Semantic Similarity in the Medical Domain

Exposé of Master Thesis
Bonn University & Fraunhofer IAIS

Abbas Goher

April 23, 2018
St. Augustin

The goal of this thesis is to use a neural network model to map patient records to semantically similar and potentially helpful research work. For training the model large volume of raw texts from PubMed [2] will be used.

The traditional approach for finding relevant documents for a given query is to count repetitions of query terms in the documents. Different weight schemes for these counts lead to a variety of TF-IDF ranking features. However, the basic forms of such a ranking system only consider query terms, under the assumption that non-query terms are less useful for document ranking. This makes such ranking systems incapable of capturing the deep semantic meaning of the text all by itself.

Neural network models have recently shown impressive results in capturing the underlying semantics of the documents[1]. The main stumbling block in creating such a semantic model is the lack of freely available data, PubMed mostly provides abstracts of research articles for free.

The main idea of this thesis is to explore ways in which deep learning can be used for solving the problem of finding semantically similar/helpful research articles given a patient record. The chosen direction of the research is to apply Doc2Vec model proposed in [7] and explore different ways in which the Doc2Vec model can be extended/improved.

Background & Literature Review

Embedding Models

Usage of deep learning in the context of NLP tasks requires representing the text as the input for neural networks. As of late, the most used and powerful representations are one of the following embeddings: Word2Vec [8] and GloVe [12], which are word level embeddings. An extension to Word2Vec known as Doc2Vec was proposed in [7].

We look closer at the Word2Vec and Doc2Vec models, as they are the most relevant to this work. Word2Vec model uses distributed vector representation of words, a well-known framework for learning word vectors as shown in the Figure 1. The task is to learn to predict a word given other words in the context. More formally, given a sequence of training words $w_1, w_2, w_3, ..., w_T$, the objective of the word vector model is to maximize the average log probability

$$\frac{1}{T}\sum_{t=K}^{T-K} \log p(w_t \mid w_{t-1}, ..., w_{t+1}) \tag{1}$$

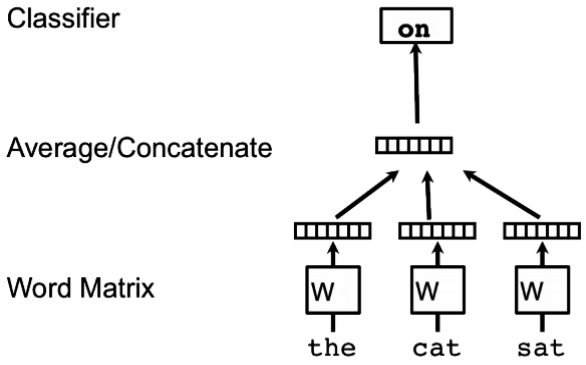

Figure 1: A framework for learning word vectors. Context of three words (the, cat, and sat) is used to predict the fourth word (on). The input words are mapped to columns of the matrix W to predict the output word.

Some commendable efforts have been made to go beyond word level representations [10] [17] [16] [5] [9]. A simple approach is to use the weighted average of all the words in the document. But weighted averaging of word vectors loses the word order. A more sophisticated approach is combining the word vectors in an order given by a parse tree of a sentence, using matrix-vector operations [15]. A drawback of such an approach is that it only works

on sentences as it relies on parse trees.

Doc2Vec is capable of constructing representations of input sequences of variable length. Unlike some of the previous approaches, it is general and applicable to texts of any length: sentences, paragraphs, and documents. In Doc2Vec framework (see Figure 2), every document is mapped to a unique vector, and every word is also mapped to a unique vector. The document vector and word vectors are averaged or concatenated to predict the next word in a context. The only difference to a Word2Vec model is the additional document token.It acts as a memory that remembers what is missing from the current context or the topic of the document. The document vectors and word vectors are trained using stochastic gradient descent and the gradient is obtained via back-propagation.

