The Devil is in the Details: Evaluating Limitations of Transformer-based Methods for Granular Tasks

Brihi Joshi[†] **Neil Shah**[♣] **Francesco Barbieri**[♣] **Leonardo Neves**[♣]
[†]Indraprastha Institute of Information Technology Delhi, New Delhi, India
[♣]Snap Inc., Santa Monica, CA 90405, USA
[†]brihi16142@iiitd.ac.in,
[♣]{nshah,fbarbieri,lneves}@snap.com,

Abstract

Contextual embeddings derived from transformer-based neural language models have shown state-of-the-art performance for various tasks such as question answering, sentiment analysis, and textual similarity in recent years. Extensive work shows how accurately such models can represent *abstract*, semantic information present in text. In this expository work, we explore a tangent direction and analyze such models' performance on tasks that require a more *granular* level of representation. We focus on the problem of textual similarity from two perspectives: matching documents on a granular level (requiring embeddings to capture fine-grained attributes in the text), and an abstract level (requiring embeddings to capture overall textual semantics). We empirically demonstrate, across two datasets from different domains, that despite high performance in abstract document matching as expected, contextual embeddings are consistently (and at times, vastly) outperformed by simple baselines like TF-IDF for more granular tasks. We then propose a simple but effective method to incorporate TF-IDF into models that use contextual embeddings, achieving relative improvements of up to 36% on granular tasks.

1 Introduction

In recent years, contextual embeddings (Peters et al., 2018; Devlin et al., 2018) have made immense progress in semantic understanding-based tasks. After being trained using large amounts of data, for example via a self-supervised task like masked language-modeling, such models learn crucial elements of language, such as syntax and semantics (Jawahar et al., 2019; Goldberg, 2019; Wiedemann et al., 2019) from just raw text. The best performing contextual embeddings are trained with Transformer-based methods (TBM) (Vaswani et al., 2017; Devlin et al., 2018). These embeddings have been shown to frequently achieve state-of-the-art results in downstream tasks like question answering and sentiment analysis (van Aken et al., 2019; Sun et al., 2019). Contextual embeddings are also often used to capture the similarity between pairs of documents; for example, on the Semantic Textual Similarity (STS) task (Cer et al., 2017) included in the GLUE benchmark (Wang et al., 2018), TBMs have shown competitive performance, substantially outperforming embedding baselines like Word2Vec (Mikolov et al., 2013) and GloVE (Pennington et al., 2014). However, their performance on similarity tasks beyond abstract, semantic ones (Mickus et al., 2019) – for example, on granular news article matching – is less understood.

In this work, we study the performance of TBMs in textual similarity tasks with the following research question: *Are transformer-based methods as performant for granular tasks as they are for abstract ones?* Here, *granular* and *abstract* reflect varying amounts of coarseness in the concept of *similarity*. For example, consider the news domain: A granular notion of similarity might be whether a pair of articles both report the exact same news event. Conversely, an abstract notion might be when the articles share the same topical category, like sports or finance. Figure 1 illustrates this with an example for clarity.

Firstly, we define separate tasks to explore these two notions of similarity on two datasets from different domains – News Articles, and Bug Reports. Our analysis on both datasets reveals that contextual

Proceedings of the 28th International Conference on Computational Linguistics, pages 3652–3659
Barcelona, Spain (Online), December 8-13, 2020

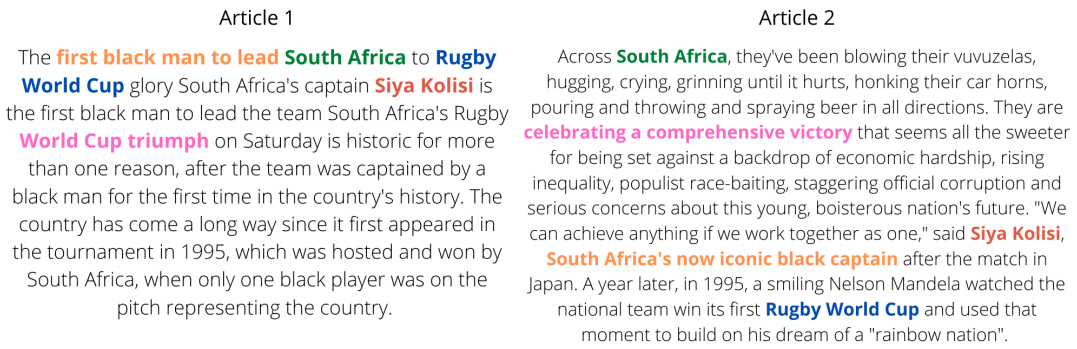

Figure 1: An example pair of articles from the **News Dedup** dataset: Both report the same news event, and are thus *similar on a granular level*; the colored text indicates fine-grained details associated with this determination. Both articles are also of the "sports" topic, and are thus *similar on an abstract level*.

