CSC413 Final Project

Philipp Eibl
Department of Computer Science
University of Toronto
philipp.eibl@mail.utoronto.ca

Jaipratap Grewal
Department of Computer Science
University of Toronto
jp.grewal@mail.utoronto.ca

Jason Lee
Department of Computer Science
University of Toronto
jasonkjl.lee@mail.utoronto.ca

Abstract

Paragraph/Context-based question-answer neural networks have been researched for quite a while. In a similar light, we compare the performance of various types of word embedding models on reading comprehension based questions using different types of classifier models.

1 Introduction

This project is based on reading comprehension with neural networks - one of the hot research topics in recent times. We compare the effectiveness of different word embeddings in the task of classifying a question/answer pair as correct or incorrect.

Specifically, given a reading passage (ex. "In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity"), an input query (ex. "What causes precipitation to fall?"), and a proposed answer as a sequence of text from the passage ("gravity"), we want a model that determines whether the proposed sequence correctly answers the input query.

2 Related Works

A number of approaches have been developed to tackle the problem of question answering. The most successful models mostly stem from the last 10 years, with a great abundance in neural approaches. Below are examples of the papers discussed in this project.
Word2Vec: A 2-layer neural network is used in either one of the Continuous Bag of Words (CBOW) or Skip-gram model, where the former produces attempts to predict a word using its surrounding words in a sequence and the latter predicts the surrounding words given only one input word.
GloVe: Similar to word2vec, GloVe also attempts to encode the co-occurrence probability ratios of two words into the vector mapping of a given word.
BERT: Aside from using a more complex architecture, BERT's transformer encoder takes in an entire sequence at once (as opposed to the previous models, which read one token at a time from left to right). Two other notable differences of BERT are [MASK] tokens and next sentence prediction, which it does using a bi-directional architecture.
GPT: While similar to BERT, GPT only looks at the previous word(s) in context when predicting ahead (in training), compared to BERT which looks at the context word before/after.

3 Methods

3.1 Data Processing

The first task deals with processing the JSON formatted file containing the SQuAD dataset. We clean the data by breaking it into arrays containing the paragraphs (from now on contexts), questions, answers, and the corresponding labels (correct/incorrect). Being given only the correct answers, as the original task deals with picking an answer from the context and comparing with the correct one, we manually picked substrings from the context and add them as incorrect training example (making sure we don't pick the correct answer randomly!). Note that this allowed us to add arbitrary incorrect answers for one correct answer.

3.2 Models

We worked with 4 pre-trained models (BERT, GPT, Word2Vec, GloVE) to generate the word embeddings (further finetuning was not considered due to level of resources needed), and we trained 3 types of ML models to compare the performance of these embeddings (Linear, Linear + ReLU, Random Forest), in addition to a random baseline.

3.2.1 BERT & GPT

We used an open source PyTorch implementation of the transformers (similar to A2). The resulting tokenizers were used to act upon the context, question, answer arrays to retrieve 512 (maximum parameter length) component vectors and the corresponding attention masks (as BERT, GPT are directional/bi-directional sequence based). The only difference for the procedure between the two was that BERT incorporates start/end of sentence tokens in its examples. It was a similar case when comparing with BERT, which did much better than these two models.

For a single training example, we concatenate all three vectors (1 for the reading passage, 1 for the question, 1 for the answer) into a single 1536 vector which is used as input into our linear classification model.

3.2.2 Word2Vec & GloVe

We utilised similar open source models as in A1. For the word2vec embeddings, we used word-to-vector mappings based on a model trained on the Brown corpus (a fairly old and small text corpus, so a more recent and/or bigger one would most likely result in higher classification accuracies). Similarly the GloVe model uses 50-dimensional embeddings from a pre-trained that was trained on an openly available dataset comprising of 6bn words from Wikipedia articles in 2014. A model pre-trained on a 42bn-word corpus using 300 dimensional embeddings resulted only in a marginal increase in accuracy, leading us to use the much more computationally efficient corpus mentioned before. We then used the word embeddings to test 2 separate approaches for classification: First, use a distance measure (one of cosine similarity or euclidean distance) of the answer token embedding(s) and the mean of the context token embeddings as a threshold to classify whether the answer token is correct. Second, pick the sentence in the context with the highest similarity to the question (using the mean vectors of the token embeddings), then use only the similarity of that phrase and the answer-candidate as a threshold.