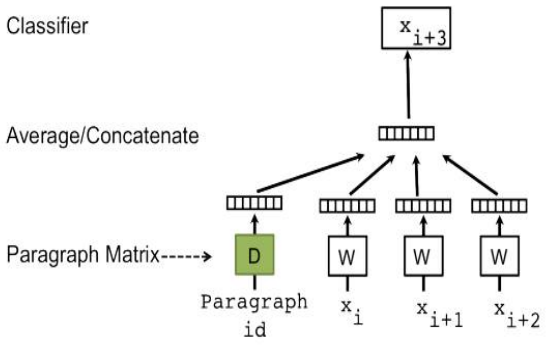

Figure 2: A framework for learning paragraph vector. This framework is similar to the framework presented in Figure 1; the only change is the additional paragraph token that is mapped to a vector via matrix D. In this model, the concatenation or average of this vector with a context of three words is used to predict the fourth word. The paragraph vector represents the missing information from the current context and can act as a memory of the topic of the paragraph.

Embedding Model Extension

Classical emitting methods such as Doc2Vec and Word2Vec are generally unsupervised requiring no domain information and as such have broad applicability. However, for highly specified domains with a moderately sized corpus, classical methods fail to find meaningful semantic relationships. [3] introduce a vocabulary driven Word2Vec method known as Dis2Vec which is used to generate disease-specific word embeddings from unstructured health-related news corpus. The input corpus D consists of a collection of word

context pairs. Based on the vocabulary V, we can categorize the word context pairs into three types as shown below:

- $D(d) = (w, c) : w \in Vc \in V$, i.e. both the word w and the context c are in V
- $D(\rightharpoonup d) = (w, c) : w \notin Vc \notin V$, i.e. neither the word w nor the context c are in V
- $D(\rightharpoonup d) = (w, c) : w \in V \oplus c \in V$, i.e. either the word w is in V or the context c is in V but both cannot be in V

Counter-fitting Word Vectors

One drawback of learning word embeddings from co-occurrence information in corpora is that such methods will generally fail to tell synonyms from antonyms (Mohammad et al., 2008). For example, words like east and west or expensive and cheaper appear in near-identical contexts, which means that distributional models produce very similar word vectors for such words. [11] proposed a novel counter-fitting method which injects antonymy and synonymy constraints into vector space representations in order to circumvent this issue. 1 shows the results [11] achieved using their counter-fitting technique.

	east	expensive	British
Before	west	pricey	American
	north	cheaper	Australian
	south	costly	Britain
	southeast	overpriced	European
	northeast	inexpensive	England
After	eastward	costly	Brits
	eastern	pricy	London
	easterly	overpriced	BBC
	-	pricey	UK
	-	afford	Britain

Table 1: Nearest neighbours for target words using GloVe vectors before and after counter-fitting

Planned work

The work is planned to consist of the following stages:

- Use the gensim [13] implementation of the Doc2Vec model as described in ([7]), and train it on the following
  - Full Wikipedia articles
  - Wikipedia info-boxes
  - A smaller subset of PubMed only containing only colorectal cancer abstracts
  - Full PubMed abstracts
  - PMC another Subset of PubMed which contains full articles
- Manual preprocessing
  - PubMed uses special language/notation to describe different age-groups. i.e elderly, middle-aged, >60 years old and 50-60 years old. Similar language/notation will be used while generating patient descriptions from a given database.
  - Extending the query document by adding rephrases of the key parts. For example, The query document "75 years old patient has rectal cancer " is extended to " Elderly patient, >70 years old, The patient has rectal cancer, cancer in rectum ". The query document is generated in a structured way to allow an easy extension. The basic structure of the query document is "Age, Gender, Type of cancer, Type of metastasis".
  - Instead of generating the full patient record, only the following information is recorded
    * The type of cancer
    * Patients age
    * Patients gender
    * Primary tumor location
    * Type of surgery to be performed
- Use different variants of Doc2Vec like DBOW, DM, and DMC to see which performs the best.
- Extend the Dis2Vec [3] model to work on document level.

- Use different distance metrics such as [14] [6] and [4] to find out which distance metric produces the best results.
- Compare the final results against the vanilla Doc2Vec model.