embeddings *do not* perform well on granular tasks, and are outperformed by simple baselines like TF-IDF. Secondly, we demonstrate that TBM contextual embeddings *do* in fact contain important semantic information, and a simple interpolation strategy between the two methods can help boost the relative individual performance of TBMs (TF-IDF) by up to 36% (6%) on the granular task.

2 Related Work

We discuss related work in two areas: textual similarity, and TBMs.

Textual Similarity has been studied from various perspectives – comparing documents of different lengths in order to capture varying levels of detail (Gong et al., 2018), evaluating semantic similarity between reference and generated corpus (Clark et al., 2019), and semantic similarity for long documents in a hierarchical fashion (Jiang et al., 2019). It is also shown that sentence meta-embeddings (obtained from combining ensembles of sentence embeddings) perform better (Poerner et al., 2019) for semantic similarity tasks compared to the individual baselines. For duplicate detection, which is a more granular task compared to semantic similarity, Rodier and Carter (2020) show that detection of near-duplicates in news articles can be identified by evaluating n-gram level overlap in documents. In the news domain, Liu et al. (2018) shows that article similarity can be improved by extracting common 'concepts' from the two articles using graph-based approaches.

TBMs (Liu et al., 2019; Devlin et al., 2018) have been shown to consistently perform better in the semantic similarity tasks. Peinelt et al. (2020) also shows that BERT-based architectures appended with topic-related details from topic models lead to an increase in semantic similarity performance. However, few works have highlighted TBMs' ability to capture granular information. (Khattab and Zaharia, 2020) shows that BERT can be used for document retrieval by matching embeddings of each word in the query and document, capturing granular similarity.

Unlike previous approaches that focus on either a granular or abstract similarity task, we compare the performance of TBMs with other baseline methods across the two tasks, and in addition, provide a simple method to improve the performance of TBMs on granular similarity tasks.

3 Method

In this section, we describe the methodology used to compare two documents from a granular and abstract perspective. Further, we also define the granular and abstract text similarity tasks in detail.

3.1 Problem Definition

We consider both granular and abstract tasks to be similarity classification tasks operating on a pair of documents. The task-specific labels are binary, indicating whether the pair is judged to be similar or not (one label for abstract, one for granular). From a corpus $\mathcal{C}$, we consider a pair of two documents d_1, d_2 and their task-specific similarity judgment y (without loss of generality). We define $e_k = f(d_k)$, where

e_k is d_k's embedding, produced by $f(\cdot)$. In practice, f could be a vector space method like TF-IDF, or the final layer from a TBM. Upon obtaining e_1, e_2, we generate a (symmetric) pair similarity score $g(e_1, e_2)$ corresponding to the given task, and use it to arrive at a binary prediction $\hat{y}$. Performance is measured using standard metrics quantifying agreement between $\hat{y}$ and y across pairs.

