Finding most similar movie characters using BERT

ABSTRACT

An optimal identification of closest neighbors (paragraph-level text that are most similar to one another) using the Bidirectional Encoder Representations from Transformers (BERT) Next Sentence Prediction model requires an exhaustive pairwise comparison between each text. However, this is an $O(n^2)$ operation with respect to the number of paragraph-level text, which becomes unfeasible for a corpus with a large number of paragraph-level text. This is because each pairwise comparison takes an extremely long time even on GPU computation. Practically, only a tiny fraction of all pairwise comparisons can be performed. Identifying which pairwise comparisons are to be performed is therefore a critical step in finding closest neighbors. We propose a simple but effective method that greatly outperforms state-of-the-art methods to find closest neighbors. In the task of identifying which paragraph-level text comes from the same category (29871 text and 6229 categories), our method performs at least 150% better than the best state-of-the-art paragraph-level embedding baseline. This suggests that our method can greatly increase the effectiveness of using the BERT Next Sentence Prediction model to find closest neighbors.

KEYWORDS

Natural Language Processing, Information Retrieval, Narrative Understanding, Online Forum, Persona Modelling, Extreme Classification

1 INTRODUCTION

Most methods of finding closest neighbors (sentence-level, paragraph-level or document-level text that are most similar to one another) involve the calculation of a similarity metric between the embedding of two pieces of text [15]. An effective and popular method is to calculate the cosine similarity between two pieces of text [16, 18]. The invention of the BERT Next Sentence Prediction model architecture [8] brings new promises to calculating an alternative metric of similarity that is not solely based on the embedding of two pieces of text. This resulted in a more accurate measure of similarity between the two text [8].

However, a core limitation of this method is its high computation overhead, preventing its widespread use, A single BERT Next Sentence Prediction operation has extremely high runtime demands due to the number of calculations it has to perform given the high number of perimeters within the model (110 million in the base model). Furthermore, an optimal identification of closest neighbors requires the operation to be performed on every pair of text, a $O(n^2)$ procedure with respect to the number of paragraph-level text. This makes the optimal identification unfeasible on datasets comprising of more than a few thousand text. To overcome this limitation, we invented an easy but effective method of identifying a tiny fraction ($< 2\%$) of text-pairs on which BERT Next Sentence Prediction (NSP) has to be performed while maintaining good performance. Our method does so by finding text-pairs whose BERT Masked Language Modelling average embedding is most cosine similar to each other.

To evaluate the effectiveness of our method of identifying text-pairs, we had to choose an appropriate task. A critical requirement of the task is that it has ground-truth labels that proxies for closest neighbors. Furthermore, each text must not have too many closest neighbors in the ground truth labels to make the identification task difficult enough to distinguish between effective and ineffective methods. Preference was also given to tasks that utilise language in an informal context. Compared to tasks that use language in a formal context, these tasks have more restricted vocabulary due to fewer rare/domain-specific terms. The restricted vocabulary of these tasks can better take advantage of the BERT model's ability to provide contextual word embedding and represent the meaning of a word in the context of other words in the paragraph.

Superman's 1990s enemy Conduit. This is also a facet of both the Master Jailer's hatred of Superman and that of Lex Luthor himself. In Lex's case, he hates Superman because he's the only person alive who is simply better than him, and he knows if Superman wasn't around he would be humanity's greatest hero instead. In Blackest Night, Wonder Woman used her lasso to make Lex admit that he wanted to be Superman. **Loki** One of the reasons Loki's relationship with his brother Thor was so conflicted was because Loki envied Thor for being Odin's favored son. Of course, Loki's constant scheming against Thor in his efforts to one-up him gave Odin and the rest of Asgard more and more reasons to hate Loki, to the point that now every Asgardian except Thor wants to kill Loki even after he was reincarnated as a relatively innocent and powerless child with no memories of his past self.

Table 1: Character descriptions from the trope "Driven by Envy"

We created a task using highly concise character descriptions on various media (novels/films/anime) from All The Tropes[1]. We created Ground-Truth positive labels of closest neighbors based on whether these character descriptions share a common theme (also known as a trope), with an example from the trope "Driven by Envy" shown in Table 1. Other themes (tropes) include "Parental Neglect", "Fallen Hero", and "A Friend in Need". Not only is the task suitable for testing our method of identifying text-pairs, but we also found the task intrinsically motivating. This is because many of these character descriptions are inspired by and related to true stories of people so understanding how to identify similarities between character descriptions might ultimately help us to better understand similarities in human characteristics and experiences as well.

