Deep Sparse Factorization for Fast Retrieval of Text Embedding

Sung-En Chang (NUID: 001423864)
Eva Sharma (NUID: 001266715)

1 Introduction

Embedding representation learning via neural networks is prevalent in modern similarity search problems. In the text retrieval task, a typical approach is generating an embedding vector from a neural network model that exploits the semantic relations between words or characters (e.g. ELMo), which is one of state-of-the-art models pre-trained on huge amounts of data for text embedding. However, computing the similarity between a query embedding and millions of embeddings in the database is computationally extremely expensive.

In the past, there has been a lot of studies on speeding up the similarity search of vectors, with Aritificial Neural Networks techniques such as KD-tree, Product Quantization, Hashing or Hierarchical Clustering. However, in high dimensions, ANN methods have a significant gap in accuracy compared to the exact Neural Network. A couple of recent works aim to address this issue through end-to-end training of neural networks to optimize the structure of embeddings to give better trade-off between accuracy and efficiency.

In this project, we observe that a sparsity ratio of p in the embedding could reduce number of multiplication in the similarity search to p^2 of the original. For example, a 1000-dimensional sparse embedding of 10 non-zeros could be 10000 times faster than 1000-dimensional dense embeddings, and 100 times faster than 10-dimensional dense embeddings. Therefore, we propose an end-to-end approach to learn sparse embeddings to facilitate fast retrievals of text strings.

The main contributions of this project are:

- Propose methodology to achieve efficient embedding representation using sparsification.
- Compare trade-off between accuracy and efficiency in case of dense and sparse embedding representation

In this report, we first go over the work that has been done before in this area and also introduce some literature on the methods we intend to use in this project. We describe the datasets we acquire along with the pre-processing steps we take to prepare the data for the experiments. We then formally define the problem statement and introduce the two tasks or problem setups we consider for this project. Next, we present the implementation details for each dense and sparse versions along with the evaluation metric. Finally we present the results for both the embedding representations and compare their performance on one of the datasets.

2 Related work

Most of the research on efficiently learning embedding representation using deep learning focuses on binarizing embeddings so that the similarity between the embeddings can be calculated faster using

Preprint. Work in progress.

the Hamming distance metric [1, 2, 3, 4]. These methods focus on learning the binary codes for hamming distance ranking and perform an exhaustive linear search over the entire dataset which is not likely to be suitable for large scale problems. Similarly, other works either minimize the difference between the similarity label and the cosine distance of network embedding or define a distance between a quantified data and continuous embedding, and back-propagates the metric loss error only with respect to the continuous embedding [5, 6]. Both these approaches require running k-means clustering on whole dataset which again is not practivcal for large scale problems due to computation complexity [6, 7]. Moreover, so far, 8-bit quantization has been a standard approach to represent embeddings. But reducing 8-bit to 1-bit could only give 8 times speedup, while also sacrificing accuracy significantly. Recently, instead of learning binary hamming code representation, Yeonwoo et al. tries to learn an embedding that is not only binary but also sparse [8]. However, it employs discontinuous objective function which as we know, is hard to optimize. We have also found its experimental setup to be faulty as it does not prohibit overfitting by copying class label.

In this project, for loss function, we employ Siamese networks. Siamese neural network is a class of neural network architectures that contain two or more identical subnetworks [9]. Here identical here means that they have the same configuration with the same parameters and weights and parameter updating is mirrored across both the subnetworks. Siamese neural networks are popular among tasks that involve finding the similarity or a relationship between two comparable things. Some examples are paraphrase scoring, where the inputs are two sentences and the output is a score of how similar they are; or signature verification [9, 10], where figure out whether two signatures are from the same person. Generally, in such tasks, two identical subnetworks are used to process the two inputs, and another module will take their outputs and produce the final output. Similarly, in Question Answering, some recent studies have used Siamese architectures to score relevance between a question and an answer candidate [11]. Thus, rather than letting the model learn to classify the inputs, the neural network learns to differentiate between two inputs through their embeddings. It will label the examples that are closer to each other as similar and the ones that are farther away as dissimilar. We use this property of the siamese network in our project to define the loss function for both the problem set ups.

We use Embedding from Language Models (*ELMo*) to get the word representations for both the problem tasks. ELMo is a deep contextualized word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy) [12]. These word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. They can be easily added to existing models and significantly improve the state of the art across a broad range of challenging NLP problems, including question answering, textual entailment and sentiment analysis. It's been shown to outperform GloVe and Word2Vec embeddings on a wide variety of NLP tasks. We use ELMo embedding in both the two tasks.