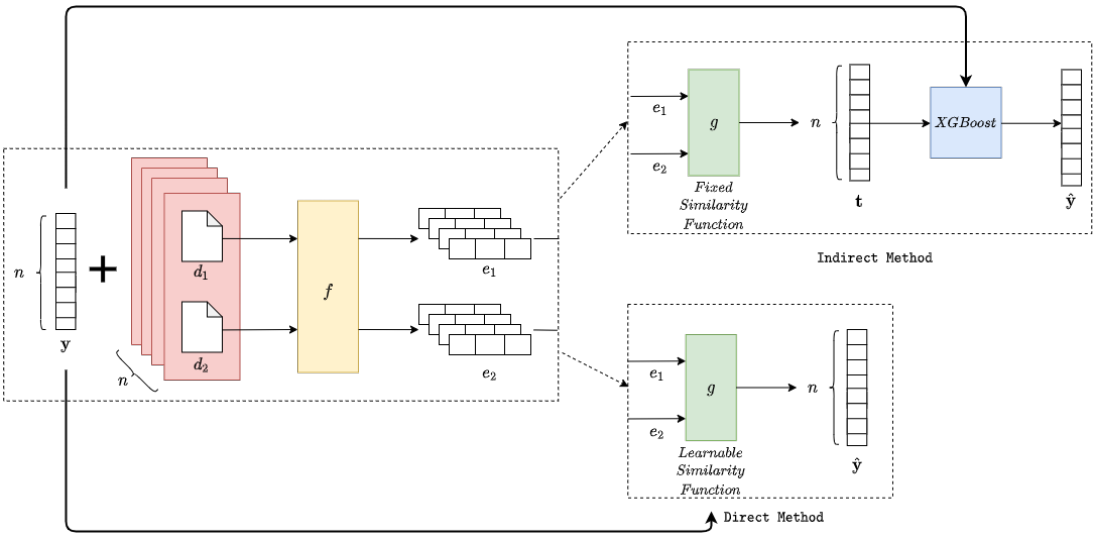

Figure 2: Our experimentation setups take n pairs consisting of two documents (d_1, d_2) and their similarity label (y) and yields their similarity score ($\hat{y}$)

3.2 Experimental Setup

We consider two experimental settings: *indirect* and *direct*. These are also illustrated in Figure 2[1]. In the *indirect* setting, we indirectly learn to predict $\mathbf{y}$ via a fixed g. Specifically, we a priori define g as the cosine similarity function, and define a vector $\mathbf{t}$, with each entry corresponding to $g(e_1, e_2)$ for a given pair of document embeddings w.l.o.g. We then feed $\mathbf{t}$ as a feature and the task-specific label vector $\mathbf{y}$ to XGBoost (Chen and Guestrin, 2016) to obtain the predictions $\hat{\mathbf{y}}$. We evaluate using several embedding functions:

- **TF-IDF:** TF-IDF (Ramos and others, 2003) weights corresponding to the 1-gram tokens inserted in their respective indices in an array of same length as the train's set vocabulary.
- **WME:** Word Mover's Embedding (WME) (Wu et al., 2018) generated from static embeddings like word2vec (Mikolov et al., 2013) using the Word Mover's Distance metric (Kusner et al., 2015).
- **SIF:** The SIF(Arora et al., 2017) weighting scheme employed over pretrained GloVE embeddings.
- **RT:** The embedding corresponding to the `CLS` token in the final layer of a pretrained RoBERTa (Liu et al., 2019) model.
- **LF:** The embedding corresponding to the `CLS` token in the final layer of a pretrained LongFormer (Beltagy et al., 2020) model.
- **ST-RT:** Sentence embeddings generated using Sentence Transformers (Reimers and Gurevych, 2019), which is a RoBERTa model fine-tuned on the STS benchmark.

In the *direct* setting, we directly learn to predict $\mathbf{y}$ from embeddings e_1, e_2 in an end-to-end manner, rather than through a predefined similarity measure. We use the best performing embeddings from the previous setting as input to train the model g which produces a score, which is thresholded to derive $\hat{\mathbf{y}}$.

- **TF-IDF-E2E:** We compute the absolute difference between the TF-IDF vectors of the article pairs and train a Logistic Regression classifier with the labels corresponding to the similarity task.
- **RT-E2E:** Since ST-RT embedding uses a pre-trained RoBERTa model, we train RoBERTa end-to-end instead (as Reimers and Gurevych (2019) mentions, ST is not intended for end-to-end use). We

[1]The code for our experiments is available at https://github.com/brihijoshi/granular-similarity-COLING-2020

	Whole dataset			Granular		Abstract	
	Avg # words	Train	Test	Train	Test	Train	Test
ND	388.6	3105	695	1763/1342	53/642	2452/653	509/186
BR	5.7	72142	8220	40967/31175	3954/4266	47199/24943	4804/3416

Table 1: Dataset and evaluation split details: The dataset statistics capture number of unique *text pairs*. For the task-specific statistics, we report the total number of similar pairs and non-similar (similar/not-similar) pairs according to the task for each split. The imbalance in the test set of **ND** replicates the distribution found in the real-world news event similarity detection problems.

