Transformer-based Models for Long-Form Document Matching: Challenges and Empirical Analysis

Akshita Jha
Virginia Tech, Arlington, VA
akshitajha@vt.edu

Adithya Samavedhi
Virginia Tech, Arlington, VA
adithyas@vt.edu

Vineeth Rakesh
InterDigital, CA
vineethrakesh@gmail.com

Jaideep Chandrashekar
InterDigital, CA
jaideep.chandrashekar@interdigital.com

Chandan K. Reddy
Virginia Tech, Arlington, VA
reddy@cs.vt.edu

Abstract

Recent advances in the area of long document matching have primarily focused on using transformer-based models for long document encoding and matching. There are two primary challenges associated with these models. Firstly, the performance gain provided by transformer-based models comes at a steep cost – both in terms of the required training time and the resource (memory and energy) consumption. The second major limitation is their inability to handle more than a pre-defined input token length at a time. In this work, we empirically demonstrate the effectiveness of simple neural models (such as feed-forward networks, and CNNs) and simple embeddings (like GloVe, and Paragraph Vector) over transformer-based models on the task of document matching. We show that simple models outperform the more complex BERT-based models while taking significantly less training time, energy, and memory. The simple models are also more robust to variations in document length and text perturbations.

1 Introduction

Matching long documents (*e.g.:* research papers, Wikipedia articles, patents, etc.) is an important task that can help understand the (dis)similarity between documents for downstream tasks like long document search. The first step towards better document matching is obtaining meaningful long document representations. Recent advances in this area have primarily focused on using transformer-based models for long document encoding and matching (Beltagy et al., 2020; Jha et al., 2022; Yang et al., 2020b; Zaheer et al., 2020). We use the term transformers to mean pre-trained transformers. Despite promising results, there are two primary challenges associated with such models. First, the performance gain provided by the huge transformer-based language models (LMs), like BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019), and Longformer (Beltagy et al., 2020) come at a steep cost – both in terms of the required training time, and the resource (memory and energy) consumption. For example, the smaller $BERT_{BASE}$ model has 110 million parameters, whereas the bigger $BERT_{LARGE}$ model has a total of 340 million parameters and fine-tuning a single $BERT_{BASE}$ model on GPU can take hours. The second major limitation of transformer-based models is their inability to handle more than a pre-defined input token length at a time (512 tokens for BERT, and 4096 tokens for Longformer). This is a big drawback as they cannot handle long documents like research papers, patents, long articles, etc., without using aggregation techniques (Reimers and Gurevych, 2019).

In this work, we empirically demonstrate that *embeddings obtained from GloVe (Pennington et al., 2014), and Paragraph Vectors (Le and Mikolov, 2014)* along with simple neural models, such as feed-forward networks, and CNNs, *outperform several transformer-based models for the document matching task.* We define these models as simple as they take significantly less time to train and consume less memory and energy overall when compared to complex transformer-based models. Our long document matching setting is fundamentally different from long-form question answering and sentence similarity tasks. For the latter tasks the query is 'short', unlike the long document matching task where both the query and the target text are 'long'. We experiment with three different kinds of semi-structured long document datasets in English: (i) ACL Anthology research papers, (ii) English Wikipedia articles, and (iii) Patents from US Patent and Trademark Office (USPTO). Our primary contributions are summarized as follows:

- We provide insights into the challenges of using transformer-based models for the task of long document matching. For this task, simple neural models are as effective and take a

Findings of the Association for Computational Linguistics: EACL 2023, pages 2345–2355

fraction of the training time and resources to outperform transformer-based models.

- We provide insights into the best input embeddings for the simpler models in this task.
- We demonstrate that simple models are also more robust to changes in document length and text perturbation.
- We create benchmark long document datasets (by pre-processing ACL Anthology 2014 papers and Wikipedia articles) that will be made publicly available.

2 Related Work

Early work on long document matching focused on clustering techniques (Friburger et al., 2002; Huang et al., 2008; Strehl et al., 2000). Recently, Guo et al. (2016) proposed a deep learning based architecture for ad-hoc retrieval when comparing documents. Some works have also used convolutional networks (Hu et al., 2014; Pang et al., 2016; Yu et al., 2018), with weighting mechanism (Yang et al., 2016a) to generate a final query-document score. Mitra et al. (2017) propose a combination model that uses weighted sum representation-based and interaction-based results. Yang et al. (2016b) propose HAN, a hierarchical attention network for document matching. Jiang et al. (2019) propose SMASH, a multi-depth attention based hierarchical recurrent neural network for long-document matching. However, Yang et al. (2020b) pre-train SMITH, a transformer based hierarchical model for text matching that outperforms SMASH across multiple datasets. Jha et al. (2022) use supervised contrative learning for interpretable long document matching. We compare several of these models on the required training time and resources.

