Assignment 3

Due on May 16, 2019 (23:59:59)

Click here to accept your Assignment 3

Introduction

Word embeddings are used to represent a word in a low-dimensional space. It is a vectorial representation that bears any semantics or syntax regarding a word. Neural word embeddings have been quite popular in the last years, where the journey is believed to begin with word2vec [4]. Word embeddings have been used in almost every field of Natural Language Processing applications and it has been shown to boost the performance in NLP applications such as text classification [3], sentiment analysis [8], and machine translation [7].

Following the neural word representations[1], the representation of any type of textual data has been also investigated in the last couple of years. Some examples to those representation learning methods are doc2vec (for document representation [2]), tweet2vec (Tweet representation [1]), sent2vec (sentence representation [6]), etc.

In this assignment, you will use word and document embbedings for two tasks respectively: anlogy prediction and document classification.

0.1 Task 1: Analogy Experiments

Embeddings are generally learned on a large corpus. Once the embeddings are trained, how can we measure the quality of learned embeddings? How good an embedding represents the word's features? One of the methods to evaluate an embedding's quality is the analogy. Analogy is question to find the relationships between words. For example, *man is to woman, what king is to* _____. Here, the answer is *queen*. An illustration is given in Figure 1. Therefore in this task, you will perform some analogy experiments[2].

In the analogy prediction task, for a given a pair of words $< a, b >$ and a third word c, choose a fourth word d so that the analogy is built so that a is to b as c is to d holds. In other words, the relationship between c and d should be as close as possible to that of between a and b.

Mikolov et al. [4] proposed some simple algebraic operations to apply for embeddings to find an analogy between words. Let v_a be the vector of a and v_b the vector of b. For

[1] Here, the words *embedding*, *representation*, and *vector* will be used interchangeably.

[2] This assignment is adapted from
https://courses.cs.washington.edu/courses/cse490u/17wi/assignments/assignment2.pdf/

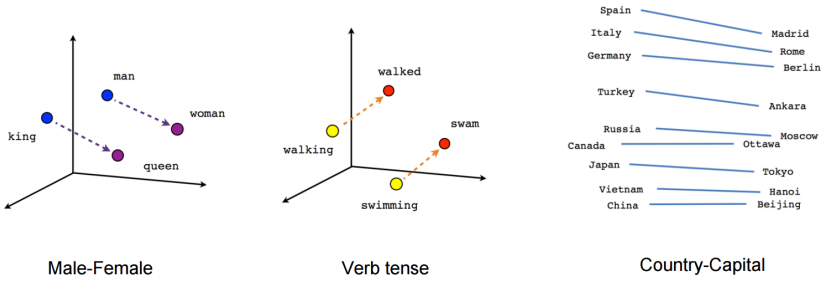

Figure 1: Word vectors in the low dimensional space.

another word d with a vector v_d we expect the following analogy:

$$v_b - v_a \approx v_d - v_c \tag{1}$$

For example, $v_{woman} - v_{man} \approx v_{queen} - v_{king}$. We therefore seek:

$$d = \underset{d \in V - \{a,b,c\}}{argmax}\ cos(v_d, v_b - v_a + v_c) \tag{2}$$

which means you will seek the word which has a vector closest to $v_b - v_a + v_c$, that will be v_d. Note that the given words a, b, and c are excluded from consideration. The cosine similarity is often used as a similarity metric between nonzero vectors. So, $cos(v_a, v_b)$ gives how similar/close the two vectors are.

0.1.1 Analogy Dataset

You will be given a subset of Mikolovs [4] analogy dataset [4], which includes four semantic relations and four syntactic relations. In the test files, each line represents one analogy question, in the form of four words $< a, b, c, d >$. For example:

Bangkok Thailand Cairo Egypt

A question is counted as correctly answered only if the predicted word is the same as the given word. For example, given the first three words "Bangkok Thailand Cairo", the task is to predict "Egypt". Note that this relationship is semantic and gives the *capital of* relation between the two words.