Technical Description

Implementation

The following tools are to be used in the project

- Gensim library [13] implementation will be used for the vanilla Doc2vec model and its variants.
- Pythons nltk library will be used for tokenization and stop-word removal
- Python library regex will be used for parsing the full PubMed abstracts
- To reduce the training time some elements will be implemented in Cython. Cython file will then be used to generate a .c file for faster computation
- GCC 4.9 compiler will be used to compile the .c file generated by Cython

Datasets

For the basic research on the quality of the network, the following datasets can be used:

- Wikipedia articles corpus
- Wikipedia info-box corpus

For the medical domain, the following can be used:

- Pubmed abstracts subset corpus on colorectal cancer
- Full set of Pubmed abstracts
- PMC, another subset of PubMed which contains full articles

Validation

Wikipedia and Info-box corpus

Following steps describe the technique to be used as a proof of method on Full Wikipedia corpus and Wikipedia info-box corpus.

- Take a random target article and use the part of its information to infer a vector. Infer-vector is a method to obtain an embedding for the out-of-sample documents. Inference starts with a low-magnitude random vector, that is then incrementally adjusted to be more predictive.
- Use this inferred vector to find 10 similar documents.
- The prediction task is to predict the article the given information belongs to.

Initial experiments revealed an interesting correlation between the size of the document and the number of iterations (a.k.a steps) used to infer a vector.

PubMed corpus

For the medical domain, we first infer a vector for a patient description and then find PubMed abstracts which are semantically closer to the given patient record. For result evaluation, we have 400 labeled abstracts. For each of these 400 abstracts, we have 10 other abstracts which have been labeled to be semantically close.

For a given query document the task is to predict semantically similar documents.

Evaluation Metric

Precision and recall are single-value metrics based on the whole list of documents returned by the system. For systems that return a ranked sequence of documents, it is desirable to also consider the order in which the returned documents are presented. Average precision (AP) is thus define as:

$$AP = \frac{\sum_{k=1}^{n}(P(k) \times rel(k))}{\text{number of relevant documents}} \tag{2}$$

where k is the rank in the sequence of retrieved documents, n is the number of retrieved documents, P(k) is the precision at cut-off k in the list

and rel(k) is an indicator function equaling 1 if the item at rank k is a relevant document, zero otherwise.

For multiple queries, an extension of average precision known as mean average precision is used. Mean average precision for a set of queries is the mean of the average precision scores for each query. In this thesis mean average precision (MAP) will be used as an evaluation metric.

References

[1] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. *Neural Probabilistic Language Models*, pages 137–186. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[2] N. C. for Biotechnology Information. Pubmed. `https://www.ncbi.nlm.nih.gov/pubmed`.

[3] S. Ghosh, P. Chakraborty, E. Cohn, J. S. Brownstein, and N. Ramakrishnan. Characterizing diseases from unstructured text: A vocabulary driven word2vec approach. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1129–1138. ACM, 2016.

[4] Github. Annoy. `https://github.com/spotify/annoy`.

[5] E. Grefenstette, G. Dinu, Y.-Z. Zhang, M. Sadrzadeh, and M. Baroni. Multi-step regression learning for compositional distributional semantics. *arXiv preprint arXiv:1301.6939*, 2013.

[6] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966, 2015.

[7] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[10] J. Mitchell and M. Lapata. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429, 2010.

[11] N. Mrkšić, D. Ó Séaghdha, B. Thomson, M. Gašić, L. Rojas-Barahona, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young. Counter-fitting word vectors to linguistic constraints. In *Proceedings of HLT-NAACL*, 2016.

[12] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[13] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[14] G. Sidorov, A. Gelbukh, H. Gómez-Adorno, and D. Pinto. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, 18(3):491–504, 2014.

[15] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.

[16] A. Yessenalina and C. Cardie. Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 172–182. Association for Computational Linguistics, 2011.

[17] F. M. Zanzotto, I. Korkontzelos, F. Fallucchi, and S. Manandhar. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1263–1271. Association for Computational Linguistics, 2010.