BERT has a Mouth, and it can Taste:

Identifying Points of Interest using BERT with Triplet Loss

Le Zhang, Richa Rastogi, Yanlin Qu

Abstract

While Named Entity Recognition is a well defined NLP task, verifying those entities against a database with several million POI's stored is not. In this project, we aim to develop a system using BERT with triplet loss, where the word embeddings are trained on a dataset of 17 million POI's and 11 million non-POI's. Given a user query the system classifies whether the user query is a valid POI or not. If it is a valid POI, then it is searched through the corpus and returned.

1 Introduction

Extracting some entities from a snippet of text is a common task when performing any sort of natural language processing. For those extracted entities, how can we efficiently distinguish our points of interest (POI) from other things (non-POI)? In this project, POI's are location names.

The POI verification task is crucial in many applications like navigation systems while this verification can be challenging because of the following possible reasons. Firstly, whether an entity is a POI may depend on context due to polysemy. For example, in the text "Go Stanford", the entity of Stanford can be a university or a person. Secondly, the extracted entity may not be exactly the same as the corresponding POI in the database (e.g. Stanford vs Stanford University). People may refer to a POI by its nickname or omit some characters of the name.

Currently, the POI verification is performed by searching the database. Some use sliding window to split the entity string into prefix, pure name and suffix then search the prefix and pure name in a big dictionary and use k-NN and another dictionary to find the suffix.

In this project, we aim to develop a properly trained deep learning language model with triplet loss to produce good embeddings for this verification purpose. Moreover, we use these embeddings to identify POI's. The input of our model is an Chinese string of some entities. Then we compare the embedding of this string with those POI embeddings to determine whether this string is a POI, if yes, the model will output the most possible POI that the string can be.

2 Related work

Word embeddings is a well studied topic and while there have been many algorithms to convert text to word embeddings so that various tasks can be performed with these vectors, we chose to focus on BERT [1] with this project. A language model can be used either for sentence scoring or for sentence

generation. BERT is a language model trained using a “masked language model” (MLM) technique (Taylor, 1953), and has been utilized largely as a pure language model. For instance, a blog titled *Can We Use BERT as a Language Model to Assign a Score to a Sentence?*link here tried to assign the mean perplexity to a random sentence using the pretrained BERT, where the predicted word itself was fed as an evidence. In the GitHub repository xu-song/bert-as-language-model [11], the author attempts to replace the traditional RNN language model with a bidirectional language model achieving better performance. Alex Wang and Kyunghyun Cho also presented work [12] [10] that discusses how BERT is a generative model and performs very well for both sampling and ranking as a pure language model. In another related work Elman, Alex, Kyunghyun discuss employing undirected language model to generate sequences with competitive results.In the context of sentence generation, Sean Welleck et al [13] discuss how text generation can be achieved without pre-specifying a generation order.

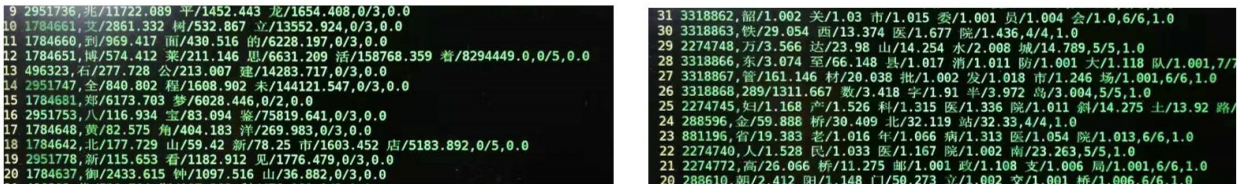
Finally, triplet loss became popular with Facenet[15] for its implementation in computer vision. More recently with Sentence Bert [3] the authors found that the idea of triplet loss can be extended to NLP with the loss function penalizing the distance between certain word embeddings more than others. In this paper, they experimented with pre-trained BERT, RoBERT and other models along with triplet loss. This led to impressive results for downstream NLP tasks such as natural language inference, semantic searching, clustering of word embeddings etc.