		Indirect (Cosine Sim.)						Direct (End-to-end)	
		TF-IDF	WMD	SIF	RT	LF	ST-RT	TF-IDF-E2E	RT-E2E
Granular	**ND**	**0.85**	0.72	0.54	0.59	0.62	0.66	0.68	0.59
	BR	**0.75**	0.62	0.43	0.66	0.69	0.71	0.71	0.70
Abstract	**ND**	0.54	0.57	0.51	0.59	0.62	0.62	0.58	**0.66**
	BR	0.69	0.47	0.40	0.67	0.70	0.73	0.51	**0.74**

Table 2: Granular and abstract similarity results: TF-IDF outperforms TBMs on granular tasks, while TBMs outperform on abstract tasks in both settings.

provide the article pairs to a pre-trained RoBERTa model, separated with the `SEP` token. It is then directly fine-tuned on the task-specific labels.

3.3 Datasets

We evaluate with datasets from News Articles and Bug Reports domains to demonstrate generality. Each includes both abstract and granular labels for the same documents.

News Dedup dataset (ND) contains pairs of news articles from 243 different English news sources, collected between September and November 2019 from RSS feeds. For each pair, we assign a granular binary label indicating whether the pair reports the same news event, and an abstract binary label reflecting whether they share the same topic (politics, business, technology, entertainment, sports, science, or other – adapted from Google News[2]). We source annotations from Amazon Mechanical Turk[3], relying on multiple annotator agreement, with Fleiss' κ coefficient of 0.68. See Appendix A for details.

Bug Repo dataset (BR) (Lamkanfi et al., 2013) contains bug reports from several open-source projects like Eclipse and Mozilla, and is used primarily for duplicate bug detection. Each report consists of a title, a description of the error, the broad category that the bug belongs to (e.g. UI or Scripting out of 21 others), and a set of duplicate reports that flag the same bug. We indicate granular similarity as those pairs which flag the same bug, and abstract similarity as those pairs which fall under the same category, with the title of the report as the textual input. For each dataset, the documents in the train and test splits are disjoint sets, ensuring that the model does not memorize textual representations. For ND, the sets are also temporally disjoint, avoiding event overlap between train and test splits. Further details about the splits are provided in Table 1.

4 Results

Table 2 summarizes the experiments on the two datasets using the methods mentioned in Section 3. We can observe that a simple TF-IDF based approach performs better than all embedding methods for the granular-level similarity tasks. However, for the abstract-level similarity task, training a RoBERTa model to perform the task end-to-end achieves, as expected, the highest performance.

Despite the better results, the complete absence of semantic understanding is a disadvantage of TF-IDF. To mitigate this issue, we propose a simple approach to merge the best performing indirect methods.

[2]https://news.google.com/

[3]https://www.mturk.com/

		Values of weight w						
		0	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**	**1**
Granular	**ND**	0.66	0.77	0.83	0.89	**0.90**	0.86	0.85
	BR	0.71	0.56	0.62	0.69	**0.79**	0.76	0.75
Abstract	**ND**	**0.62**	0.60	0.55	0.56	0.60	0.57	0.54
	BR	**0.73**	0.72	0.72	0.70	0.69	0.68	0.69

Table 3: Performance for granular and abstract tasks the 2 datasets as we vary the value of w. Note that best granular results are achieved by interpolating TF-IDF with TBM predictions ($w = 0.7$).

Let g_t and g_r be the similarity scores obtained from the TF-IDF and the ST-RT approaches respectively: we obtain a new, interpolated score $g_i = w \cdot g_t + (1-w) \cdot g_r$, that is then used as an input to the classifier. As Table 3 shows, performance drastically changes when varying w. For both datasets, we observe the best results when $w = 0.7$, demonstrating that combining semantic and fine-grained information is helpful for granular tasks. [4] Conversely, we achieve the best performance on the abstract level when using only ST-RT. We hypothesize the noise introduced by the granular information results in the performance drop in cases prioritizing abstract, semantic relevance.

5 Conclusion

In this work, we study the use of contextual embeddings derived from transformer-based models (TBMs) for semantic similarity tasks of varying granularity level. Through empirical analysis, we show that while TBMs achieve higher performance in the abstract similarity tasks, simple methods like TF-IDF outperform these models for granular similarity tasks (like event matching). We then propose a simple but effective method to merge these two approaches, achieving relative improvements of 36% (6%) when compared to using only TBMs (TF-IDF). In future work, we plan to investigate the scope for integrating granular information into TBM contextual embeddings to toggle the granularity that such embeddings inherently encode.

