments and suggestions.

References

Anant K. Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Yen Li, Marc Najork, and Thorsten Joachims. 2019. Estimating position bias without intrusive interventions. In *WSDM*.

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 C. Murray, Hsu-Han Ooi, Matthew E. Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Christopher Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the literature graph in semantic scholar. In *NAACL-HLT*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *EMNLP*.

Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar. 2018. Content-Based Citation Recommendation. In *NAACL-HLT*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. *ICLR*.

Liqun Chen, Guoyin Wang, Chenyang Tao, Dinghan Shen, Pengyu Cheng, Xinyuan Zhang, Wenlin Wang, Yizhe Zhang, and Lawrence Carin. 2019. Improving textual network embedding with global attention via optimal transport. In *ACL*.

Tsung Teng Chen and Maria Lee. 2018. Research Paper Recommender Systems on Big Scholarly Data. In *Knowledge Management and Acquisition for Intelligent Systems*.

K. Bretonnel Cohen, Helen L. Johnson, Karin M. Verspoor, Christophe Roeder, and Lawrence Hunter. 2010. The structural and content aspects of abstracts versus bodies of full text journal articles are different. *BMC Bioinformatics*, 11:492–492.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*.

Sergey Feldman, Waleed Ammar, Kyle Lo, Elly Trepman, Madeleine van Zuylen, and Oren Etzioni. 2019. Quantifying Sex Bias in Clinical Studies at Scale With Automated Data Extraction. *JAMA*.

J Ganesh, Manish Gupta, and Vijay K. Varma. 2016. Doc2sent2vec: A novel two-phase approach for learning document representation. In *SIGIR*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A Deep Semantic Natural Language Processing Platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*.

Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2017. Neural Vector Spaces for Unsupervised Information Retrieval. *ACM Trans. Inf. Syst.*

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017a. Inductive Representation Learning on Large Graphs. In *NIPS*.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017b. Inductive representation learning on large graphs. In *NIPS*.

Erik Holmer and Andreas Marfurt. 2018. Explaining away syntactic structure in semantic document representations. *ArXiv*, abs/1806.01620.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *ACL*.

Chanwoo Jeong, Sion Jang, Hyuna Shin, Eunjeong Lucy Park, and Sungchul Choi. 2019. A context-aware citation recommendation model with bert and graph convolutional networks. *ArXiv*, abs/1903.06464.

Anshul Kanakia, Zhihong Shen, Darrin Eide, and Kuansan Wang. 2019. A Scalable Hybrid Research Paper Recommender System for Microsoft Academic. In *WWW*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *ArXiv*, abs/1412.6980.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *ICLR*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Richard Klavans and Kevin W. Boyack. 2006. Identifying a better measure of relatedness for mapping science. *Journal of the Association for Information Science and Technology*, 57:251–263.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Rep4NLP@ACL*.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*.

Jimmy J. Lin. 2008. Is searching full text more effective than searching abstracts? *BMC Bioinformatics*, 10:46–46.

Carolyn E Lipscomb. 2000. Medical Subject Headings (MeSH). *Bulletin of the Medical Library Association*.

Chundi Liu, Shunan Zhao, and Maksims Volkovs. 2018. Unsupervised Document Embedding with CNNs. *ArXiv*, abs/1711.04168v3.

Pengfei Liu, King Keung Wu, and Helen M. Meng. 2017. A Model of Extended Paragraph Vector for Document Categorization and Trend Analysis. *IJCNN*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar S. Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke S. Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, abs/1907.11692.

Laurens van der Maaten. 2014. Accelerating t-SNE Using Tree-based Algorithms. *Journal of Machine Learning Research*.

Allen Nie, Erin Bennett, and Noah Goodman. 2019. DisSent: Learning Sentence Representations from Explicit Discourse Relations. In *ACL*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *arXiv*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *LREC*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP*.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A Conditional Entropy-based External Cluster Evaluation Measure. In *EMNLP*.

J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer.

Martijn J. Schuemie, Marc Weeber, Bob J. A. Schijvenaars, Erik M. van Mulligen, C. Christiaan van der Eijk, Rob Jelier, Barend Mons, and Jan A. Kors. 2004. Distribution of information in biomedical abstracts and full-text publications. *Bioinformatics*, 20(16):2597–604.

Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Improved semantic-aware network embedding with fine-grained word alignment. In *EMNLP*.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Paul Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *WWW*.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *NIPS*.

Wenlin Wang, Chenyang Tao, Zhe Gan, Guoyin Wang, Liqun Chen, Xinyuan Zhang, Ruiyi Zhang, Qian Yang, Ricardo Henao, and Lawrence Carin. 2019. Improving textual network learning with variational homophilic embeddings. In *Advances in Neural Information Processing Systems*, pages 2074–2085.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *NAACL-HLT*.

Felix Wu, Amauri H. Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019a. Simplifying graph convolutional networks. In *ICML*.

Lingfei Wu, Ian En-Hsu Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J Witbrock. 2018. Word Mover's Embedding: From Word2Vec to Document Embedding. In *EMNLP*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019b. A Comprehensive Survey on Graph Neural Networks. *ArXiv*, abs/1901.00596.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *ArXiv*, abs/1906.08237.

Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik G. Learned-Miller, and Jaap Kamps. 2018. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *CIKM*.

Xinyuan Zhang, Yitong Li, Dinghan Shen, and Lawrence Carin. 2018. Diffusion maps for textual network embedding. In *NeurIPS*.