The contributions of this paper can be summarized as follows:

[1] https://allthetropes.org

(1) The discovery of a simple but effective method of finding a tiny proportion of text-pairs on which to perform BERT Next Sentence Prediction, while maintaining good performance.
(2) Doing so makes it feasible to use BERT Next Sentence Prediction to find paragraph-level text that is most similar to one another in a corpus of more than thousands of text (when it is previously not).
(3) To evaluate the method and compare it with other existing models, we create a novel task involving finding character descriptions that share a common theme (also known as a trope).

2 RELATED WORK

2.1 Embedding-based Extreme Classification

Extreme classification refers to techniques used in classification tasks with many classes, ranging from hundreds to millions. Because many extreme classification tasks[2] deal with textual information, researchers have found it useful to transform feature vectors from high-dimensional sparse Bag-of-Words models into low-dimensional dense embedding. This is often achieved by adapting the methodology described in [16, 18]. Most recent research in extreme classification [4, 9, 14, 19, 21–24] focuses on finding the relationship between class labels and low-dimensional dense embedding.

However, the use of such low dimensional dense embedding might no longer be the most appropriate due to advances in the field of natural language processing. First, there has been a rise in the use of contextual word vectors [8, 12, 20]. Contextual word vectors refer to word vector embedding that changes based on the context a word is used in rather than remaining the same across different context as with Word2Vec [18]. This is important to take into account because many words such as "play" are polysemous, which means that they have distinctly different meanings in different settings ("play a game" versus "watch a play"). Second, there is an ascent in the use of transfer learning [8, 12, 20]. Under this paradigm, a significantly larger external corpus (e.g. Wikipedia) can be first used to pre-train a language model to capture semantic and syntactic information that is common to both the external corpus and the extreme classification text corpus. Thereafter, the language model can be fine-tuned on the smaller extreme classification text corpus. This overcomes a key bottleneck of an inadequately trained language model when the extreme classification corpus is greatly limited. Third, the relationship between feature embedding is generally ignored [9, 14, 19, 21–23] or used shallowly [4, 24]. By used shallowly, we mean that the relationship between a text-pair is calculated through a simple distance metric (Cosine/Euclidean/-Manhattan distances) between the text embedding of a text-pair instead of being re-computed using the text-pair itself. This leads to complex, non-linear relationships between individual words in a text-pair to be often overlooked.

2.2 Analysis of characters in film and fiction

Characters in movies and novels have been computationally analyzed by many researchers. Bamman *et al.* [1, 2] attempted to cluster various characters into prototypes based on topic modelling techniques [5]. On the other hand, Frermann and Szarvas [11] and Iyyer *et al.* [13] sought to classify fictional characters alongside the relationships between them using recurrent neural networks and matrix factorization. The relationships between characters are also pertinent to our task because what characters share in many tropes is their involvement in a relationship of a similar nature. For instance, one theme "Parental Neglect" concerns a common experience of being uncared for by parents and guardians. However, the corpus we use differs significantly from those used in existing research. We use highly concise character descriptions of around 100 words whereas existing research mostly uses movie/book-length character mentions. This is useful for our task because the concise character description typically exemplifies a specific trait/experience of a character. This allows the differences between characters to be more discriminative compared to a longer description, which might include more points of commonality (going to school/work, eating and having a polite conversation).

2.3 Deductive coding of individual experiences

Because character descriptions are related to individual experiences, it is helpful to understand how such experiences can be analyzed. Mostly researched in the field of psychology, individual experiences are often analyzed through asking individuals to document and reflect upon their experiences. Trained analysts then seek to classify such writing into predefined categories. Demorest *et al.* [7] interpreted an individual's experience in the form of three key stages: an individual's wish, the response from the other and the response from the self in light of the response from the other. Each stage consists of around ten predefined categories such as wanting to be autonomous (Stage 1), being denied of that autonomy (Stage 2) and developing an enmity against the other (Stage 3). Thorne and McLean [25] organized their analysis in terms of central themes. These central themes include experiences of interpersonal turmoil, having a sense of achievement and surviving a potentially life-threatening event/illness. Such analysis can inform us why characters in the same trope are considered to be similar, allowing us to better interpret how our model discriminates whether characters are similar.

