Embeddings Investigation and Evaluation

Christopher Daly
daly.ch@northeastern.edu
Northeastern University

Cesar Hernandez
hernandez.ce@northeastern.edu
Northeastern University

Paxton Howard
howard.p@northeastern.edu
Northeastern University

Christopher McCune
mccune.c@northeastern.edu
Northeastern University

ABSTRACT

There are many different types of embeddings and they all can be used on the downstream NLP tasks. However, each embedding model will have varying levels of performance across tasks and datasets.

In this project, we are attempting to understand the effectiveness and characteristics of various pretrained embedding models in the context of NLP tasks, with a focus on their suitability for downstream tasks without fine-tuning.

The data collected and visualizations generated are intended to showcase the models' performance on various tasks and illustrate specific characteristics of the embeddings. We focused on five different embedding models: Word2Vec, FastText, BERT, Sentence-BERT (SBERT), and Universal Sentence Encoder (USE). These embedding models were evaluated on standard NLP tasks: word similarity, sentiment classification, sentence similarity, and sentence entailment.

Overall, the results were as expected. Word-based vectors perform better on the word similarity task compared to the sentence-based ones. Similarly, the sentence-based embeddings perform better on the sentence entailment and sentence similarity tasks. However, there were some surprising results with word-based models performing rather well on the sentiment classification task and the BERT model performing poorly on similarity (word and sentence). We demonstrate that while word-level embeddings perform predictably poorly on sentence-level tasks, the reverse is not true, with SBERT emerging as the clear leader in performance across all tasks.

1 INTRODUCTION

From the beginning, a critical component of NLP has been the problem of representation - how does one represent a word or text in a manner such that it can be used as input in a learning model of some kind?

The question we are interested in answering is whether it is possible to represent words and text so that they can be used on downstream tasks without fine-tuning. Is it reasonable to expect to find a universal representation of a word or sentence? Can one expect to find a single embedding that performs well at both the word and sentence level? We will do our best to provide a foundation for answering this question by investigating the embeddings available to us now and test them on several downstream tasks with no fine-tuning, just the frozen weights from the pre-trained model.

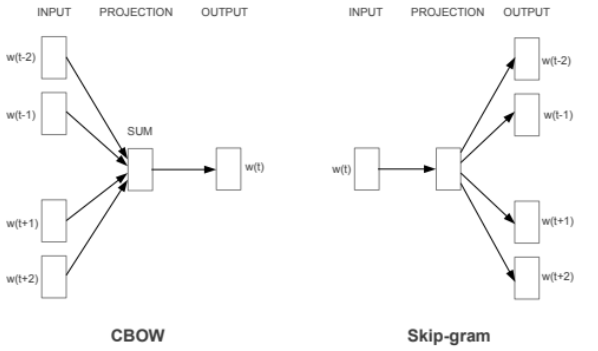

Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

Figure 1: Illustration of CBOW and SkipGram from original Word2Vec paper [Mik+13]

1.1 Vector Semantics

The notion of text representation has evolved significantly over the years. Early on there was the idea of bag-of-words (BOW), but this method had many limitations. BOW gives equal importance to all words and ignores the order of words, which leads to a lack of context and a loss of sequential information. The matrices produced are very sparse and require significant computational resources. Finally, computing similarity between these vectors is impossible due to the orthogonal nature of one-hot encoding.

Thus, the NLP community turned to vector semantics. Vector semantics are the standard way of representing word meaning, and when used this way they are called embeddings, referencing the mathematical concept of visualizing an embedding as a point in a high-dimensional space.[JM20] Because of this, measurements like cosine similarity (or any other distance calculation), can be performed on the vectors.

Intermediate efforts resulted in sparse vectors with lengths that usually matched the vocabulary, so the vectors were often exceptionally long and quite sparse. While it is not altogether clear why, dense vectors perform much better on NLP tasks. This shouldn't be completely shocking, considering a sparse vector might be as long as 50k elements, while a dense vector might only be 300. [JM20] What was needed was a dense encoding alternative with more meaningful values in a smaller dimensional package.

1.2 Word2Vec

Word2Vec [Mik+13] provides that alternative by calculating not how many times a given word occurs in a document, but rather calculating the probability one word is likely to appear with other words. [JM20] Not only did Word2Vec provide a viable dense vector representation of a word, but it retained contextual information such that similarity could now be calculated between two words.