As mentioned above, the two tasks that we consider are Author Book Recommendation and Query Completion. Recommendation systems, in general, provides people with recommendations of products, that they will appreciate, based on their past preferences, history of purchase, and demographic information [13, 14, 15]. Recent most successful recommendation systems are location based, content based or make use of collaborative filtering [16]. Query completion is the problem of taking the prefix of a search query from a user and generating several candidate completions [17]. Po-Wei et al. uses unsupervised deep language models to complete and correct the queries given an arbitrary prefix [18, 19].

Sparsity often leads to efficient and interpretable representations for data. There have been efforts to propose ways to fine tune the word vectors in a classification setup while promoting sparsity making it more computationally cheaper than the traditional models [20, 21]. In this work, we use L1 Norm Regularization to induce the sparsity in the embedding representation [22].

3 Datasets

Towards the goal of our project, we explore two datasets appropriate for book recommendation and query completion problem. In this section, we will describe the two datasets and present some preliminary statistics for the same.

3.1 Book-Crossing Dataset

This dataset was gathered from BookCrossing (http://www.bookcrossing.com) [15]. The BookCrossing community caters for book lovers exchanging books all around the world and sharing their experiences with others. This dataset consists originally of three tables: BX-Users, BX-Books, BX-Book-Ratings. For this project, we have only used BX-Books which contains books related information.

We get the book ids rated by each unique user from BX-Book-Ratings table and mappings between book ids and book names from the BX-Books table. We get the title of the book and its author to gather information of the books publised by particular authors. We construct an adjacency matrix for books, where each entry in the list denotes whether both the two books were written by same author. A "0" entry denotes that the author who wrote one book is not the same as the one who wrote the other book, which means user reading one book will not be recommended the other book. Whereas, "1" entry denotes that both the books have the same author. For the experiment purposes, the entries with "1" are the positive samples and entries with "0" are the negative samples.

In our final dataset, we have $271,379$ unique books and $16,807$ unique authors. On average, we have 16.14 authors per book (positive samples). We show an example from our dataset in Table 2.

3.2 AOL Dataset

We used the 500k User Session Collection dataset which consists of 20M web queries collected from 650k users over three months. It provides real query log data that is based on real users which can be used for personalization, query reformulation or other types of search research. The dataset includes an anonymous user ID number, the time at which the query was submitted for search,if the user clicked on a search result, the rank of the item on which they clicked is listed and the clicked URL if the user clicked on the search result [23].

We use the complete query to create random prefixes of that query. This pair of completed query and its prefix is a positive sample and any other completed query and this same prefix is a negative sample. We constructed our dataset in a similar way. This gave us a total of $1,228,387$ prefixes and an average of 1.67 positive samples (Refer Table 1). We show an example from our dataset in Table 2.

Dataset	Total # Samples	Avg # Pos. Samples
BookCrossing Dataset	271,379	16.14
AOL Dataset	1,228,387	1.67

Table 1: Datasets used for the experiments

Dataset	Data Point	Positive Samples	Negative Samples
BookCrossing	Die Teufelin.	'East, West', 'Endlich Nichtraucher.'	'Rites of Passage'
AOL	food	food 20supplements, food 911	jeoffrey ballet,jeon ji-hyun

Table 2: Examples from Book Crossing and AOL dataset

4 Problem formulation

We formally define our problem statement as below. Consider a neural network denoted by the function S that maps an input data $x \in X$ onto a d dimensional sparse vector. $S(x, \theta) : X \to \mathbb{R}^d$ is a neural network transformation with respect to θ and takes input x and produces a d dimensional embedding vector. We add a regularization term to penalize $S(x)$ to convert it to a sparse vector. We then optimize an objective of the form

$$\min_{\theta} \; L(\{S(x_i, \theta)\}_{i=1}^{K}) + \lambda \sum_{i=1}^{K} \|S(x_i, \theta)\|_1 \tag{1}$$

where we use pairwise metric to define our loss function, the triplet loss. Triplet loss tries to bring close the current sample with the positive sample (A sample that is in theory similar with the current sample) as far as possible from the negative (A sample that is different from the current sample) [24]. We illustrate this concept in terms of figure 1