3 METHODS

In this section, we first discuss how we train a BERT Next Sentence Prediction model. Then, we share details of how our evaluation dataset was created and used for testing the feasibility of BERT Next Sentence Prediction model. Next, we explore how our BERT Next Sentence Prediction model was utilized to find the closest neighbors efficiently. Finally, we present a metric to evaluate the performance of our model.

3.1 Fine-tuning BERT Next Sentence Prediction model

We entirely re-trained a BERT Next Sentence Prediction model (English-base-uncased) with the pre-trained weights used as an initialization. Re-training the model was necessary because our task contains language used in an informal context while the original BERT Next Sentence Prediction model was trained on the language

[2] http://manikvarma.org/downloads/XC/XMLRepository.html

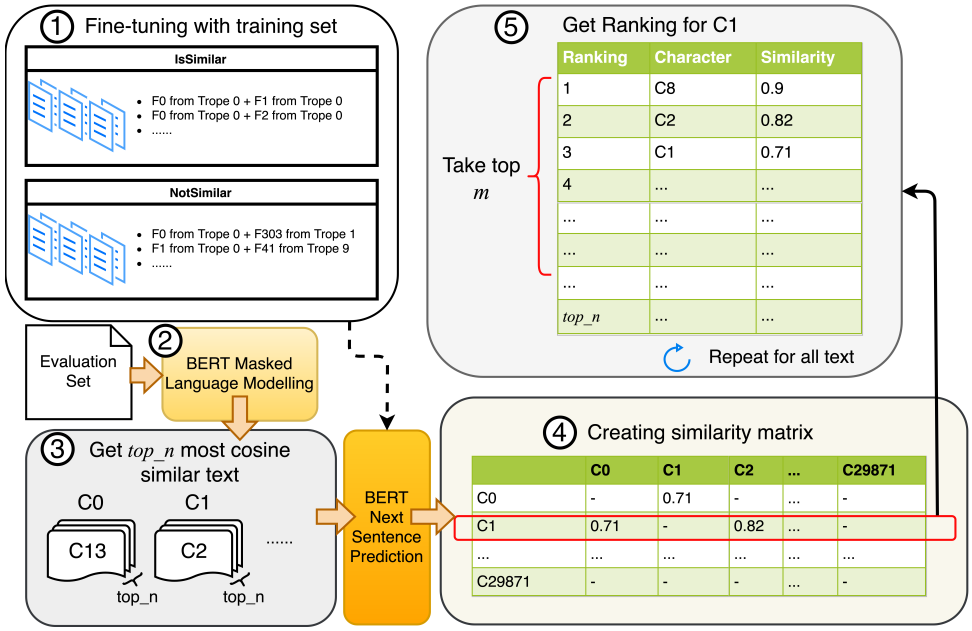


Figure 1: Workflow of finding closest neighbours: (1) Fine-tuning BERT Next Sentence Prediction using training dataset (Sec 3.1) (2) Obtain text embedding using BERT Masked Language Model (3) For each text, *top_n* most cosine similar text are found using text embedding. (4) BERT Next Sentence Prediction is utilized to create a similarity matrix. (5) Top m character descriptions are selected as closest neighbors (Sec 3.4)

used in a formal context (Wikipedia and Books). Furthermore, the model was originally pre-trained on a sentence-level task whereas our task is a paragraph-level one.

Our first step was to download character descriptions from All The Tropes[3]. We downloaded all character descriptions that had more than 100 words because character descriptions that are too short are unlikely to provide sufficient textual information for comparing similarity with other character descriptions. This amounted to around 300,000 character descriptions that are classified under 22,000 tropes (themes). We then filtered our data to retain only tropes that contain more than one character descriptions. Inspired by the methodology of [8], we generated all possible combination-pairs of character descriptions that are classified under each trope and gave the text-pair a label of "IsSimilar". For each "IsSimilar" pair, we took the first item and randomly selected a character description that is not in the same trope as the first item and gave the new pair a label of "NotSimilar". As a result, this gave us a balanced training set of 4 million text-pairs (approximately 8 GBs).

All hyper-parameters used to train the model were default[4] except adjusting the maximum sequence length to 512 tokens (to adapt to the paragraph-level text), batch-size per GPU to 8 and epoch number to 2, as recommended by [8]. We also used the default pre-trained BERT English-base-uncased tokenizer because only a small proportion of words in the training corpus were out-of-vocabulary. As a result, our training took 4 days on 4 Nvidia Tesla P100 GPUs.