1.3 FastText

One concern that arises is the limitation for morphologically rich languages such as French and Spanish. There can be cases where verbs have many different forms or many different cases for nouns. Because of this, it is often the case that many of these word forms are not used in training data. If many forms of a verb or noun are not used in training, then difficulty arises when learning the representation of a word. This can be seen as even more difficult if the word isn't even in the vocabulary at all.

FastText was an iterative improvement over Word2Vec that helped address this issue. Developed in 2016 by Facebook, FastText's main purpose was to solve the issue of handling "out-of-vocabulary" (OOV) words. The method breaks each word down to character n-grams. For example, using the word 'where' and n = 3, the representation as character n-grams would be: '<wh', 'whe', 'her', 'ere', 're>' and '<where>'. [Boj+16] Using character n-gram representations allows FastText to gather semantic information that otherwise would have been overlooked. This produced promising results compared to Word2Vec, but as more powerful architectures like neural networks became computationally feasible, even better performances were achieved.

1.4 Other Approaches

Recurrent Neural Networks (RNNs) improve on the standard neural network by adding a "memory" that can capture information about previous computations. Long Short Term Memory networks (LSTMs) are a special type of RNN that use computational gates to reduce the probability of vanishing/exploding gradients which makes them capable of learning long-term dependencies.

However, RNNs are sequential models by nature and thus prohibit parallelization, which in turn leads to long and computationally expensive training. This drawback led to the idea of the transformer, which sheds recurrence and relies solely on the concept of attention. [Vas+17] Not only are transformers efficient, but they also outperform all previous models.

1.5 BERT

A very successful formulation of the transformer architecture is the Bidirectional Encoder Representation from Transformers (BERT). BERT uses contextual embeddings instead of static embeddings, which allows for the meaning of words to be captured based on the surrounding words in a sentence, including words after the current word. Additionally, since BERT is based on the transformer architecture it has a self-attention mechanism [Vas+17] that can give a different weight/importance to each word in the sentence. These improvements over previous methods as well as the amount of data used for training produce an embedding that outperforms all previous models on benchmark NLP tasks. [Dev+18] However, BERT is still fundamentally a word-level technique.

1.6 Sentence BERT

The next logical iteration on this theme is SBERT, which is optimized for sentences and arbitrary-sized pieces of text. The downside to BERT is that it struggles to derive independent sentence embeddings. Sentence-BERT (SBERT) solves this issue by generating high-quality sentence embeddings.

SBERT is a modification of the BERT network utilizing siamese and triplet networks [RG19]. Siamese networks can be thought of as using two identical BERTs sharing the same network weights. The concept of using a Siamese network passes two sentences independently through the same BERT model. Once a sentence is passed through, a pooling layer is then applied. This yields the lowest dimensionality representation. Siamese networks can be utilized for identifying duplicate values and even facial recognition.

We can then use cosine similarity to compare the sentence embeddings produced.

In a collection of about 10,000 sentences that requires roughly 50 million computations that would take BERT about 65 hours, SBERT could complete this in roughly five seconds. It can be used in situations that aren't computationally feasible to be modeled using BERT. On a GPU, it is also about 55% faster than Universal Sentence Encoder [RG19].

This brings us to the notion of a universal encoder (the idea that we could embed enough of the 'nature' of a piece of text that we could use on downstream tasks without fine-tuning).

1.7 Universal Sentence Encoder

The Universal Sentence Encoder is a type of pre-trained deep learning model that uses a complex neural network to convert text sentences into embeddings. It is designed to capture the semantic meaning of sentences by using word order, making it possible to compare and analyze sentences in a meaningful way. This universal encoder is trained on a large amount of text data and can be used for various natural language processing tasks without the need for task-specific training while still providing state-of-the-art results especially when context and nuanced semantics are crucial. [Cer+18]

We put these embeddings put to the test on four NLP tasks each associated with well-known datasets; the Wordsim-353 dataset for word similarity, the IMDB reviews dataset for sentiment classification, the STSB dataset for sentence similarity, and the SNLI dataset for sentence entailment. The results provide practical conclusions as to which model should be used based on the task, strengths, and weaknesses of each, and which is best overall in terms of performance on downstream tasks.

2 DATA

2.1 Wordsim-353

Wordsim-353 contains 353 word pairs along with mean similarity scores as assigned by humans. Human volunteers were asked to rate word pairs from 0 (for totally unrelated) to 10 (for nearly the same or identical). [Fin+01]

2.2 IMDB Movie Reviews Dataset

[Maa+11] A widely used dataset for sentiment classification, it contains 50,000 movie reviews, divided into evenly divided train and test sets, with balanced positive and negative labels.