A Appendix A - Baseline Details

1. **Random** Zero-mean 25-dimensional vectors were used as representations for each document.

2. **Doc2Vec** Doc2Vec is one of the earlier neural document/paragraph representation methods (Le and Mikolov, 2014), and is a natural comparison. We trained Doc2Vec on our training subset using Gensim (Řehůřek and Sojka, 2010), and chose the hyperparameter grid using suggestions from Lau and Baldwin (2016). The hyperparameter grid used:

```
{'window': [5, 10, 15],
 'sample': [0, 10 ** -6, 10 ** -5],
 'epochs': [50, 100, 200]},
```

for a total of 27 models. The other parameters were set as follows: `vector_size=300`, `min_count=3`, `alpha=0.025`, `min_alpha=0.0001`, `negative=5`, `dm=0`, `dbow=1`, `dbow_words=0`.

3. **Fasttext-Sum** This simple baseline is a weighted sum of pretrained word vectors. We trained our own 300 dimensional fasttext embeddings (Bojanowski et al., 2017) on a corpus of around 3.1B tokens from scientific papers which is similar in size to the SciBERT corpus (Beltagy et al., 2019). We found that these pretrained embeddings substantially outperform alternative off-the-shelf embeddings. We also use these embeddings in other baselines that require pretrained word vectors (i.e., SIF and SGC that are described below). The summed bag of words representation has a number of weighting options, which are extensively tuned on a validation set for best performance.

4. **SIF** The SIF method of Arora et al. (2017) is a strong text representation baseline that takes a weighted sum of pretrained word vectors (we use fasttext embeddings described above), then computes the first principal component of the document embedding matrix and subtracts out each document embedding's projection to the first principal component.

We used a held-out validation set to choose a from the range [1.0e-5, 1.0e-3] spaced evenly on a log scale. The word probability $p(w)$ was estimated on the training set only. When computing term-frequency values for SIF, we used scikit-learn's TfidfVectorizer with the same parameters as enumerated in the preceding section. `sublinear_tf`, `binary`, `use_idf`, `smooth_idf` were all set to `False`. Since SIF is a sum of pretrained fasttext vectors, the resulting dimensionality is 300.

5. **ELMo** ELMo (Peters et al., 2018) provides contextualized representations of tokens in a document. It can provide paragraph or document embeddings by averaging each token's representation for all 3 LSTM layers. We used the 768-dimensional pretrained ELMo model in AllenNLP (Gardner et al., 2018).

6. **Citeomatic** The most relevant baseline is Citeomatic (Bhagavatula et al., 2018), which is an academic paper representation model that is trained on the citation graph via sampled triplets. Citeomatic representations are an L2 normalized weighted sum of title and abstract embeddings, which are trained on the citation graph with dynamic negative sampling. Citeomatic embeddings are 75-dimensional.

7. **SGC** Since our algorithm is trained on data from the citation graph, we also compare to a state-of-the-art graph representation learning model: SGC (Simple Graph Convolution) (Wu et al., 2019a), which is a graph convolution network. An alternative comparison would have been GraphSAGE (Hamilton et al., 2017b), but SGC (with no learning) outperformed an unsupervised variant of GraphSAGE on the Reddit dataset[14], Note that SGC with no learning boils down to graph propagation on node features (in our case nodes are academic documents). Following Hamilton et al. (2017a), we used SIF features as node representations, and applied SGC with a range of parameter k, which is the number of times the normalized adjacency is multiplied by the SIF feature matrix. Our range of k was 1 through 8 (inclusive), and was chosen with a validation set. For the node features, we chose the SIF model with $a = 0.0001$, as this model was observed to be a high-performing one. This baseline is also 300 dimensional.

8. **SciBERT** To isolate the advantage of SPECTER's citation-based fine-tuning objective, we add a controlled comparison with SciBERT (Beltagy et al., 2019). Following Devlin et al. (2019) we take the last layer hidden state corresponding to the `[CLS]` token as the aggregate document representation.[15]

9. **Sentence BERT** Sentence BERT (Reimers and Gurevych, 2019) is a general-domain pretrained model aimed at embedding sentences. The authors fine-tuned BERT using a triplet loss, where positive sentences were from the same document section as the seed sentence, and distractor sentences came from other document sections. The model is designed to encode sentences as opposed to paragraphs, so we embed the title and each sentence in the abstract separately, sum the embeddings, and L2 normalize the result to produce a

[14] There were no other direct comparisons in Wu et al. (2019a)

[15] We also tried the alternative of averaging all token representations, but this resulted in a slight performance decrease compared with the `[CLS]` pooled token.

final 768-dimensional paper embedding.[16]

During hyperparameter optimization we chose how to compute TF and IDF values weights by taking the following non-redundant combinations of scikit-learn's TfidfVectorizer (Pedregosa et al., 2011) parameters: `sublinear_tf`, `binary`, `use_idf`, `smooth_idf`. There were a total of 9 parameter combinations. The IDF values were estimated on the training set. The other parameters were set as follows: `min_df=3`, `max_df=0.75`, `strip_accents='ascii'`, `stop_words='english'`, `norm=None`, `lowercase=True`. For training of fasttext, we used all default parameters with the exception of setting dimension to 300 and `minCount` was set to 25 due to the large corpus.

[16]We used the 'bert-base-wikipedia-sections-mean-tokens' model released by the authors: https://github.com/UKPLab/sentence-transformers