[3]https://allthetropes.org
[4]https://github.com/huggingface/transformers/

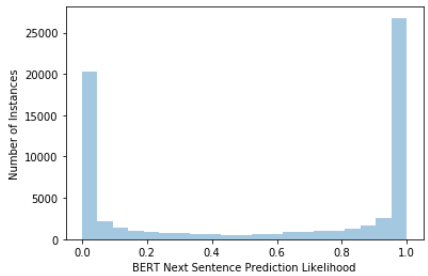
3.2 Evaluation dataset preparation

The evaluation dataset was prepared in a similar method as generating "IsSimilar" pairs described in Section 3.1, with the only difference being that we used character descriptions (text) with more than 90 but less than 100 words to prevent overlap with the training corpus. As a result, our evaluation dataset contains 29871 text (character descriptions) classified under 6229 tropes. Amongst them, there were 66780 text-pairs from the same trope, which were denoted as Pairs-From-Same-Trope.

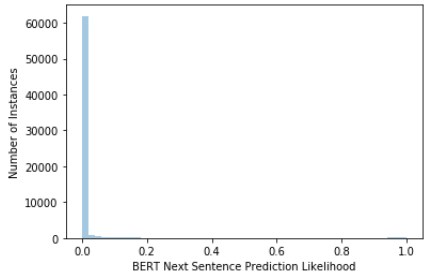
3.3 Feasibility study

Before conducting experiments to test if our model can distinguish relative similarity between text, it is critical to understand whether our retrained BERT Next Sentence Prediction model is able to infer absolute similarities between text-pairs. Therefore, we performed a feasibility study to investigate the distributions of BERT Next Sentence Prediction likelihood for Pairs-From-Same-Trope in our evaluation dataset and a corresponding Randomly-chosen Non-Pairs-From-Same-Trope. Randomly-chosen Non-Pairs-From-Same-Trope was obtained through a similar method as generating "NotSimilar" pairs described in Section 3.1).

Figure 2 demonstrates that Pairs-From-Same-Trope text-pairs have a significantly higher chance of being predicted a high likelihood (mean = 0.556) compared to Randomly-chosen Non-Pairs-From-Same-Trope text-pairs (mean = 0.0279). This provides us with the confidence that our retrained BERT Next Sentence Prediction model has the ability to predict whether two character descriptions are from the same trope.



(a)



(b)

Figure 2: 2a: BERT Next Sentence Prediction likelihood for Pairs-From-Same-Trope Text-pairs; 2b: BERT Next Sentence Prediction likelihood for Randomly-chosen Non-Pairs-From-Same-Trope Text-pairs

3.4 Finding closest neighbors

To address the key limitation of utilizing BERT Next Sentence Prediction model in practice - impractically long runtime of exhaustive pairwise comparison - we propose to use BERT Masked Language Modelling (thereafter BERT MLM) embedding to determine which text-pairs to perform BERT Next Sentence Prediction on. For each text, we choose *top_n* other text whose BERT MLM fine-tuned average embedding[5] are most cosine similar to its own embedding, forming *top_n* most similar text-pairs. This procedure is outlined in Algorithm 1.

BERT Next Sentence Prediction model will then only be performed on the *top_n* most similar text-pairs as shown in Algorithm 2a, reducing the number of operations for each text from the length of the text corpus (l_{text}) to only *top_n*. As a consequence, the runtime complexity of the overall operation is reduced from $O(l_{text}^2)$ to $O(top\_n * l_{text}) == O(l_{text})$, given *top_n* is a constant.

As shown in in Algorithm 2a, we generate a similarity matrix with shape (l_{text}, l_{text}) and initial value 0. With each text in the corpus, BERT Next Sentence Prediction likelihood is computed for every pair of most similar *top_n* text-pairs that are returned from Algorithm 1. This likelihood is then used as the value for $similarity\_matrix_{ij}$ and $similarity\_matrix_{ji}$, where i and j represents the indices of both text in a text-pair. The resultant matrix is then sorted along each row in Algorithm 2b and indices with the m highest likelihood are selected as the closest m indices for subsequent evaluation.