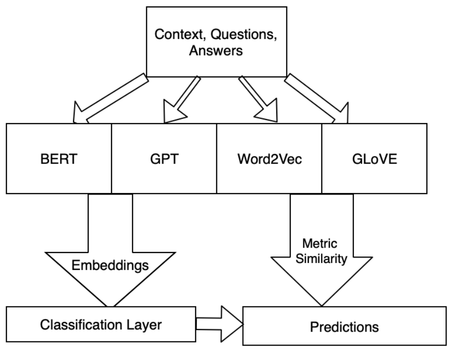


Figure 1: Algorithm

3.2.3 Classification Models

One model was a simple fully-connected layer corresponding to the dimension of the output word embedddings. The second model incorporated a non-linearity by using a ReLU activation layer in between two fully-connected layers. For further performance comparisons, a simple 'sklearn' based Random Forest with varying tree-depth and parameters was used.

Note: All models were added on top of the embedding architectures. A random baseline was also used for comparison.

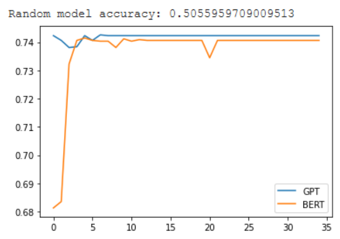
4 Results/Discussion

932 Q/A pairs	
BERT(2 layer)	0.823
GPT(2 layer)	0.759
BERT(1 layer)	0.764
GPT(1 layer)	0.754
BERT/GPT Random Forest	0.860
word2vec	0.747
GloVe	0.780
Random Model	0.487

17868 Q/A pairs	
BERT(2 layer)	.741
GPT(2 layer)	.742
BERT(1 layer)	.739
GPT(1 layer)	.734
BERT/GPT Random Forest	0.792
word2vec	0.748
GloVe	0.755
Random Model	.506

Our experiments correspond to the performance of these various mentioned word-embeddings under the classification models.

The results on various number of Q/A pairs is shown above - with the BERT/GPT being trained for 35/50 epochs, thus helping to maintain a uniform standard, whereas the Word2Vec/GloVE models predicted the answers directly - but we also analyse the effects of number of paragraphs.



```
Random model accuracy: 0.5055959709009513
```

Figure 2: Val Acc/Epoch: 2 Layer, 300 Paras

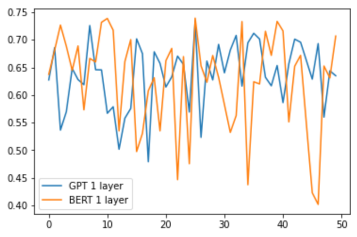


Figure 3: Val Acc/Epoch: 1 Layers, 300 Paras

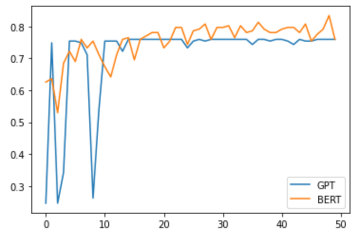


Figure 4: Val Acc/Epoch: 2 Layers, 30 Paras

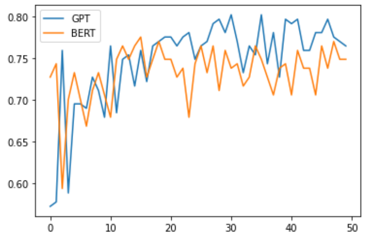


Figure 5: Val Acc/Epoch: 1 Layer, 30 Paras

4.1 Classification Models (for BERT/GPT)

We expected the random-forest classifier to perform the best due to its highest non-linear nature out of the bunch. The logistic regression model was expected to be a close second because first, the random-forest may not always correspond well to NLP tasks, and second, the ReLU non-linearity gives a better truth-approximator than say the linear layer - which was predicted to have the lowest accuracy.