3 Dataset

Data Cleaning and preprocessing included standardizing aumlets, tokenizing the POI’s, removing spaces, converting to lowercase (for English dataset) and forming the dataset as a triplet.

We have a training corpus containing 17 million POI’s (location names in Chinese) and 11 million non-POI’s (location-irrelevant queries) from Alibaba. Our dev set contains 10 thousand POI’s and 10 thousand non-POIs. This dataset is protected by the Alibaba data security policy. Some samples are provided in the pictures below. All our results are based on this dataset.

We also used a dataset from Kaggle that contained 400,000 POI’s for experimenting, which was divided into 80% training, 10% dev and 10% test set. Each record consists of an anchor, a positive that was randomly sampled from the set of POI’s and a negative. The negative examples were obtained from user queries for language, currency, dictionary [5][6]. These negatives especially language, currency did not contain location POI’s and provided hard triplets. Example: anchor - France, positive - Italy, Negative - French.



Some samples of the Chinese POI dataset

4 Methods

The model used for forming embeddings is trained from scratch on the masked language model of base BERT model. To improve these embeddings, the model is trained on triplet loss thereafter. For the small English dataset, we tried to search the existing dataset for the incoming POI to find the closest one. For the big Chinese dataset, searching will be slow, so we tried to taste the incoming POI using the BERT's mouth to tell if our model has read something similar.

Architecture and Training Details

1. Training BERT Model - We started with the base BERT model [1]. This model was trained from scratch by making modifications to how the training data was fed for training. In the training set, one character was masked at random for each POI and each POI was sampled 10 times. Training was then conducted using Adam Optimizer, batch gradient descent with batch size of 512 and masked language model loss function. Hardware used for training consisted of 8 V100 cards and training duration lasted for about 5 hours. We provide a diagram for this training mode (pretraining) on the next page.

2. Adding triplet loss - In order to improve the word encodings and be able to discriminate POI from non-POI, the language model obtained from #1 was further trained using the triplet loss. Here hard triplets were mined using Batch All strategy for online triplets as described in [2]. The distance metric used for triplet loss evaluation were Euclidean, cosine and Manhattan distance as in [3]. Though not significant improvements were achieved with the addition of triplet loss for this task, the language model loss reduced from 2.8 to 2.3, resulting in better word embeddings. We provide a diagram for this training mode (fine tuning) on the next page.

3. Search for a POI - Given a user query, such as 'Italy', the system is intended to classify it as a POI with binary classification of true and false. For a user query classified as a POI, k-NN was used to search the corresponding POI in the dataset. We used k=10 to return 10 nearest neighbours for that user query.

4. Let BERT taste - For a sentence of 15 characters, each time we masked one character then put into the BERT. We get a distributions after the softmax final activation layer. We can sample top ten characters and see whether they contain the true masked character. If this is true for 9 characters, then we use the ratio 9/15 to predict whether this sentence is a POI (true if it is larger than some threshold). This method is reasonable because the BERT has only been trained using POI's so it has a good taste only for POI's and bad taste for others. We provide a diagram for this predicting mode on the next page.