[5]Details of their preparation in Section 4.1

3.5 Evaluation metric

The evaluation metric we will use (equation 1) is recall @ m. This is obtained by calculating the percentage of Pairs-From-Same-Trope each model is able to predict. It does so by choosing m text that are most similar to each text in the evaluation dataset in order to make text-pairs. In this way, it examines how many among the $m * l_{text}$ text-pairs chosen belongs to Pairs-From-Same-Trope before dividing it by the total number of Pairs-From-Same-Trope.

$$Recall\ @\ m = \frac{\sum_{i=1}^{l_{text}} \sum_{j=1}^{m} \mathbb{1}_{text-pair\ in\ Pairs-From-Same-Trope}}{l_{Pairs-From-Same-Trope}} \times 100\% \quad (1)$$

where $l_{Pairs-From-Same-Trope}$ is the total number of Pairs-From-Same-Trope and m is the number of most cosine similar text to be selected in Algorithm 2b.

For our task, this metric is more appropriate than others which are more commonly used in extreme classification tasks [3] such as precision @ m. This is because we seek to optimize our prediction of whether two character descriptions are in the same trope rather than which specific trope a character description belongs to. Furthermore, having a trope in common is only one of the proxies for the similarity between character descriptions. Character descriptions may share something else in common that is not captured by their belonging to a common trope. Therefore, it is more appropriate to use recall because it does not directly penalize a model for identifying similarities in character descriptions that are not captured by their belonging to the same trope.

4 EVALUATION

In this section, we first present how we prepared several different baseline models including state-of-the-art paragraph-level embedding models, and then investigate the effects of varying *top_n* (the number of most cosine similar text to be compared to each text). Finally, we compare the performance of our algorithm to baseline models.

4.1 Baseline Methods

Baseline measurements were obtained for Google Universal Sentence Encoder-large [6], Word2Vec [18], and BERT Masked Language Model [8] and RoBERT-base embeddings [17]. For each type of embedding, the closest neighbors were obtained by finding other text that are most cosine similar.

Google Universal Sentence Encoder-large model [6] (thereafter **USE** unfine-tuned) on Tensorflow Hub was used to obtain a 512-dimensional vector representation of each character description. **Word2Vec** was implemented by using pre-trained 300-dimensional Word2Vec embedding[7]. Character descriptions were first tokenized, lemmatized and stripped of stop-words before being transformed into a Bag of Words corpus. Thereafter, the embedding for each character description was calculated by taking a simple mean of all word embedding. BERT Masked Language Modeling (thereafter **BERT**) average embedding of 768 dimensions were obtained by

[6]https://tfhub.dev/google/universal-sentence-encoder-large/3
[7]https://code.google.com/archive/p/word2vec/

Algorithm 1

```
BERT_MLM_embedding = BERT_Masked_Language_Modelling(text)        ▷ matrix of shape( l_text, 768 )
procedure most_cosine_similar(index, top_n)
    cosine_similarities = cosine_similarity(BERT_MLM_embedding[index], BERT_MLM_embedding)
    most_similar = sort(cosine_similarities)[-top_n:]            ▷ exclude index
    return most_similar
```

Algorithm 2

2a

```
procedure generate_similarity_matrix(text, top_n)
    similarity_matrix = matrix.zeros(len(text),len(text))
    for index in range(len(text)) do
        first_text = text[index]
        list_of_top_n = most_cosine_similar(index, top_n)
        for second_index in list_of_top_n do
            second_text = text[second_index]
            NSP_likelihood = BERT_Next_Sentence_Prediction(first_text, second_text)
            similarity_matrix[index][second_index] = NSP_likelihood
            similarity_matrix[second_index][index] = NSP_likelihood
    return similarity_matrix
```

2b

```
procedure generate_closest_m_from_matrix(similarity_matrix, m)
    closest_m = {}
    length_matrix = similarity_matrix.shape[0]
    for index in range(length_matrix) do
        similarity_to_index = argsort(similarity_matrix[index])
        closest_m[index] = similarity_to_index[-m:]
    return closest_m
```

average-pooling all the word embedding of tokens in the second-to-last layer, as recommended by [26], CLS embedding, also of 768 dimensions were obtained by extracting the embedding of the CLS token in the second to last layer. The unfine-tuned variant refers to the pre-trained model whereas the fine-tuned variant refers to our retrained model. In all cases above, the English-base-uncased version was used. **RoBERTa** embedding of 768 dimensions was extracted using PyTorch Hub [10]. Text embedding for each character description was created by averaging all word embedding in the last layer, as recommended as a method to extract features for downstream tasks. The version used was the base model.