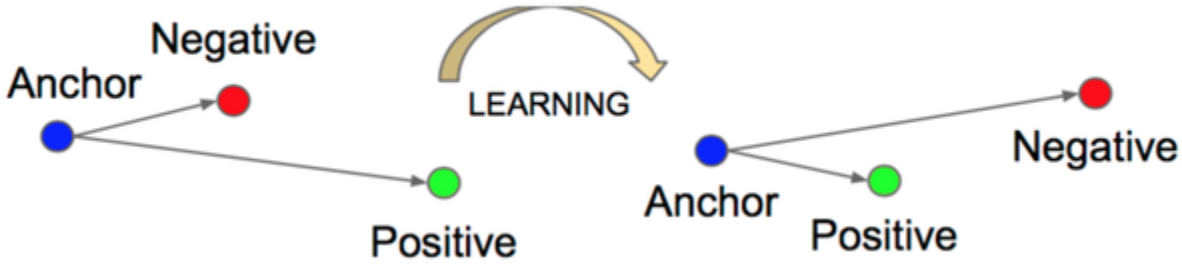

Figure 1: Illustrating learning using triplet loss function. Here Anchor is the current sample and, Positive and Negative are the samples closer to and farther from the current sample respectively.

Using this, in our work we define the loss function as

$$L = \sum_{i=1}^{K} \sum_{j_p \in P_i} \sum_{j_n \in N_i} w_{j_p} \max(\|S_i - S_{j_p}\|_2^2 - \|S_i - S_{j_n}\|_2^2, 0) \tag{2}$$

where i is the index of the data sample, and P_i, N_i are the sets of positive, negative indices of data sample i respectively. We require to have the distance between data sample and positive embedding vector less than that of data sample and negative one.

Now understanding this in terms of two aforementioned problem setups, we will consider book recommendation problem and query completion problem. In book recommendation problem, we will define the positive pairs of two books if both the books were written by sample author, whereas for the negative pair, both the books are not from the same author.

Similarly, in the case of query completion task, we split the sentence to prefix and its completed query. The positive pair is the prefix and its completed query, whereas the negative pair is prefix and the completed query of some other prefix. An alternative setup here can be to consider the frequency of the positive pair occurring in the dataset denoted by w_j.

5 Implementation Details

We only concentrate on the Book-crossing Dataset and learn the dense embedding representation as the baseline and compare it with its sparse version. In this section we list out the details of the model implementations for both dense and sparse versions. We further provide details of the evaluation methodology used.

General Setup We train both the dense and the sparse embedding representations in tensorflow using the triplet loss function implemented in `https://github.com/omoindrot/tensorflow-triplet-loss`. As described in problem formulation, we are using the triplet loss and implement it with online triplet mining in TensorFlow. For the datasets, we define the loss over triplets of embeddings using an anchor, a positive of the same class as the anchor and a negative of a different class.

As mentioned above, we are using online triplet mining which was first introduced in Facenet [24]. In this technique, we compute useful triplets on the fly, for each batch of inputs. Given a batch of B examples, we first compute the B embeddings and we then can find a maximum of B triplets. Most of these triplets will not be valid triplets. A triplet is valid if all the three embeddings are identical, and there are two positive samples and one negative sample. This technique is giving us more triplets for a single batch of inputs, and we don't need any offline mining. Thus this is more efficient also. In addition to this, we also have a module to calculate the similarity distance to get the distance between two embeddings.

Dense representation We first implement the dense version. We train this model using ELMo to learn the dense embeddings of size 1024 for 20 epochs. The learning rate of the model is $1e-4$ and batch size is 50. The triplet loss for this model trained with 10,000 datapoints is 0.0884 and with 100,000 is 0.0876. Similarly, the mean embedding norm value is 0.84 and 0.93 for model with 10,000 datapoints and 100,000 datapoints respectively.

Sparse representation We reuse the model parameters from above while adding a fully-connected layer with Relu the activation function along with adding L1 norm in the objective function to penalize the dense embeddings to become sparse. We add parameter lambda to decide how much percent of the elements in the embeddings are non-zero. The triplet loss for this model trained with 10,000 data points is 0.0874 and with 100,000 is 0.0911. Similarly, the mean embedding norm value is 0.95 and 0.90 for model with 10,000 data points and 100,000 data points respectively.

Evaluation We use Fast-KNN search algorithm for evaluation. KNN is a great tool for interpreting the outputs and not having to handle black boxes. The idea behind this clustering algorithm is to compare a new point to the K most similar points in the data set, and to give it the mainly represented label. The main computational complexity is to calculate all the distances between the current point and all the remaining points. For us, the query size is the value K is the query size(we consider query size = 500, 1000, 2500, 5000). For recall, we consider top 10 retrieved results for calculating precision and recall. The total number of label classes are $16,807$ and sparsity ratio achieved for the sparse embedding representation for model trained with 10,000 data points is and 0.509 and for 100,000 data points is 0.51069.