A growing body of literature has used transformer based model for long document encoding (Child et al., 2019; Ho et al., 2019; Kitaev et al., 2019; Liu and Lapata, 2019; Qiu et al., 2020; Yang et al., 2020a). Longformer (Beltagy et al., 2020) adapts transformers to use an attention mechanism that scales linearly with sequence length. Big bird (Zaheer et al., 2020) uses a sparse attention mechanism that reduces BERT's quadratic dependency on the sequence length to linear. CogLTX (Ding et al., 2020) uses text blocks for rehearsal and decay over key sentences to overcome the insufficient long-range attentions in BERT. Transformer-XL (Dai et al., 2019) and Compressive Tranformers (Rae et al., 2019) compress the transformers to use attentive sequence over long text. Although promising, we demonstrate that transformer-based models are not considerably better than simple neural models on the task of long document matching.

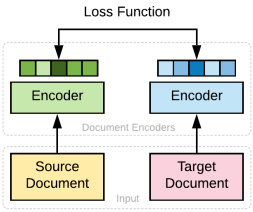


Figure 1: A schematic siamese comparison model

3 Empirical Evaluation

Here we provide details of the simple and the transformer-based models and present an empirical comparison between them based on their overall performance, training time, resource consumption, and robustness on the document matching task.

Task Formulation We define the task of document matching as follows. Given a source document s, and a set of target documents D_T, the goal is to estimate the semantic match $\hat{y} = sim(s, t)$, where $t \in D_T$ for every document pair (s, t). Similar target documents will have a higher similarity score. The document matching problem can be seen as a binary classification task, where the model predicts 1 for similar documents, and 0 for dissimilar documents. We use the term "matching" in the broad sense of document relevance (see Appendix A.2). The models take as input a pair of documents (source and target), and compute the cosine similarity between the encoded document representations. If the cosine similarity is greater than a similarity threshold θ, they are considered similar; otherwise they are considered dissimilar.

Models We pick a representative set of models from different categories and compare them by building their siamese versions (shown in Figure 1). The siamese network has three primary components: (i) Input (Source and Target Document), (ii) Document Encoder, and (iii) Loss Function. The source and target document encoder networks share weights. We experiment with three simple neural models: (i) DSSM: A simple feed-forward network (Huang et al., 2013), (ii) ARC-I: A CNN-based model (Hu et al., 2014) , and (iii) HAN: An RNN-based Hierarchical Attention Network (Yang et al., 2016b) designed for long documents. Their performance is compared with three state-of-the-art transformer-based models: (i)

	AAN				WIKI				PAT			
Model	**P**	**R**	**F1**	**Acc**	**P**	**R**	**F1**	**Acc**	**P**	**R**	**F1**	**Acc**
HAN-G	0.504	0.881	0.641	0.607	0.566	0.584	0.575	0.775	0.609	0.848	0.709	0.522
DSSM-T	0.768	0.809	0.787	0.780	0.823	0.939	0.877	0.869	0.869	0.957	0.911	0.905
DSSM-G	0.550	0.541	0.545	0.549	0.966	0.986	0.975	0.975	0.992	**0.998**	**0.995**	**0.995**
DSSM-D	**0.852**	0.763	**0.805**	**0.815**	0.933	0.984	0.958	0.957	0.841	0.959	0.896	0.949
ARC-I-G	0.643	0.873	0.734	0.743	**0.992**	**0.983**	**0.987**	**0.987**	0.905	0.963	0.933	0.939
ARC-I-D	0.841	0.763	0.800	0.809	0.987	0.985	0.986	0.986	0.967	0.958	0.962	0.983
BERT	0.760	**0.914**	0.793	0.761	0.980	0.950	0.960	0.960	**1.0**	0.988	0.994	0.994
LONG	0.681	0.833	0.749	0.773	0.974	0.960	0.967	0.967	**1.0**	0.984	0.992	0.992
SMITH	0.726	0.565	0.635	0.676	0.949	0.982	0.965	0.963	0.892	0.939	0.865	0.939

Table 1: Performance on the document matching task up to the model's maximum allowed input token length (512 for BERT; 4096 for Longformer, $>$ 8000 for all other models). We experiment with Trigrams (T), GloVe (G), and Doc2Vec (D) Embeddings as input for the simple neural models. The best performance is highlighted in bold.

BERT (Devlin et al., 2019), (ii) LONG: Longformer (Beltagy et al., 2020), and (iii) SMITH: Siamese Multi-depth Transformer based Hierarchical Encoder (Yang et al., 2020b). We report the mean precision, recall, F1, and accuracy over 5 folds for the best performing hyper-parameters. The code can be found here: https://github.com/AkshitaJha/SimpleModelsforLongDocumentMatching.