2.3 Semantic Textual Similarity Benchmark (STS-B)

This dataset is frequently used as a benchmark to evaluate the performance of models trained to detect similarity in text. It comprises 8,628 sentence pairs obtained from news headlines, user forums, and image captions. Each pair of sentences is labeled with a human score ranging from 0 to 5. [Wan+18]

2.4 Stanford Natural Language Inference(SNLI)

[Bow+15] Also known as Recognizing Textual Entailment (RTE), this is a collection of sentence pairs, text and hypotheses, with the assessment of five human judges as to whether the sentence pair provides an example of entailment, contradiction, or neutral. A consensus label is also provided.

3 METHODOLOGY

We evaluated our chosen embedding types against four tasks which were chosen to help expose the relative strengths and weaknesses of each model. These tasks were: Word similarity, Sentiment Classification, Sentence Similarity, and Text Entailment.

Each pre-trained model is loaded and used to embed the input text for the dataset associated with the given task. As much as possible, we avoided preprocessing the data, but in certain cases, it was necessary, such as when using BERT, or to concatenate input vectors, as was the case in the entailment experiments. We preferred the simplest models possible, generally a default sklearn logistic regressor, though, in the case of the entailment task, a small neural net was required.

3.1 Word Similarity

After the embeddings were generated, similarity was calculated by determining the scaled dot product between the two words, and adjusted to range from 1 to 10 as the human ratings do. Pearson and Spearman coefficients were calculated between the similarity ratings and scatterplots and histograms were used to help visualize the relationship between human and computer scores.

3.2 Sentiment Classification

After obtaining embeddings for the text reviews, a logistic regression model was fit to the training data, and used to predict the test data. Accuracy, precision, recall, and f1 values were all obtained, and Precision-Recall and ROC curves were both plotted.

3.3 Sentence Similarity

The methodology was nearly the same as for word similarity, except that the embeddings are now sentences, and in the case of word2vec and fasttext, this involves taking the mean embedding of all the words. The cosine similarity was scaled to match the human judges and the same comparisons were performed as for word similarity.

Model	Word Similarity (Pearson)	Word Similarity (Spearman)	Sentence Similarity (Pearson)	Sentence Similarity (Spearman)	Sentiment Classification (Accuracy)
Word2Vec	0.6525349615944825	0.7000166486272194	0.4578895419694716	0.4739970106262557 7	0.61060
FastText	0.7401008932966374	0.7913901262734915	-0.06215074335035552 4	-0.05913654287610031	0.85396
BERT	0.16738713919264835	0.2530104289523 9685	0.22312660547358934	0.22836265440866804	0.87244
S-BERT	0.6873068579940221	0.7238981956171668	0.8348265370 27969	0.8190235436086622	0.81124
USE	0.6588904889187679	0.6903941009151461	0.7820558951457802	0.7497158243373299	0.85948

Figure 2: Raw Pearson/Spearman Correlation Results by Model

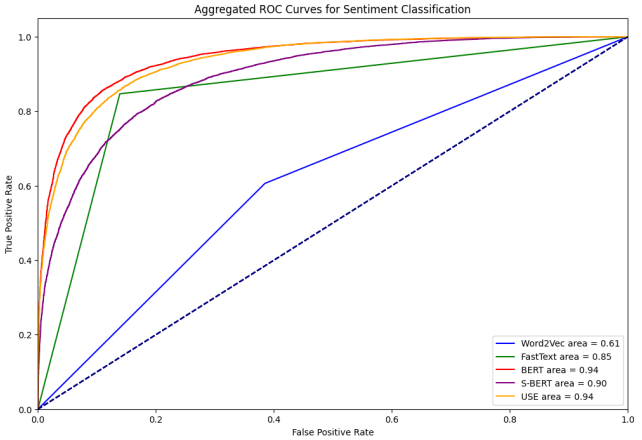

Figure 3: Aggregated ROC Curves for Sentiment Classification

3.4 Sentence Entailment

Using the SNLI dataset, the concatenated embeddings for text and hypothesis were used to train a neural model. This was compared against human responses.

4 CODE

There were some challenges developing the code for this. Primarily, it was often difficult to find a way to calculate all the embeddings necessary in a way that the environment could handle it. BERT was the worst in this regard.