The full set of analogy questions can be found in the file word-test.v1.txt in the assignment tarball. The groups of relations are delimited by lines starting with a colon (:) and you will only work with these relations: capital-world, currency, city-in-state, family, gram1-adjective-to-adverb, gram2-opposite, gram3-comparative, and gram6-nationality-adjective.

0.1.2 Word Embeddings

You will use pretrained word embeddings so that you will not need to train your embeddings in this task. For this assignment, use will use word2vec trained embeddings [4, 5] `https://code.google.com/archive/p/word2vec/`.

0.1.3 Evaluation

Once you perform the analogy prediction on the given dataset, you will measure the accuracy of your prediction on the eight analogy tasks. Accuracy is the ratio of the correctly answered questions to the total number of questions:

$$A(W) = \frac{\#of\_correctly\_answered\_questions}{\#of\_questions\_attempted} \tag{3}$$

0.2 Task 2: Document Classification

Distributional semantics can represent also a larger chunk of text that consists of words, such as phrases, sentences and documents. In this task, you will build document representations based on doc2vec [2]. Doc2vec is an unsupervised algorithm to generate vectors for sentence/paragraphs/documents that represent a larger text in a low dimensional space similar to word embeddings. The algorithm is an adaptation of word2vec [4]. Once you learn the document embeddings, you will train a classifier to group a set of documents based on their topic.

You will train doc2vec model using *Gensim*[3] in *Python* and train Logistic Regression classifier using a library. You are free to use any library for the classifier, but we recommend sckitlearn.

0.2.1 Dataset

For the document classification task, you will classify a given list of movies based on their topic. The movie dataset contains short movie plots. Each movie plot is associated with a label that represents its genre. There are six genres in the dataset:

1. Sci-fi
2. Action
3. Comedy
4. Fantasy

[3] https://radimrehurek.com/gensim/models/doc2vec.html

5. Animation
6. Romance

The data has 2448 rows. You will use the first 2000 rows for training and use the rest for testing purposes.

0.2.2 Evaluation

Once you train your classifier on the training data, and predict the genres of the movies in the dataset, you will measure the accuracy of your document classifier. The accuracy is defined as the ratio of the correctly classified documents to the total number of documents:

$$A(W) = \frac{\#of\_correctly\_classified\_documents}{\#of\_documents} \tag{4}$$

Submit

You are required to submit all your code. You will implement the assignment in **Python** (Python 3.5). You will submit a report in latex format template). The codes you will submit should be well commented. Your report should be self-contained and should contain a brief overview of the problem and the details of your implemented solution. Give the answers of all questions raised in the definition of the assignment above. You can include pseudocode or figures to highlight or clarify certain aspects of your solution.

- report.pdf
- code/ (directory containing all your codes as Python file .py)

CHALLENGE (For Bonus Points)

In addition to the default tasks defined above, you can improve your document classifier better to compete with other participants in the class. The accuracy of your classifier is the main criteria to be compared with others. The participants in the class that obtain a higher tagging accuracy will deserve more points.

To enter the competition, you have to register kaggle with your department email account. The webpage of the challenge will be announced later. Top 5 assignments will earn extra points (10 points).

Grading

- Code (90 points) : Task 1: 40 points, Task 2 : 50 points (Doc2vec : 25, Classifier: 25)
- Report (10 points)

Note: Preparing a good report is important as well as the correctness of your solutions! You should write your results for each part of the task and answer of the questions related with parts.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

References

[1] Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481*, 2016.

[2] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[3] Qian Liu, Heyan Huang, Yang Gao, Xiaochi Wei, Yuxin Tian, and Luyang Liu. Task-oriented word embedding for text classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2023–2032, 2018.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[5] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[6] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017.

[7] Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Janani Padmanabhan, and Graham Neubig. When and why are pre-trained word embeddings useful for neural machine translation? *arXiv preprint arXiv:1804.06323*, 2018.

[8] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.