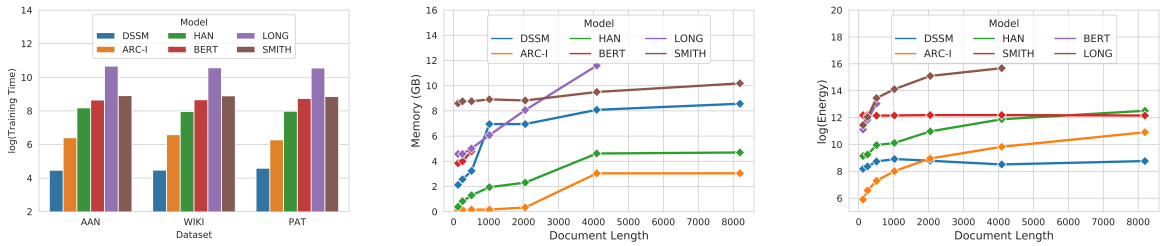
Datasets We follow the previous literature (Yang et al., 2016b; Jiang et al., 2019; Yang et al., 2020b) and experiment with the following three standard long document datasets: (i) ACL Anthology Network Corpus (AAN)[1], (ii) Wikipedia Articles (WIKI)[2], and (iii) USPTO Patents (PAT)[3]. Each dataset consists of balanced 15,000 pairs of documents with 50% of them being similar pairs, and the remaining being dissimilar. The PAT dataset is an industry gold standard but we will publicly release the pre-processed AAN and the WIKI datasets (see Appendix A.2 for details). The dataset can be found here: https://github.com/AkshitaJha/SimpleModelsforLongDocumentMatching

Performance on Document Matching Task We experiment with three input representations for simple neural models: (i) char-Trigram Hashing (T) (Huang et al., 2013), (ii) GloVe Embeddings (G) (Pennington et al., 2014), and (iii) Paragraph Vector/Doc2Vec Embeddings (D) (Le and Mikolov, 2014) (See Appendix 4). Unlike most transformer-based models that take as input tokens up to a pre-defined length (512 for BERT, and 4096 for Longformer), simple models and SMITH have the ability to take the entire long document ($>$ 8000 tokens) as input. Table 1 demonstrates the performance of different models on the task of document matching up to their maximum allowed input document length (see Appendix A.8 for different document lengths.) We observe that despite being relatively simple and not taking into account contextual embeddings, *DSSM and ARC-I outperform the transformer based models using GloVe and Doc2Vec Embeddings* on the AAN, WIKI, and the PAT dataset.

Training Time Figure 2a shows the training time taken to reach the best performance for every model for their maximum allowed input token lengths. We only report the fine-tuning time after downloading the pre-trained models. All experiments were done on a 16GiB Tesla V100. *The simple models like DSSM, ARC-I, and HAN take 1/12 to 1/15 of the training time taken by the transformer-based models to outperform them on all three datasets* (see Appendix A.4 for the training time for different document lengths on all datasets.)

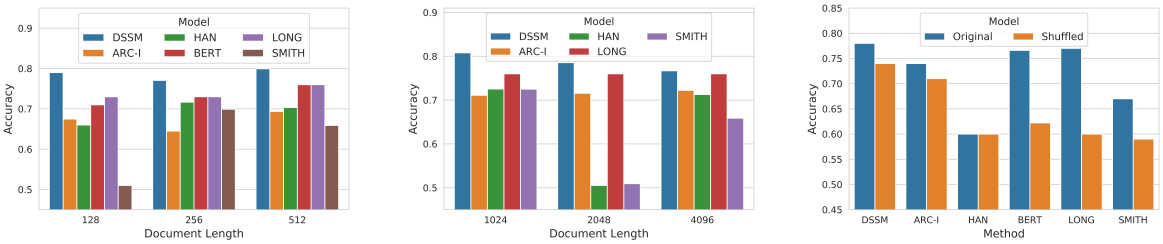
Memory and Energy consumption Memory consumption on a 16GiB Tesla V100, for a batch size of 1, for different models can be seen in Figure 2b. Compared to transformer-based models simple neural models consume significantly less memory for the same document length (12 GiB for Longformer vs. a maximum of 8 GiB for DSSM for 4000 tokens). We also compute the overall energy required for training the models to achieve their best performance (Figure 2c) by measuring the power consumption of the GPU over their training lifetime. Longformer consumes $>$ 6MJ of energy for fine-tuning on documents with 4096 tokens, BERT consumes $\approx$ 500kJ of energy for fine-tuning on documents with just 512 tokens, and the SMITH model consumes $\approx$ 200kJ of energy for fine-tuning on longer documents; whereas the simple models consume $<$ 100kJ of energy for training from

[1] https://aan.how/download/#aanNetworkCorpus
[2] https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2
[3] https://github.com/google/patents-public-data



(a) Training time on the *log* scale (b) Memory consumption (c) Energy Consumption on *log* scale

Figure 2: Comparison of simple neural models with transformer-based models based on (a) Training Time, (b) Memory Consumption, and (c) Energy Consumption.