Where possible, we favored parallelization and vectorization for efficiency.

The code for this paper is available at https://colab.research.google.com/drive/1B_icQFI4EYb4-0_rpsBH3RX9pHOVGz6K.

5 RESULTS

Here we'll report the actual results in a matter-of-fact way.

5.1 Word2Vec

Despite its age, Word2Vec performed moderately well. It received .65 Pearson and .7 Spearman scores. While it did perform poorly on sentence similarity, with a .45 and .47 on Pearson/Spearman, this was expected for a word-based embedding.

What was just a bit surprising was how well it performed on sentiment classification (.61 accuracy). Entailment was predictably poor with a .48 accuracy.

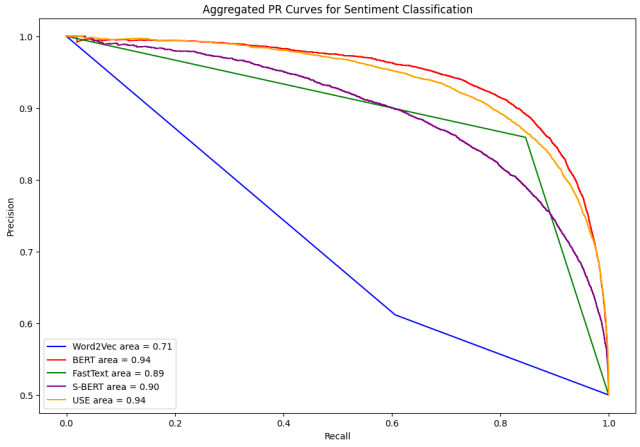

Figure 4: Aggregated PR Curves for Sentiment Classification

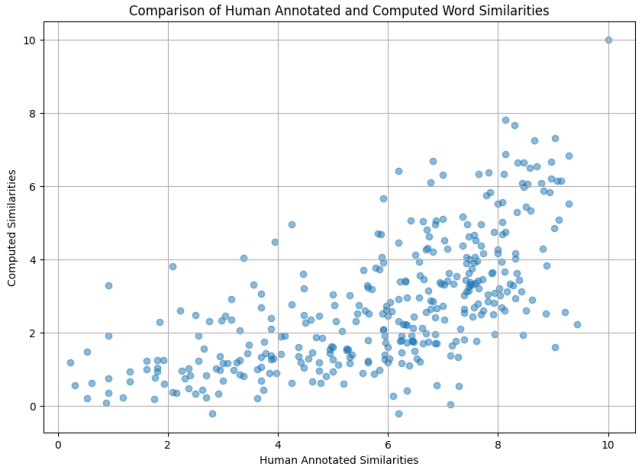

Figure 5: Wordsim-353: Human-Rated vs Machine-Calculated Similarity Scores Using Word2Vec

5.2 FastText

FastText performed quite well on word similarity, receiving a .74 and .79 for Pearson and Spearman Correlation scores, respectively, but failed spectacularly to detect sentence similarity with scores of -0.06, and -0.06

Sentiment classification was a bright spot with .85 accuracy, while entailment trailed at .48, more or less tied with Word2Vec, the other word-based model in the survey.

5.3 BERT

BERT performed quite poorly in both similarity tasks, achieving .19 and .25 on Pearson and Spearman correlation scores on the word similarity and just .223 and .228 on sentence similarity.

It performed excellently on sentiment classification with 87.2 accuracy, but just middling on entailment with a .61 percent.

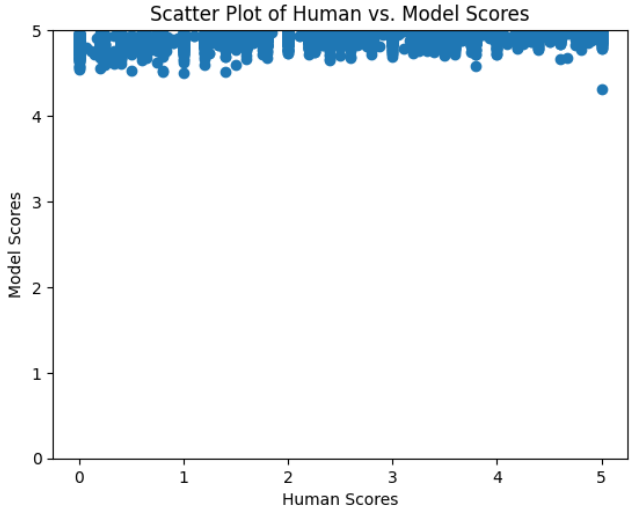

Figure 6: STS-B: Human-Rated vs Machine-Calculated Similarity Scores Using Word2Vec