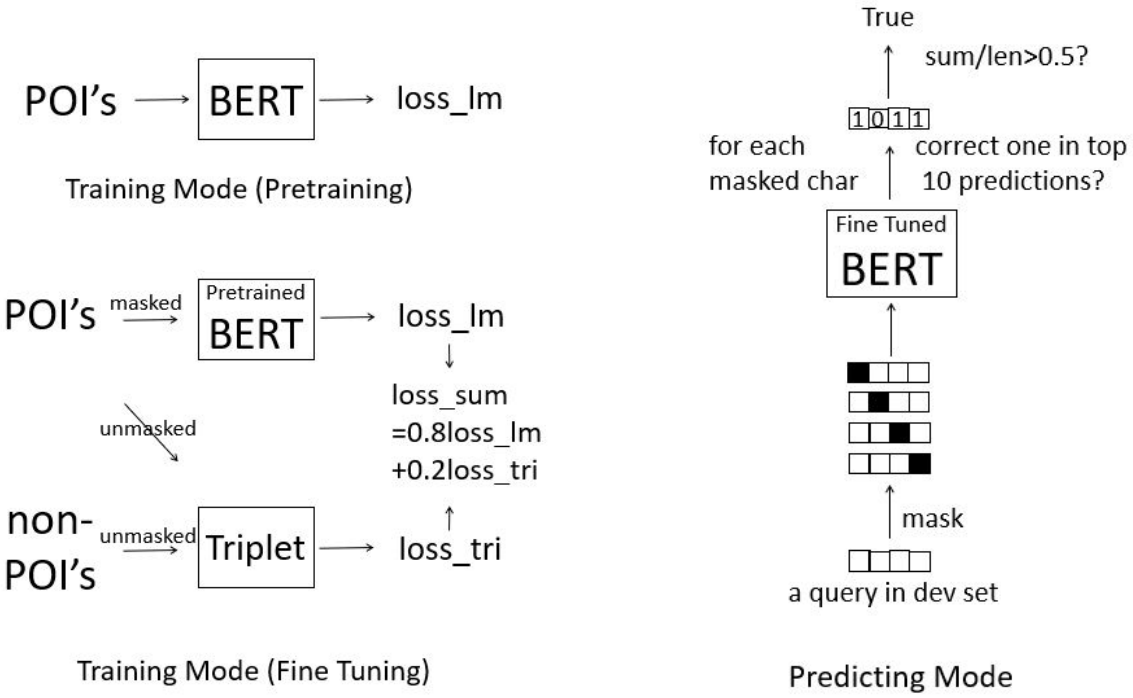
Hyperparameter Tuning

We tried a model of just four layers and a model of seven layers and found that they did not make much difference in our task, but we chose the 7-layer model with the attention and hidden layer dropout of 0.1. The learning rate was starting with 2.5e-6 and decreased using a cycle polynomial decay strategy with the power of 1.0 and end_learning_rate of 1e-6.

Code

Final code on which our results are based (run on GPU machine with 8 V100 cards) - Here.

Experimental code with Kaggle dataset (run on AWS EC2 instance p3.2xlarge GPU linux machine) -Here.



(loss_lm is the loss of the language model and loss_tri is the triplet loss)

5 Results and Discussion

The language model we trained from scratch using Chinese POIs perform very well and the fine tuned model using triplet loss performed a little better. Before we use triplet loss the language model misclassified 1671 non POIs in 9909 untrained non-POIs and 1105 POIs in 9740 untrained POIs, and the model fine tuned using triplet loss performs better in the non POIs and it only misclassified 1034 non POIs. The misclassified non POIs are very misleading, and the misclassified POIs normally contain no prefix and suffix. In the following table, "lm" is the language model without triplet loss and "lm+tri" is the language model with triplet loss.

	training F0.5	dev/test F0.5
lm	0.84	0.84
lm+tri	0.90	0.89

The accuracy for the language model (in the training set) reached 0.49 in the pretraining process, but it started off with only 0.32 and increased by 0.03 after we added the triplet loss in the dev set. Triplet loss is harder to train because of the combinatorial explosion problem. We would like to explore other loss functions to improve the word embedding accuracy beyond currently achieved.

We have tried to search the existing small POI dataset for the incoming POI to find the closest one, and we also tried to taste the incoming POI using the BERT's mouth to tell if our model has read something similar before for the big corpus. We use different strategies for different datasets because the time complexity is linear to the size of our corpus, so we didn't use the searching method for our big corpus. We grouped all types of non-POIs as one class and we thought it would be better if we use many groups as non-POIs. And the strategy we used for the large corpus to tell if the query/extracted POIs is a POI we just set a threshold as 0.5 and we used top 10 predicts for every character. We thought for characters with different locations and frequency should be treated differently.