(a) Comparison with BERT [512 tokens] (b) Comparison with LONG [4096 tokens] (c) Original vs. Shuffled Documents

Figure 3: Comparison between the robustness of simple neural models and transformer-based models w.r.t. document length and text perturbation on the AAN dataset.

scratch for documents with $>$ 8000 tokens.

Robustness to Document Length We limit the maximum number of tokens in each document during training and testing, and observe the final test accuracy on the document matching task. It should be noted that documents of different lengths are actually 'truncated long documents' without the complete contextual information needed to compute the actual similarity between two long documents. Figure 3 compares the model accuracy of simple models (with their default input embeddings) with transformer-based models upto their maximum allowed token lengths for the AAN dataset – 512 tokens for BERT (Figure 3a), and 4096 tokens for Longformer (Figure 3b). DSSM outperforms the baseline models for all documents lengths. BERT and Longformer have a consistent performance on AAN for different input lengths, unlike HAN and SMITH that are not as robust to the variations in document length, although they were designed specifically for long documents. We found similar results for WIKI and PAT dataset (see Appendix A.5). We also experiment with documents of length $>$ 512 tokens for BERT, and $>$ 4096 tokens for Longformer by aggregating the chunk representations upto their maximum allowed token length. We used the SUM and AVG aggregation techniques and observed an overall performance drop (see Table 5).

Robustness to Text Perturbation For text perturbation, we split documents into paragraphs of 512 tokens and randomly shuffle the order of these paragraphs before training different models to check for learned positional bias. We measure their test accuracy on the original document matching task. Figure 3c shows the model performance for all the baseline methods on AAN dataset. We observe a significant drop in the model performance for transformer-based models (BERT, Longformer, and SMITH). There is no significant change in the accuracy for the simple models – DSSM, ARC-I, and HAN. The transformer based models are more sensitive to text perturbation. The simple models, on the other hand, use non-contextual embeddings, such as GloVE, and Doc2Vec and are more robust to text perturbation (see Appendix A.6).

4 Conclusion

We empirically demonstrate the trade-off of using transformer-based models for semi-structured long English documents like research papers, Wikipedia articles, and patents. Transformer-based models have an overall strong performance and smaller variability across datasets. However, we observe that for the task of long document matching, using contextual embeddings do not provide any added advantage. **A simple feed-forward network or a CNN-based model using GloVe or Doc2Vec embedding outperforms several state-of-the-art**

pre-trained transformer-based models at a fraction of their overall training time and resources (memory and energy). These simple neural models are also more robust to changes in document length and text-perturbation.

5 Limitations

One of the limitations of our work is that we experimented only with long documents in English. Comparing simple neural models and transformer-based models in different languages would be an interesting study but is outside the scope of this short paper. We would also like to highlight that we use classification metrics instead of information-retrieval metrics due to the limitations of the dataset which has very few positive samples (2-3) for every document.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Cogltx: Applying bert to long texts. *Advances in Neural Information Processing Systems*, 33.

Nathalie Friburger, Denis Maurel, and Arnaud Giacometti. 2002. Textual similarity based on proper names. In *Proc. of the workshop Mathematical/Formal Methods in Information Retrieval*, pages 155–167.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 55–64.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. 2019. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. *Advances in neural information processing systems*, 27:2042–2050.

Anna Huang et al. 2008. Similarity measures for text document clustering. In *Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand*, volume 4, pages 9–56.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338.

Akshita Jha, Vineeth Rakesh, Jaideep Chandrashekar, Adithya Samavedhi, and Chandan K. Reddy. 2022. Supervised contrastive learning for interpretable long-form document matching.

Jyun-Yu Jiang, Mingyang Zhang, Cheng Li, Michael Bendersky, Nadav Golbandi, and Marc Najork. 2019. Semantic text matching for long-form documents. In *The World Wide Web Conference*, pages 795–806.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2019. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1291–1299.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. 2020. Blockwise self-attention for long document understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2555–2565.

Dragomir R. Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. 2013. The acl anthology network corpus. *Language Resources and Evaluation*, pages 1–26.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. 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 3973–3983.

Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. 2000. Impact of similarity measures on web-page clustering. In *Workshop on artificial intelligence for web search (AAAI 2000)*, volume 58, page 64.

Linyi Yang, Tin Lok James Ng, Barry Smyth, and Riuhai Dong. 2020a. Html: Hierarchical transformer-based multi-task learning for volatility prediction. In *Proceedings of The Web Conference 2020*, pages 441–451.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. 2016a. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 287–296.

Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020b. Beyond 512 tokens: Siamese multi-depth transformer-based hierarchical encoder for long-form document matching. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1725–1734.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016b. 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.

Jianfei Yu, Minghui Qiu, Jing Jiang, Jun Huang, Shuangyong Song, Wei Chu, and Haiqing Chen. 2018. Modelling domain relationships for transfer learning on retrieval-based question answering systems in e-commerce. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 682–690.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*.