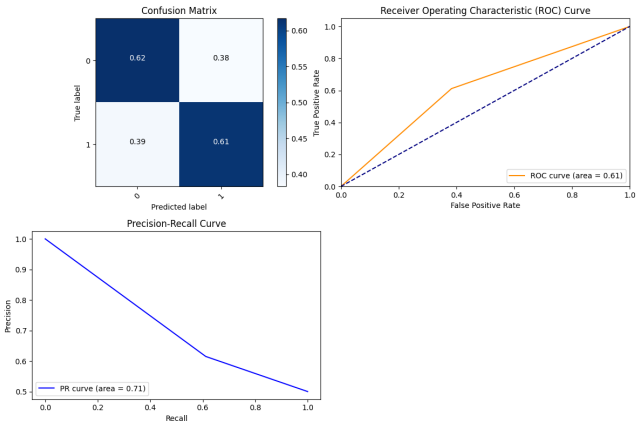

Figure 7: Word2Vec's Performance on the Sentiment Classification Task

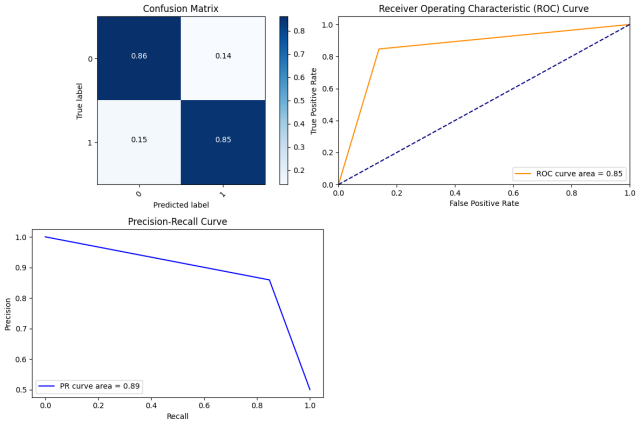

Figure 8: FastText's Performance on Sentiment Classification

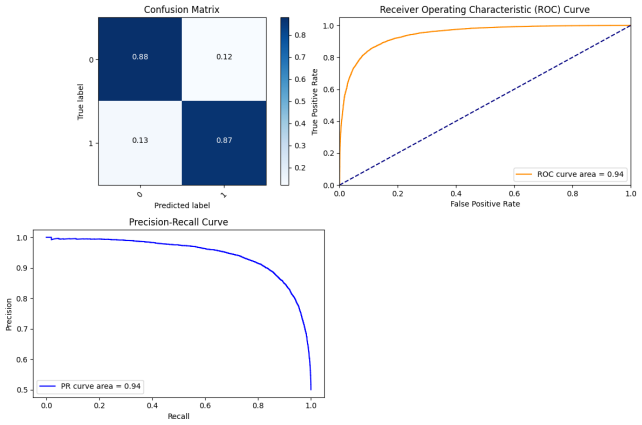

Figure 9: BERT Performance on Sentiment Classification Task

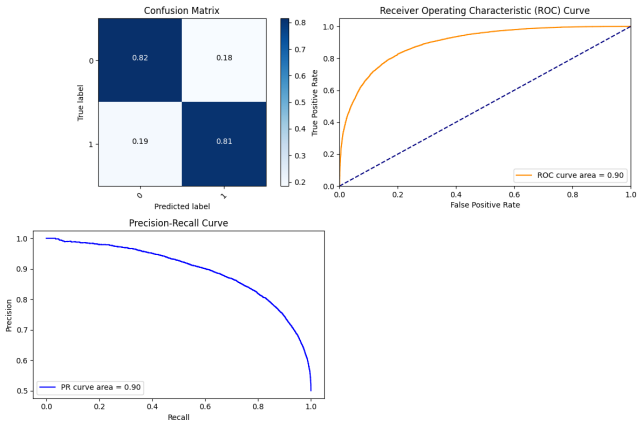

Figure 10: SBERT Performance on Sentiment Classification Task

5.4 SBERT

- SBERT received a .69/.73 result on the word similarity task, and an impressive .83/.81 on sentence similarity.

For sentiment classification, the result was .82 accuracy and for entailment, .79.

5.5 USE

- For word similarity, USE earned a .66 Pearson and .69 Spearman.