References

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

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

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August. Association for Computational Linguistics.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug.

Elizabeth Clark, Asli Celikyilmaz, and Noah A. Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2748–2760, Florence, Italy, July. Association for Computational Linguistics.

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

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Hongyu Gong, Tarek Sakakini, Suma Bhat, and JinJun Xiong. 2018. Document similarity for texts of varying lengths via hidden topics. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2341–2351, Melbourne, Australia, July. Association for Computational Linguistics.

[4] Even though w is robust to the two datasets that we are using, we would recommend tuning it for other datasets.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. 2017. news-please: A generic news crawler and extractor. In Maria Gaede, Violeta Trkulja, and Vivien Petra, editors, *Proceedings of the 15th International Symposium of Information Science*, pages 218–223, March.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

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*, WWW '19, page 795–806, New York, NY, USA. Association for Computing Machinery.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 957–966. JMLR.org.

A. Lamkanfi, J. Pérez, and S. Demeyer. 2013. The eclipse and mozilla defect tracking dataset: A genuine dataset for mining bug information. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 203–206.

Bang Liu, Di Niu, Haojie Wei, Jinghong Lin, Yancheng He, Kunfeng Lai, and Yu Xu. 2018. Matching article pairs with graphical decomposition and convolutions.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Timothee Mickus, Denis Paperno, Mathieu Constant, and Kees van Deemter. 2019. What do you mean, bert? assessing bert as a distributional semantics model.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain, July. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

Nicole Peinelt, Dong Nguyen, and Maria Liakata. 2020. tBERT: Topic models and BERT joining forces for semantic similarity detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7047–7055, Online, July. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June. Association for Computational Linguistics.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2019. Sentence meta-embeddings for unsupervised semantic textual similarity.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. New Jersey, USA.

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*. Association for Computational Linguistics, 11.

Simon Rodier and Dave Carter. 2020. Online near-duplicate detection of news articles. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 1242–1249, Marseille, France, May. European Language Resources Association.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. 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 380–385, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Betty van Aken, Benjamin Winter, Alexander Löser, and Felix A. Gers. 2019. How does bert answer questions? a layer-wise analysis of transformer representations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1823–1832, New York, NY, USA. Association for Computing Machinery.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding.

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4524–4534, Brussels, Belgium, October-November. Association for Computational Linguistics.

Appendix A - News Dedup (ND) dataset details

The News Dedup dataset was collected from a set of 243 English news sources, over a span of 3 months (September 2019 - November 2019). Further details about the dataset are as follows -

1. **Collection:** With a list of pre-prepared news sources, we extract the links to the RSS feeds of these sources. We then use the News-please package (Hamborg et al., 2017) to extract details like the title, body, attached images, etc from the updated articles from the RSS feeds. This process is scheduled ever 3 hours, so that new articles are immediately scraped. Given that most of the news sources report articles belonging to the topic 'Politics', we attempted to ensure articles from diverse topics to be equally represented during annotation.

2. **Postprocessing:** In order to create relevant article pairs from the set of scraped news articles, we first extract keywords using TextRank (Mihalcea and Tarau, 2004). We then extract embeddings for the keywords using Word2vec and average the embeddings to obtain an embedding for the entire article. We then use cosine similarity between all possible embeddings in the corpus. This similarity score is used as a 'proxy' to extract relevant pairs for annotation. We bin our obtained pairs into 3 categories – positives, easy negatives and hard negatives. Easy negatives are pairs with a similarity score less than a certain threshold (upon observation, this threshold seemed to be ranging from 20-30%). They are however, verified during annotation process. Hard negatives are pairs with high similarity, but do not satisfy our criterion of similarity (this happens mostly for the granular task). Before the annotation, we ensured that the dataset did not contain transitive pairs, i.e, if articles A and B and articles B and C are present as pairs, we ensured that articles A and C are not present in the dataset while it is being annotated.

3. **Annotation:** During the annotation process, each article pair is labelled by 3 annotators. While constructing the task on Amazon Mechanical Task, it is ensured that the task contains *golden questions* – a set of common sense questions based on current affairs, so that the annotators are judged on their capability to read lengthy news articles. The annotators had substantial agreement for the Granular simlilarity task (with Fleiss' kappa coefficient of 0.68).