A Appendix

A.1 Model Description

The encoder networks and their default inputs have been described below:

A.1.1 Simple Models

- **DSSM** (Huang et al., 2013): A *simple three-layered feed forward network* that takes as input the vectorized representation of a document.

- **ARC-I** (Hu et al., 2014): A *CNN-based model* that takes as input a 2D-matrix representation of a document where words in the sentences are represented using *GloVe embeddings* (Pennington et al., 2014). These are then passed through convolutional and max-pooling layers to finally obtain a document representation for both source and target documents, independently. The document representations are concatenated and passed through a multi-layer perceptron to predict if the pair of documents are similar or not.

- **Hierarchical Attention Network (HAN)** (Yang et al., 2016b): A hierarchical *GRU-based model* with attention mechanism that aggregates *GloVe embeddings* at word level into sentence representations to arrive at the final document representation.

A.1.2 Transformer-Based Models

- **BERT** (Devlin et al., 2019): A siamese matching model with *BERT*. For long document inputs, BERT only uses the first 512 tokens of each document. We use a pre-trained $BERT_{BASE}$ model which is fine-tuned during training.

- **Longformers (LONG)** (Beltagy et al., 2020) : A siamese model with transformer-based *Longformers* for long sequences. It has an attention mechanism that scales linearly with sequence length and takes as input a maximum of 4096 tokens. We consider an attention window of size 256.

- **Siamese Multi-depth Transformer based Hierarchical Encoder (SMITH)** (Yang et al., 2020b): A *transformer-based hierarchical encoder* which is the current SOTA model for long-form document matching task.

A.1.3 Implementation Details

For all the models presented in the paper, we use the same architecture as the original papers. We tune the hyperparamters and report the best results. The DSSM, ARC-I, HAN, and SMITH models were implemented in Tensorflow. BERT and Longformer were implemented in PyTorch. DSSM has hidden units of dimension 300 for its hidden layers and an output dimension of 128. The learning rate was 0.0075. ARC-I takes as input a 2D matrix of the size [no. of sentences x sentence length]. This is given as input to two 1D-convolutional (filter size of 200, kernel size 3) and MaxPooling layers of size 2, in order to get the final document representations. The representations of both the source and the target documents are concatenated and passed through a two-layer multi-layer perceptron. The first hidden layer of the MLP has a dimension of 64 with ReLU activation. The second layer has 1 node and sigmoid activation which predicts if the pair of documents are similar or not. The learning rate was set to 0.00075. HAN uses a bi-Directional GRU layer and applies attention mechanism to arrive at final sentence representation for the source and the target documents with a learning rate of 0.001. We use the pre-trained $\text{BERT}_{\text{BASE}}$[4] and the Longformer [5] models provided by the Huggingface library. The SMITH code was publicly available. The BERT, Longformer, and SMITH models are fine-tuned during training. All other models are trained from scratch. The learning rate is set to 5e-5 for the transformer based models. We use an Adam optimizer for all models with a weight decay of 0.01. We use binary cross entropy as the loss function for the simple models, pairwise loss for BERT and Longformer, and triplet loss for SMITH. These loss functions resulted in the best performance for the model. The three datasets are split into 80-10-10 for train, validation, and test sets, respectively. We use cosine similarity and the similarity threshold θ is set to 0.5. We perform 5 fold cross-validation and use early stopping on the validation set to prevent over-fitting. The models were trained on one 16GB Tesla V100 GPU.

A.2 Dataset

- **ACL Anthology Network Corpus (AAN)**[6]: The AAN corpus (Radev et al., 2013) consists

[4] https://huggingface.co/transformers/model_doc/bert.html
[5] https://huggingface.co/transformers/longformer.html
[6] https://aan.how/download/#aanNetworkCorpus

of 23,766 papers written by 18,862 authors in 373 venues related to NLP and forms a citation network. Each paper is represented by a node with directed edges connecting a paper (the parent node) to all its cited papers (children nodes). Papers that have been cited by the parent paper are treated as similar samples (Jiang et al., 2019). For every similar sample, an irrelevant paper is randomly chosen to create a balanced dataset. Sets of similar papers are given the same labels. To prevent leakage of information and make the task more difficult, the references and the abstract sections are removed. Papers without any content are also removed. We then randomly sample 15,000 research paper pairs for our experiment.

- **Wikipedia (WIKI)**[7]: We use the Wikipedia dump, and adopt a similar methodology proposed by Jiang *et al.* (Jiang et al., 2019) to process this data. From the Wikipedia dump containing 6 million articles, we randomly sampled 250,000 articles along with the articles present in their outlinks. We create a dataset of similar Wikipedia articles by assuming that similar articles have similar outgoing links. The Jaccard similarity between the outgoing links of the source and the target articles is calculated. If the Jaccard similarity > 0.5, the documents are assumed to be similar, otherwise they are considered dissimilar. Only articles with two or more similar articles are selected. We then randomly sample 15,000 research paper pairs for our experiment.