USE earned a .78 and .75 for Pearson and Spearman scores, respectively, as well as .86 accuracy for sentiment classification, and a .79 for entailment (matching SBERT's score).

6 DISCUSSION

It's fair to say that no investigative ground was broken here. Word-based vectors do indeed perform better on tasks at that level of granularity, but it's also true that the sentence-level embeddings are at no such disadvantage at the word level.

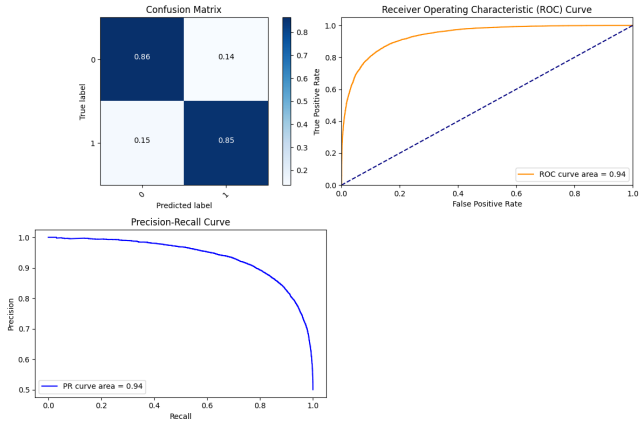

Figure 11: USE Performance on Sentiment Classification Task

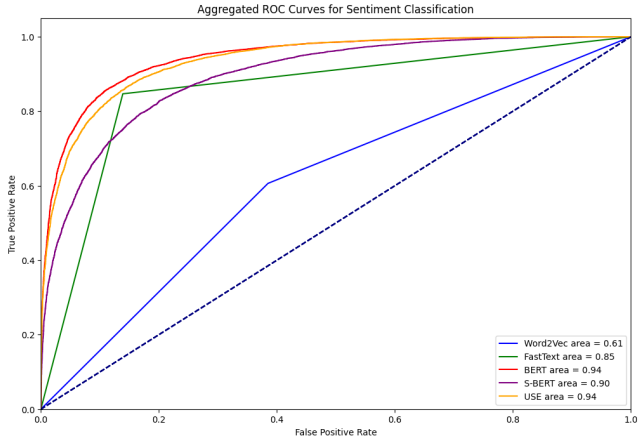

Figure 12: Aggregated ROC Curves for Sentiment Classification

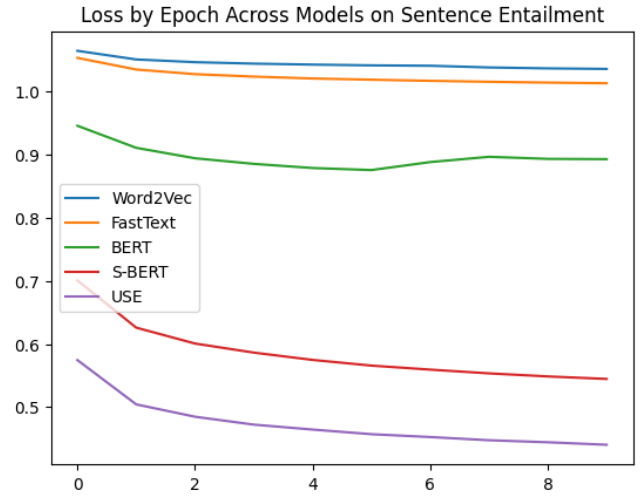

Figure 13: Loss over Epoch Across Models for Sentence Entailment

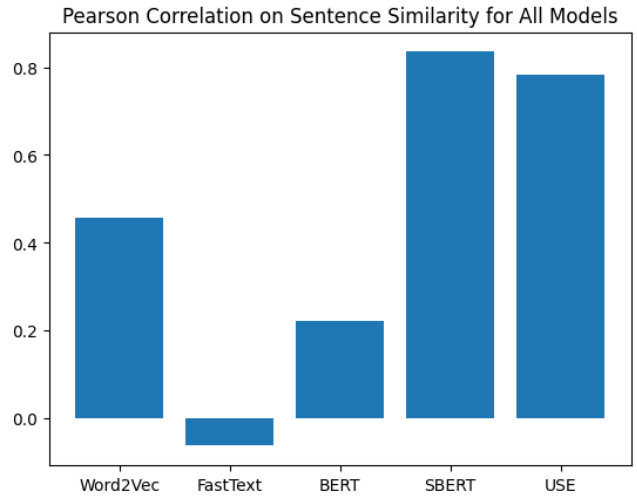

Figure 14: Aggregated Pearson Scores for Sentence Similarity

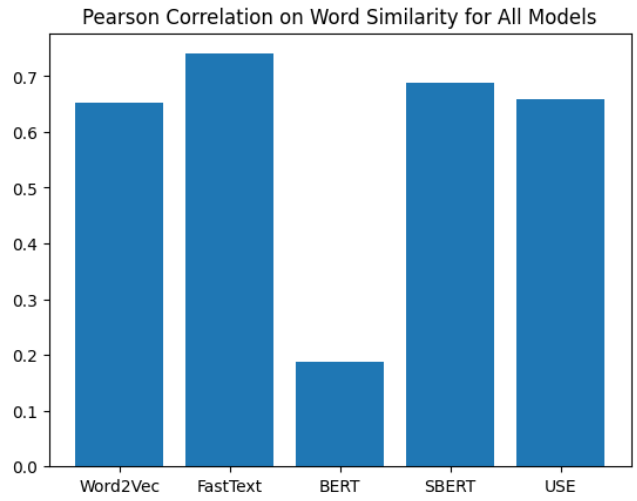

Figure 15: Aggregated Pearson Scores for Word Similarity

6.1 Interesting Results

There were some results that were genuinely interesting and should be considered, both in the context of selecting a potential embedding for a project, but also to identify areas in which further insights can be gleaned.

6.1.1 Word2Vec. Word2Vec did surprisingly well on sentiment classification, a task that relies on sentence-level embeddings. This seems to suggest that counting word frequency is sufficient for a prediction accuracy significantly better than chance.

6.1.2 BERT. BERT performed very poorly on similarity in general. While we were initially quite surprised, perhaps we ought not have been. According to the authors of the SBERT paper, the sentence-level embeddings that are obtained through averaging the words in the sentence, or even using the CLS embedding are inferior: "These embeddings are often worse than those achieved by averaging GloVe embeddings". [RG19]