The results matched very well with expectations. To begin, the random baseline model corresponded to about 51% accuracy, and as we mention next, all models performed much better than the baseline.

BERT + Single Layer gave an accuracy of .751, compared to BERT + Two layer, which gave an accuracy of .782. GPT + Single layer gave an average accuracy of .744, whereas the two layer model gave .75. Lastly, the random forest classifier overfits the data badly (training acc: .99, val acc: .85) but gave similar results on the BERT/GPT.

4.2 BERT vs. GPT

It was difficult to predict a-priori which one of these two models would perform better. But comparing the results across all categories clarifies that BERT had a better performance in most cases. This may be because the joint bi-directional conditioning suits better to this task/dataset as compared to GPT.

An interesting trend to note is that the validation accuracy peaks quite early on for the 2-layer models (300 & 30 paras) and mostly stabilises for the rest of the training whereas both 1-layer runs, the validation accuracy peaks slower but is highly unstable. The variance can be attributed to the amount of different words and types of sentence structuring - which one can expect to be more when choosing more paragraphs/pairs to train/test on. But at the same time this points to the instability of the linear layer and how much a small non-linear activation can help improve performance/stabilization.

Given its much more sophisticated architecture it was reasonable to expect GPT/BERTS's performance to beat previous approaches by a decent margin. With an improvement of around 10 absolute percentage points, it is clear that this model(s) offers a statistically significant boost in comparison to both word2vec and GloVe, who both reached peak accuracy of less than 80 percent.

With respect to the latter two, we experimented with multiple tests including different similarity measures (cosine vs. euclidean), picking the phrase in the context that most likely contains the answer sequence as the new "paragraph", as well as choosing different threshold similarity values for classification. We can note that perhaps not surprisingly, GloVe outperformed Word2Vec by a few absolute percentage points in every test.

Overall, our experiments seem to confirm that transformer-based models perform much better than more traditional embedding models. Most of the times, it seems that BERT is the best model, but this may change on different tasks/contexts/datasets.

unstable. The variance can be attributed to the amount of different words and types of sentence structuring - which one can expect to be more when choosing more paragraphs/pairs to train/test on. But at the same time this points to the instability of the linear layer and how much a small non-linear activation can help improve performance/stabilization.

4.3 Word2Vec/GloVe vs GPT/BERT

Given its much more sophisticated architecture it was reasonable to expect GPT/BERTS's performance to beat previous approaches by a decent margin. With an improvement of around 10 absolute percentage points, it is clear that this model(s) offers a statistically significant boost in comparison to both word2vec and GloVe, who both reached peak accuracy of less than 80 percent.

With respect to the latter two, we experimented with multiple tests including different similarity measures (cosine vs. euclidean), picking the phrase in the context that most likely contains the answer sequence as the new "paragraph", as well as choosing different threshold similarity values for classification. We can note that perhaps not surprisingly, GloVe outperformed Word2Vec by a few absolute percentage points in every test.

5 Conclusion

Overall, our experiments seem to confirm that transformer-based models perform much better than more traditional embedding models. Most of the times, it seems that BERT is the best model, but this may change on different tasks/contexts/datasets.

Secondly, our results are only based on a small, random subset of questions from the SQuAD data-set (less than 10%) and no fine-tuning of the BERT/GPT themselves, which if taken into account may give even more significant results. We predict that GPT may actually perform even better on the whole of SQuAD due to its sheer parameter size.

All in all, we estimate such transformer-based models will influence the present as well the coming future with their powerful architectures/new research.

Contributions

Philipp, Jaipratap, Jason worked on the (Word2Vec, GloVE), BERT, GPT respectively and double checked each others' work too. For the compilation of the report, all three authors added to the Methods/Discussion after discussing the results of their coded models. Overall, everyone contributed equally and helped each other out.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[4] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[5] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.