- **Patent (PAT)**[8]: The patent dataset is an internally curated industry gold-standard. This dataset consists of patents sampled from the publicly available USPTO patents belonging to four different categories: video, wireless, image compression, and network compression. A patent document is extremely long and primarily consists of (i) Abstract, (ii) Claims, and (iii) Description sections. We only make use of the Claims and the Description sections for our experiments to prevent leakage of information from Abstracts. Three internal human annotators, with expert domain knowledge, were given pairs of documents and were asked to label them as similar or dissimilar based based on the technology presented in the patents. They referred to the Abstract, Claims, and CPC Codes[9] of the patents to measure the similarity. The final document content similarity label was based on majority vote.

The dataset statistics can be found in Table 2. We would like to note that although considering only the cited papers and outgoing links for AAN and Wikipedia articles, respectively is not the most optimal approach for creating similar document pairs, we adopt it for the following two reasons: (i) We do not have annotated fine-grained similarity scores for AAN and WIKI datasets, and (ii) We follow an approach similar to the previously published work (Jiang et al., 2019; Yang et al., 2016b, 2020b). We use the term "document matching" or "document similarity" in the broad sense of "citation matching" or "document relevance" – Given a pair of documents, are the two documents relevant to each other and should they be cited by each other.

A.3 Performance on Document Matching Task

Table 3 shows the average performance of simple models when compared to transformer based models on the document matching task for AAN, WIKI, and PAT datasets. ARC-I with Doc2Vec embeddings has the best average precision and accuracy. The average F1 score is comparable to BERT which re-emphasizes the benefits of using simple models for document matching.

A.4 Training Time for Different Document Lengths

The training time for different document lengths for all three datasets can be seen in Figure 4. BERT can only take up to 512 tokens. DSSM, ARC-1, and HAN take considerably less time to train when compared to transformer-based.

A.5 Robustness to Document Length

We check the robustness of simple models and transformer-based models for different document lengths on the task of document matching. From Figure 5 and Figure 6, we observe that the simple

[7] https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2

[8] https://github.com/google/patents-public-data

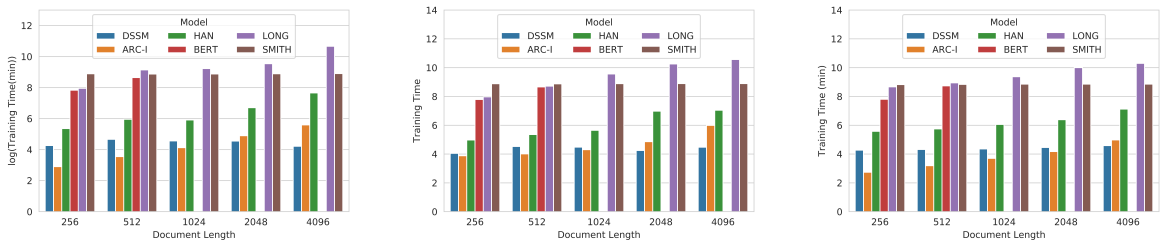
[9] https://www.uspto.gov/web/patents/classification/cpc/html/cpc.html

Dataset	Avg #Tokens	Max #Tokens	Avg #Sentences	Max #Sentences	Vocabulary
AAN	5,381.1	54,556	215.7	2,183	515,422
WIKI	3,777.0	26,172	190.6	1,685	1,151,309
PAT	8,177.4	50,322	214.1	2,709	220,023

Table 2: Dataset statistics

Model	P	R	F1	Acc
HAN-G	0.559	0.771	0.641	0.634
DSSM-T	0.820	0.901	0.858	0.851
DSSM-G	0.836	0.841	0.838	0.839
DSSM-D	0.875	0.902	0.886	0.907
ARC-I-G	0.846	0.939	0.884	0.889
ARC-I-D	**0.931**	0.902	**0.916**	**0.926**
BERT	0.913	**0.950**	0.915	0.905
LONG	0.885	0.925	0.902	0.910
SMITH	0.855	0.828	0.821	0.860

Table 3: Average performance across AAN, WIKI, and PAT datasets on the document matching task (shown in Table 1). We experiment with Trigrams (T), GloVe (G), and Doc2Vec (D) Embeddings as input for the simple neural models. The best performance is highlighted in bold.