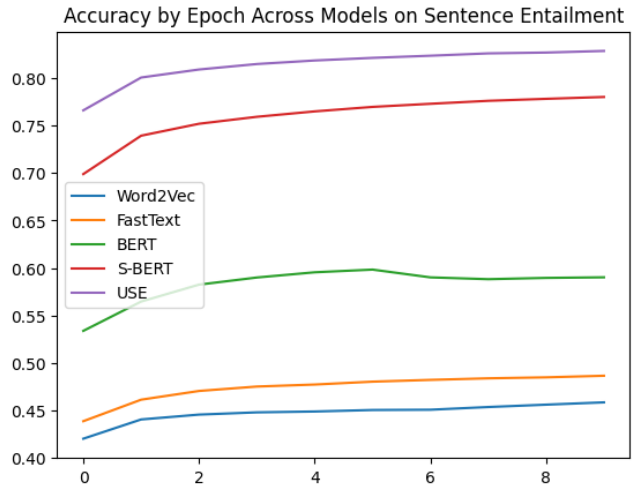

Figure 16: Performance across Models on Sentence Entailment

6.1.3 SBERT. The only notable characteristic of the SBERT results is that it was the all-around best model in our testing. It represents a significant improvement over BERT.

6.1.4 USE. Universal Sentence Encodings didn't provide the best-performing embeddings, but it was a close thing.

6.1.5 Word vs Sentence. Not surprisingly, the word-level embeddings were outclassed by the sentence-level when evaluated against a sentence similarity task. What was a bit surprising was FastText's abysmal score when compared to Word2Vec. This seems to suggest that something about FastText's subword strategy is at cross-purposes to the sentence similarity task.

The sentence-level embedding methods like USE or SBERT are not outclassed by the word-level models. This is encouraging from the perspective of finding a universal representation for text of all sizes.

For someone making the choice, it's either going to come down to performance of the model (they have the time and resources for the best), or it will come down to the limitations of the project (what's the best model we can afford for our budget?).

7 FUTURE WORK

In a rapidly evolving landscape of natural language processing (NLP), the quest for optimal embedding techniques persists, promoting a need for comprehensive exploration in two key areas: improved relevant benchmarking and the pursuit of universal embeddings. We hope to delve into the significance of these domains, providing insights into potential avenues for future research and development.