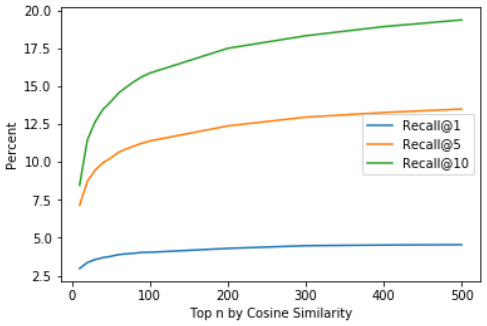


Figure 3: Recall @m of various *top_n*

4.2 Effects of varying *top_n*

Figure 3 demonstrates how Recall @ m varies with *top_n* by cosine similarity. The rate of increase in Figure 3 tapers off gradually with the minimal relative increase in all three Recall @ m after *top_n* = 100. This difference is most overt in Figure 4, which demonstrates the percent change per additional n. This metric is calculated by finding the difference in Recall @ m divided by the difference in number of *top_n* considered.

$$\textit{Percent change per additional top\_n}|_{\textit{Recall @ m}} = \frac{\textit{Recall @ m}|_{top\_n=k} - \textit{Recall @ m}|_{top\_n=k-s}}{s} \quad (2)$$

Where k is the value of *top_n* (ranging from 10 to 500) and s is the step. This metric is effectively a stepped first-derivative of Recall @ m with respect to increasing *top_n*.

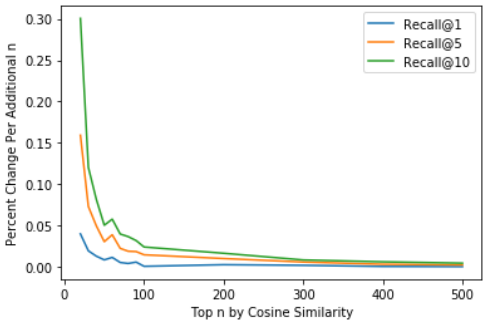


Figure 4: Percent change per additional *top_n* for Recall @m of various *top_n* by Cosine Similarity

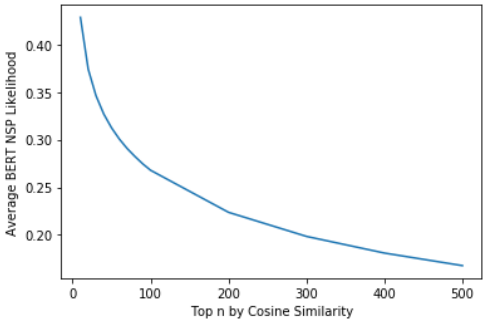


Figure 5: Average BERT Next Sentence Prediction likelihood for various *top_n* by Cosine Similarity

Interestingly, this exponential decline in Figure 4 is also observed in Figure 5, a plot of the average BERT Next Sentence Prediction likelihood with varying *top_n*. This indicates that as *top_n* increases, subsequent text-pairs are less likely to be considered "IsSimilar". Such a trend is correlated with whether these text-pairs are indeed in the Pairs-From-Same-Trope, as per the decreasing trend in Figure 4. Such observation provides additional support to Section 3.3 that our retrained BERT Next Sentence Prediction model can distinguish between text-pairs that are in Pairs-From-Same-Trope and those that are not.

Overall, this suggests that *top_n* = 100 is sufficient for this task since increasing *top_n* beyond 100 requires a linear increase in additional computational resources without significant improvements in performance. However, other tasks with different numbers and types of text might have different optimal numbers of *top_n*. Researchers who wish to adopt this methodology to a significantly different task might wish to conduct similar experiments to find out the optimal *top_n* for their task.

4.3 Comparison with baseline models

As shown in Table 2, our method performs significantly better at the task than all baseline measures at all Recall @ m. At *top_n* = 500, our algorithm performs at least 250% as well as the best performing baseline measure and at least 300% as well as BERT fine-tuned (both average and CLS), which the algorithm builds on.