(a) AAN (b) WIKI (c) PAT

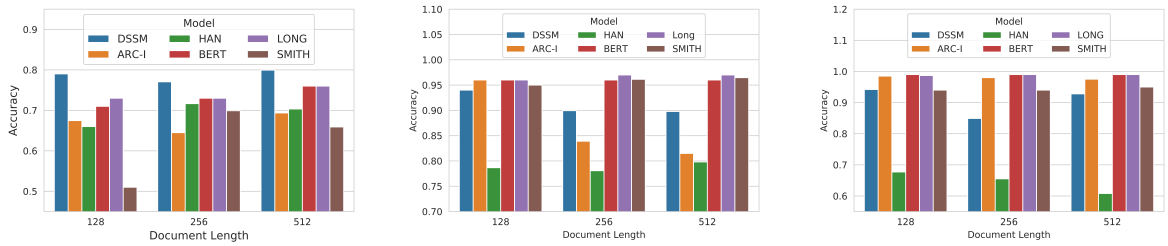
Figure 4: Doc Length vs Training Time (in *log* scale) for different document lengths.

models DSSM and ARC-I, and the transformer-based models BERT and Longformer, though not specifically designed for long documents are robust for different document lengths. BERT can only handle up to 512 tokens at a time, and Longformer can only handle up to 4096 input tokens. HAN and SMITH, on the other hand, were specially designed for long documents and have a high variance in their performance on the document matching tasks for different document lengths.

We also experimented with longer documents (> 512 tokens for BERT, and > 4096 tokens for Longformer). We obtained the final document representation by dividing the document into chunks of their maximum allowed token length. We then aggregated these chunk representations. We experimented with the SUM and AVG aggregation techniques by taking the representations of the '[CLS]' token and 'the pooler output' for these models. We observed an overall performance drop because of aggregation. The results were the same for both SUM and AVG aggregation techniques (Table 5).

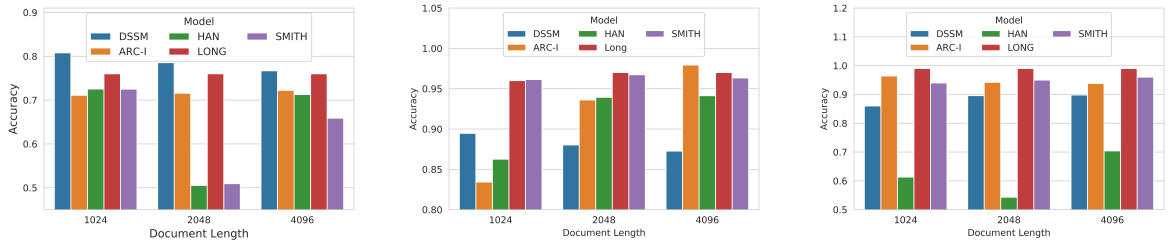
A.6 Robustness to Text Perturbation

We randomly shuffle the documents before training different models and measure their test accuracy on the original document matching task (Figure 7) for all three datasets. The first few paragraphs in Wikipedia articles, research papers, and patents are highly informative. We wanted to verify if the models give too much importance to the position of the initial text. In the context of long documents, just re-ordering the paragraphs of a document spanning pages should not have an effect on the downstream tasks of document matching. (Note: We use the term document matching broadly to refer to citation matching or document relevance. Given a document pair, we would like to verify if the two



(a) AAN (b) WIKI (c) PAT

Figure 5: Document Length vs Accuracy upto 512 tokens



(a) AAN (b) WIKI (c) PAT

Figure 6: Document Length vs Accuracy upto 4096 tokens

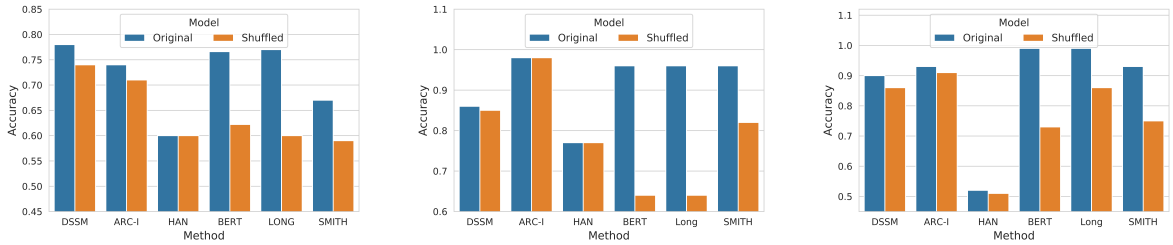
documents are relevant to each other.) In order to verify this assumption, we shuffle the paragraphs to distribute the important texts randomly and check the performance of all models on this downstream task. Although, we do observe a small drop after paragraph shuffling because the simple models do take into account a shallow context of the input text, the simple neural models overall prove more robust to text perturbation when compared to transformer-based models that take into account deep contextual information.

A.7 Best Input Embeddings for Simple Models

For simple models, we evaluate if the input vector representations play a role in the final results. We use the following input representations.