Query Size	Time (ms)		Precision		Recall	
	Dense	Sparse	Dense	Sparse	Dense	Sparse
500	5.160	1.61	0.0396	0.0398	0.174	0.172
1000	10.09	3.33	0.0467	0.0464	0.194	0.190
2500	25.32	8.78	0.0466	0.04583	0.181	0.178
5000	50.05	16.15	0.0488	0.0484	0.183	0.183

Query Size	Time (ms)		Precision		Recall	
	Dense	Sparse	Dense	Sparse	Dense	Sparse
500	51.01	14.32	0.1024	0.102	0.316	0.306
1000	102.65	33.23	0.0952	0.0945	0.313	0.307
2500	254.84	77.97	0.0881	0.0864	0.296	0.288
5000	510.83	152.17	0.0886	0.0874	0.2884	0.283

Table 3: Evaluation result for 100,000 data points from Book-crossing dataset.

6 Results

In this section, we present the evaluation results for both dense and sparse embedding representations for the book-crossing dataset. For our work, we experimented with 10,000 data points and 100,000 data points from the complete book-crossing dataset. In Table 5 and 3 we present the evaluation results for both dense and sparse representations using different query sizes for 10,000 data points and 100,000 datapoints respectively. Notice that in both the tables, there is a considerable drop in total time taken to complete the evaluation from the dense to sparse representations. We also note that, the recall and precision are slightly higher in case of dense than that of sparse representation. These observations show the that with the increase in efficiency achieved, there is a respective drop (although small) in precision and recall from dense to sparse representation which supports our methodology's assumption.

7 Conclusion

Through our experiments and results, we put forth a methodology to achieve a better trade-off between speed and accuracy in the retrieval task by sparsifying the embedding representation. In future, we intend to run our experiments on other similar applications, specially on the AOL dataset for query completion problem as well as further tune the hyper-parameters for learning the embeddings.

References

[1] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2475–2483, 2015.

[2] Mohammad Norouzi, David J Fleet, and Ruslan R Salakhutdinov. Hamming distance metric learning. In *Advances in neural information processing systems*, pages 1061–1069, 2012.

[3] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016.

[4] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1556–1564, 2015.

[5] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. Deep quantization network for efficient image retrieval. In *AAAI*, pages 3457–3463, 2016.

[6] Shicong Liu and Hongtao Lu. Learning deep representations with diode loss for quantization-based similarity search. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 2282–2289. IEEE, 2017.

[7] Anna Choromanska, Alekh Agarwal, and John Langford. Extreme multi class classification. In *NIPS Workshop: eXtreme Classification, submitted*, volume 1, pages 2–1, 2013.

[8] Yeonwoo Jeong and Hyun Oh Song. Efficient end-to-end learning for quantizable representations. *arXiv preprint arXiv:1805.05809*, 2018.

[9] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[10] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *null*, pages 1735–1742. IEEE, 2006.

[11] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*, 2015.

[12] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[13] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296. ACM, 2010.

[14] Souvik Debnath, Niloy Ganguly, and Pabitra Mitra. Feature weighting in content based recommendation system using social network analysis. In *Proceedings of the 17th international conference on World Wide Web*, pages 1041–1042. ACM, 2008.

[15] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.

[16] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[17] Konstantinos Boulis. Personalized query completion suggestion, September 27 2011. US Patent 8,027,964.

[18] Aaron Jaech and Mari Ostendorf. Personalized language model for query auto-completion. *arXiv preprint arXiv:1804.09661*, 2018.

[19] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. Learning user reformulation behavior for query auto-completion. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 445–454. ACM, 2014.

[20] Prasanna Sattigeri and Jayaraman J Thiagarajan. Sparsifying word representations for deep unordered sentence modeling. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 206–214, 2016.

[21] Md Rashadul Hasan Rakib, Magdalena Jankowska, Norbert Zeh, and Evangelos Milios. Improving short text clustering by similarity matrix sparsification. In *Proceedings of the ACM Symposium on Document Engineering 2018*, page 50. ACM, 2018.

[22] David L Donoho. For most large underdetermined systems of linear equations the minimal 1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(6):797–829, 2006.

[23] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *InfoScale*, volume 152, page 1, 2006.

[24] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.