For the smaller dataset [4], the time between encoding a search query for a single POI to returning the top 5 most similar results from the dataset takes ~ 3.6 seconds.

6 Conclusion

In this project, different methods to improve word embeddings were explored, starting with BERT as the base model. The language model was trained with the addition of triplet loss to classify between the POI and non-POI. This makes it easier to search for a user query and return results contained within the dataset. Further work to extend the model to return user query even when the POI is not present in the dataset would make the system very useful.

7 Contributions

Yanlin:

1. Explored and compared different possible methods to improve the embeddings for identification, including GAN for text [7], CipherGan [8], and Sentence-BERT [3]. The triplet structure in [3] was considered the most suitable for this task so was introduced to this project finally.

2. Worked on report writing, poster design.

Richa:

1. Worked with Kaggle dataset with location POI[4] starting with data preprocessing, tried fasttext embeddings [9], BERT as a service and finally triplet loss using Sentence-BERT [3]. Tried k-d tree, finally used KNN for semantic search to return POI query result for top 5 queries from pretrained Sentence-BERT model [3].

2. Worked on report/poster writing, topic/methodology discussions.

Le:

1. Implemented the BERT as a language model and added the triplet loss and trained the model with different parameters(different layers, lr, dropout and etc).

2. Implemented the training mode, eval mode and predict mode.

Reference

[1] Jacob Devlin and Ming-Wei Chang and Kenton Lee and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.
[2] Olivier Moindrot. 2018. Triplet Loss and Online Triplet Mining in TensorFlow. https://omoindrot.github.io/triplet-loss.
[3] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv: 1908.10084.
[4] Points of Interest POI Database. https://www.kaggle.com/ehallmar/points-of-interest-poi-database.
[5] Country-Currency Dataset. https://data.world/samayo/country-currency [column - 'currency_name'].
[6] Urban Dictionary Words And Definitions. https://www.kaggle.com/therohk/urban-dictionary-words-dataset [column - 'word']
[7] Akshay Budhkar and Krishnapriya Vishnubhotla and Safwan Hossain and Frank Rudzicz. 2019. Generative Adversarial Networks for text using word2vec intermediaries. arXiv: 1904.02293.
[8] Aidan N. Gomez and Sicong Huang and Ivan Zhang and Bryan M. Li and Muhammad Osama and Lukasz Kaiser. 2018. Unsupervised Cipher Cracking Using Discrete GANs. International Conference on Learning Representations.
[9] Bora Edizel, Aleksandra Piktus, Piotr Bojanowski, Rui Ferreira, Edouard Grave, Fabrizio Silvestri. 2019. Misspelling Oblivious Word Embeddings. arXiv:1905.09755
[10] BERT has a Mouth and must Speak, but it is not an MRF KyungHyun Cho.2019 https://sites.google.com/site/deepernn/home/blog/amistakeinwangchoberthasamouthanditmustspeakbertasamarkovrandomfieldlanguagemodel
[11] xu-song/bert-as-language-model https://github.com/xu-song/bert-as-language-model
[12] Alex Wang, Kyunghyun Cho.2019. BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model.arXiv:1902.04094v2
[13] Sean Welleck, Kiante Brantley, Hal Daume III, Kyunghyun Cho..2019. Non-Monotonic Sequential Text Generation.arxiv:1902.02192v3
[14] Elman Mansimov, Alex Wang, Kyunghyun Cho.2019. A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models.arXiv:1905.12790v1
[15] Florian Schroff, Dmitry Kalenichenko, James Philbin.2015. FaceNet: A Unified Embedding for Face Recognition and Clustering.arXiv:1503.03832v3