- **Tri-Gram Hashing (T)**: Bag-of-charTrigrams is a technique for word hashing (Huang et al., 2013) where each word is broken down into character trigrams (charTrigrams). Since, the number of possible charTrigarms are fixed and limited, this is a scalable solution for long documents. The charTrigrams are obtained for every token in the input text after appending the symbol '#' before and after every token. For example, the word 'good' [#good#] is split into [#go, goo, ood, od#] and then mapped to a 30,621 dimensional hash table. This vector representation for the document is then given as input to the models. For DSSM, each document is represented as a bag-of-charTrigrams and given as input to the model. For ARC-I and HAN, we split each document into n chunks which are represented in the form of a trigram hash. We construct a matrix of size $n \times$ trigram hash for the entire document which is given input to ARC-I and HAN.

- **GloVe Embeddings (G)**: GloVe (Pennington et al., 2014) is an unsupervised learning algorithm for obtaining vector representations for words. We download the pre-trained GloVe embeddings and get the vector representations for words in a long document. These vector representations are given as input to different models. For DSSM, we divide the document into chunks of a specified maximum length. We then take GloVe embedding representation of tokens for each chunk upto a maximum length and average them to get a document representation. For ARC-I and HAN, each document is represented as a matrix of size [embedding dimension $\times$ max length] in each document.



(a) AAN (b) WIKI (c) PAT

Figure 7: Original vs Shuffled Documents

	AAN				WIKI				PAT			
Model	**P**	**R**	**F1**	**Acc**	**P**	**R**	**F1**	**Acc**	**P**	**R**	**F1**	**Acc**
DSSM-T	0.768	0.809	0.787	0.780	0.823	0.939	0.877	0.869	0.869	0.957	0.911	0.905
ARC-I-T	0.641	0.606	0.622	0.634	0.969	0.944	0.956	0.957	0.536	0.754	0.626	0.793
HAN-T	0.665	0.885	0.759	0.720	0.911	0.929	0.920	0.920	0.477	0.857	0.618	0.751
DSSM-G	0.550	0.541	0.545	0.549	0.966	**0.986**	0.975	0.975	**0.992**	**0.998**	**0.995**	**0.995**
ARC-I-G	0.643	0.872	0.734	0.676	**0.992**	0.983	**0.987**	**0.987**	0.905	0.963	0.933	0.929
HAN-G	0.504	0.881	0.641	0.507	0.935	0.984	0.959	0.958	0.609	0.848	0.709	0.522
DSSM-D	**0.852**	0.763	**0.805**	**0.815**	0.933	0.984	0.958	0.957	0.841	0.959	0.896	0.949
ARC-I-D	0.841	0.763	0.800	0.809	0.987	0.985	0.986	0.986	0.967	0.958	0.962	0.983
HAN-D	0.709	**0.919**	0.801	0.771	0.875	0.859	0.866	0.873	0.946	0.996	0.970	0.975

Table 4: Comparison of different input aggregation techniques: (i) charTrigrams (T), (ii) GloVe Embeddings (G), and (iii) Doc2Vec embeddings (D), for simple models.

	AAN				WIKI				PAT			
Model	**P**	**R**	**F1**	**Acc**	**P**	**R**	**F1**	**Acc**	**P**	**R**	**F1**	**Acc**
BERT-CLS	0.992	0.579	0.732	0.637	0.998	0.499	0.666	0.499	1	0.639	0.783	0.714
BERT-POOL	0.572	0.992	0.726	0.625	1	0.500	0.667	0.501	1	0.637	0.778	0.711
LONG-CLS	0.599	0.842	0.699	0.743	0.968	0.764	0.854	0.835	0.927	0.911	0.919	0.917
LONG-POOL	0.727	0.819	0.770	0.783	0.994	0.812	0.894	0.883	0.996	0.918	0.955	0.953

Table 5: Aggregation using the [CLS] token, and the pooler output [POOL] from BERT and Longformer for documents > 512 and > 4096 tokens for BERT and Longformer, respectively. The results were the same for SUM and AVG aggregation techniques.

- **Doc2Vec Embeddings (D)**: Doc2Vec (Le and Mikolov, 2014) embeddings can be used to get vector representations for a document. We train Doc2Vec models from scratch on different datasets to get relevant document representations. These document representations are then given as input to different document matching models.

Table 4 shows the model performance of the simple models for the above three input representations. We observe that using GloVe (G) and Doc2Vec (D) input embeddings improve the model performance of the simple models overall.

A.8 Different Aggregation Techniques for Transformer Based Models

We experiment with different aggregation techniques (SUM and AVG) for $>$ 512 tokens for BERT, and $>$ 4096 tokens for Longformer. We chunk the documents and aggregate the representations from the '[CLS]' token and 'the pooler output'. The results can be seen in Table 5 and were the same for SUM and AVG aggregation techniques. Aggregation resulted in an overall performance drop when compared to just truncating the documents up to 512 tokens and 4096 tokens for BERT and Longformers, respectively for the document matching.