7.1 Benchmarking

Benchmarking, a cornerstone in evaluating the efficacy of embedding methods, is paramount to gauge the progress of the field.

To propel this forward, it is imperative to consider the following aspects; diversity of tasks and dataset quality. Expanding the repertoire of benchmark tasks is critical. While sentiment analysis and text summarization remain staple challenges, inclusivity of other tasks, such as question answering and multi-model challenges, is essential.

This broader scope ensures a more comprehensive evaluation of embedding methodologies. As for dataset quality, the quality of benchmark datasets plays a pivotal role. Representative, diverse datasets free from biases ensure that embeddings are not tainted by skewed perspectives. This consideration becomes particularly crucial in real-world applications where biased embeddings can have far reaching consequences.

For example, a dataset collected for sentiment analysis specifically focusing on product reviews from an e-commerce platform. The quality of this dataset would significantly influence the reliability and generalizability of any sentiment analysis model trained on it.

7.2 Universal Embeddings

The concept of universal embeddings revolves around the creation of embeddings that exhibit effectiveness across a multitude of tasks without necessitating task-specific fine tuning. Multi-Task learning and transfer learning strategies are some key considerations in this realm.

The simultaneous training of models on multiple tasks provides a promising avenue. Multi-task learning endeavors to generate representations that generalize well across various tasks, reducing the need for task-specific adaptations. Next, exploration of diverse transfer learning strategies is pivotal. This involves pre-training on extensive corpora to impart general linguistic knowledge, subsequently transferring this knowledge to downstream tasks.

One example for multi-tasks can be when a system needs to comprehend and analyze user-generated content from social media. The content will most likely encompass a multitude of tasks, each needing a unique set of language understanding capabilities such as emotional analysis and textual descriptions.

The future of embedding research rests on these two pillars: refined benchmarking practices that reflect the complexity of NLP tasks and the pursuit of universal embeddings that transcend task-specific fine-tuning. As researchers delve into these areas, the potential for more robust, adaptable, and widely applicable embedding solutions for diverse NLP tasks becomes increasingly promising.

7.3 Enhancements to this Investigation

This work could be enhanced with the addition of data describing the relative time and computation costs, which for some of the models are not insignificant. For the initial study, we concerned ourselves exclusively with task performance, but a realistic scenario can be imagined where time and computation costs were prohibitive enough that choosing a more poorly performing, but faster model would be a sensible choice.

8 CONCLUSION

We looked at a variety of embedding methods and downstream NLP tasks to determine baseline performance one can expect when using pretrained embeddings downstream with no fine-tuning.

The clear conclusion from our experiments is that word-level embeddings like Word2Vec and FastText simply aren't suited to modeling text sequences, and, as a result, generally fail at tasks that require sentence-level representations.

This isn't to say they don't perform adequately at the word level, nor that they don't sometimes manifest surprising fluency at the sentence level, as with sentiment classification. Theys simply aren't constructed to capture contextual information from text sequences.

More contextual, sequence-based embeddings, like USE or SBERT don't suffer from the same limitation. They seem to adapt with no trouble to the odd word-level task. This is promising in the context of the search for universal text representations.

9 REFERENCES

[Fin+01] Lev Finkelstein et al. "Placing search in context: The concept revisited". In: *Proceedings of the 10th international conference on World Wide Web*. ACM. 2001, pp. 406–414.

[Maa+11] Andrew L Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*. 2011.

[Mik+13] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv preprint arXiv:1301.3781* (2013).

[Bow+15] Samuel R Bowman et al. "A large annotated corpus for learning natural language inference". In: *arXiv preprint arXiv:1508.05326* (2015).

[Boj+16] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *arXiv preprint arXiv:1607.04606* (2016).

[Vas+17] Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.

[Cer+18] Daniel Cer et al. "Universal Sentence Encoder". In: *arXiv preprint arXiv:1803.11175* (2018).

[Dev+18] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[Wan+18] Alex Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 353–355.

[RG19] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *arXiv preprint arXiv:1908.10084* (2019).

[JM20] Dan Jurafsky and James H. Martin. *Speech and Language Processing*. 3rd ed. Draft, 2020.

A ANNEX-A

A.1 Team Contributions

A.1.1 Christopher Daly. Attended and contributed to design meetings. He wrote the code that produced the data analyzed and contributed to every section of the paper in addition to preparing and taking part in the video presentation.

A.1.2 Cesar Hernandez.

A.1.3 Paxton Howard.

A.1.4 Christopher McCune.