	Recall @1	Recall @5	Recall @10
Our Models			
top_n = 500	**4.543**	**13.51**	**19.40**
top_n = 100	4.040	11.40	15.88
Baseline Models			
BERT fine-tuned average	0.8730	3.968	6.494
BERT fine-tuned CLS	0.8176	3.488	5.572
BERT unfine-tuned average	1.062	3.342	5.235
BERT unfine-tuned CLS	0.3369	1.072	1.662
USE unfine-tuned	1.415	4.663	7.300
RoBERTa average	1.505	4.723	7.050
Word2Vec average	1.201	3.780	5.764

Table 2: Recall @ *m* (in %) of different models

This is because BERT Next Sentence Prediction is a more accurate technique for determining the similarity between text compared to taking a simple cosine similarity between output embedding that all baselines employ. Our method helps to identify the most likely candidate text-pairs, which can then be verified by BERT Next Sentence Prediction to determine whether it is most likely to belong to Pairs-From-Same-Trope. Among the baselines, USE unfine-tuned and RoBERTa average perform best potentially due to their pairwise-text pre-training [6] and extensive pre-training [17] respectively.

5 CONCLUSION

We have presented an algorithm to identify closest-neighbors (paragraph-level text that is most similar to one another) using the BERT Next Sentence Prediction model without requiring an exhaustive pairwise comparison. This reduces the runtime complexity of the procedure from $O(n^2)$ to $O(n)$ with respect to the number of paragraph-level text, making it feasible to be used with more than a few thousand text. Specifically, we discovered a simple but effective method of determining which pairwise comparisons should be performed: on text-pairs whose BERT Masked Language Model average embedding are most cosine similar to each another. We tested our algorithm on the task of finding character descriptions with a common trope (theme) and found that our method performs at least 250% as well as the best state-of-the-art paragraph-level embedding baseline. We also found that increasing pairwise comparisons to more than 100 of the most cosine similar character descriptions does not improve performance significantly and is instead a wasteful use of scarce GPU resources that have severe environmental impacts. This can be a valuable takeaway for researchers working on other related text-similarity tasks.

REFERENCES

[1] David Bamman, Brendan O'Connor, and Noah A Smith. 2013. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 352–361.

[2] David Bamman, Ted Underwood, and Noah A Smith. 2014. A bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 370–379.

[3] Kush Bhatia, Kunal Dahiya, Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2019. *The Extreme Classification Repository: Multi-label Datasets & Code*. http://manikvarma.org/downloads/XC/XMLRepository.html

[4] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*. 730–738.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (March 2003), 993–1022. http://dl.acm.org/citation.cfm?id=944919.944937

[6] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 169–174.

[7] Amy Demorest, Paul Crits-Christoph, Mary Hatch, and Lester Luborsky. 1999. A comparison of interpersonal scripts in clinically depressed versus nondepressed individuals. *Journal of Research in Personality* 33, 3 (1999), 265–280.

[8] 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[9] Itay Evron, Edward Moroshko, and Koby Crammer. 2018. Efficient loss-based decoding on graphs for extreme classification. In *Advances in Neural Information Processing Systems*. 7233–7244.

[10] Facebook AI (fairseq Team). 2019. *ROBERTA*. https://pytorch.org/hub/pytorch_fairseq_roberta/

[11] Lea Frermann and György Szarvas. 2017. Inducing semantic micro-clusters from deep multi-view representations of novels. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 1873–1883.

[12] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* (2018).

[13] Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1534–1544.

[14] Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. 2019. Slice: Scalable Linear Extreme Classifiers Trained on 100 Million Labels for Related Searches. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 528–536.

[15] Daniel Jurafsky and James H. Martin. 2019. *Speech and Language Processing*.

[16] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. 1188–1196.

[17] 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. *arXiv preprint arXiv:1907.11692* (2019).

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[19] Paul Mineiro and Nikos Karampatziakis. 2015. Fast label embeddings for extremely large output spaces. *arXiv preprint arXiv:1503.08873* (2015).

[20] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[21] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 993–1002.

[22] Rohan Ramanath, Gungor Polatkan, Liqin Xu, Harold Lee, Bo Hu, and Shan Zhou. 2018. Deploying deep ranking models for search verticals. *arXiv preprint arXiv:1806.02281* (2018).

[23] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 373–374.

[24] Yukihiro Tagami. 2017. AnnexML: Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 455–464.

[25] Avril Thorne and Kate C. McLean. 2001. *Manual for Coding Events in Self-Defining Memories*. Unpublished Manuscript.

[26] Han Xiao. 2018. bert-as-service. https://github.com/hanxiao/bert-as-service.