Text Based Information Retrieval (B-KUL-H02C8A)
Using Semantic Embeddings in Information Retrieval
Assignment 2015-2016

Susana Zoghbi, Guillem Collell and Marie-Francine Moens

For the 4 study points: Choose paper assignment OR programming assignment.
The assignment is due April 29, 2016.
Graded for 1/3 of points.

For the 6 study points: There is only a programming assignment.
The first part of the assignment is due April 29, 2016, the second part May 23, 2016.
Graded for 1/3 of points.

Programming assignments for both groups may be conducted in teams of 1 to 3 people. Paper assignments are individual.

1 Introduction

Recently, there has been a surge of work proposing to represent words as dense real-valued vectors, derived using various training methods inspired from neural-network language modeling. These representations, referred to as *neural embeddings*, *word embeddings* or *semantic embeddings*, have been shown to perform well in a variety of natural language processing and information retrieval tasks.

2 Paper Assignment Description

A paper of approximately 6 pages can be sent by e-mail to: `mailto:gcollell@kuleuven.be` and `mailto:susana.zoghbi@cs.kuleuven.be`.

The paper should be organized as a short survey of current research in the induction of semantic embeddings, but with the focus on the questions listed below. The paper targets only semantic embeddings induced solely from text. Besides briefly touching upon the current state-of-the-art research, the paper will have to provide answers to 2 out of 3 questions listed here (you may choose what to focus on in your paper):

1. An obvious problem of word embeddings is that vector representations do not account for ambiguity. In other words, a single word can have many different meanings or parts of speech and nevertheless it is always represented as a single vector. For example "run" can be either a noun or a verb, and "jaguar" might refer to an animal or a car. What approaches have been develped to deal with this problem?

2. Word embedding models such as skip-gram and CBOW have hyperparameters that are to be

chosen by the user and will affect task performance. Two important hyperparameters are window size and dimensionality of the word vectors. What is the effect of increasing the window size on tasks such as word similarity and word analogy? For what sort of language tasks it is more appropriate the use of a large window size? For what tasks a small window size is enough? For what tasks prefer large vector representations? For what we can get away with a rather small dimensionality (say, 25)? Provide examples from papers and discuss them.

3. What are the improvements of the new deep learning models for word representations (such as [1, 2]) over the old ones that just capture word counts such as latent Dirichlet allocation (LDA)? In what aspects and language tasks they made a big difference?

It is important that you include for each question:

- Description of the problem
- Description of the solution(s), illustrate with examples
- Critical discussion of the solution(s) (refer to the material we have seen in the course, give your personal opinion)
- Conclusions
- References to literature, URLs,...

3 Programming Assignment Description

Source code (in C++, Java, MatLab, Python, ...) can be sent by e-mail to: `mailto:sien.moens@cs.kuleuven.be`, `mailto:gcollell@kuleuven.be`, and `mailto:susana.zoghbi@cs.kuleuven.be`.

Important:

- Add commentary
- Add description of how to run your software
- Add your test examples
- If asked in the assignment, add documentation with the actual results and comparisons of different models

3.1 PART I

As mentioned before, the first part of the programming assignment may be chosen (instead of the writing assignment) by students following the 4-study-point version of the TBIR course, while it is obligatory for students following the 6-study-point version of the TBIR course (and the 6pt students will have to also complete the second part of the programming assignment which will be posted online later in the semester).

The first part of the programming assignment consists of three related tasks.

3.1.1 Warm-Up Task: Solving Analogies using Semantic Embeddings

It is important to get familiar with the embeddings that will be later used in a retrieval scenario.

In the first task, you will have to implement a small analogy solver that tries to guess the correct word given a simple analogy question as follows

$$a : b = c :?$$
$$Bratislava : Slovakia = Bishkek :?$$
$$ate : eat = found :?$$

Semantic embeddings have proven to be very useful for this task of solving various semantic and syntactic analogies and they outscore previously established computational alternatives. The embedding-based models may provide a correct answer even when humans have problems doing so (e.g., many people actually do not know what country Bishkek is the capital of; the correct answer is Kyrgysztan).

Once you obtain the vector representations (embeddings) for all words, building a model to solve the analogy task is very straightforward: given the analogy $a : b = c :?$, and word vectors $\vec{a}$, $\vec{b}$, and $\vec{c}$: (1) compute the vector $\vec{c} - \vec{a} + \vec{b}$, (2) find the closest word vector $\vec{d}$ from the vocabulary to the vector $\vec{c} - \vec{a} + \vec{b}$ using the cosine similarity. You may find additional analogy models here:
http://www.marekrei.com/blog/linguistic-regularities-word-representations/

Besides the model briefly explained here, you are free to experiment with other analogy models described in that blog post.

Datasets. *Where to get the analogy dataset?* You will work with the Google analogy dataset, available here:
http://word2vec.googlecode.com/svn/trunk/questions-words.txt

The dataset is in a very simple format, containing 4 words in each line. You have to prepare the analogy question yourself by removing the fourth word in each line and then the task is to guess the correct word in the analogy (where the removed word will act as your single ground truth answer). The file with the analogy questions also contains line starting with ':' that denote the nature of the semantic/syntactic relation coded in the corresponding block of questions, so do not forget to remove these lines.

Each analogy model that you construct should report its performance as a simple *Recall*@1 metric, which measures the proportion of correct guesses made by the system out of the total number of analogy questions.

OK, but where can I find the word vectors? A nice practice is that people often post the pre-trained word representations online, so you won't have to train the embedding learning models yourself, but will be able to use the pre-trained representations off-the-shelf. In this part of the programming assignment, you will experiment with two representation models:

(1) Word2Vec (Skip-gram and CBOW) [1, 2] - these vectors are available here:
https://code.google.com/p/word2vec/ (under the section "Pre-trained word and phrase vectors").
Note that the vectors are available in a binary file, so you might want to convert it to a plain text file. A nice shortcut to convert the file from bin to txt is presented here:
http://stackoverflow.com/questions/27324292/convert-word2vec-bin-file-to-text

(2) GloVe [3] - these vectors are available here:
http://nlp.stanford.edu/projects/glove/
These vectors are stored in a plain text file, and vectors of different dimensionality are available (50, 100, 200 and 300), use only the vectors pre-trained on Wikipedia.

Task and Questions. (10 points) Your task will be to run several analogy solving models with several different representations on the benchmarking analogy dataset and report your findings. Focus on the following questions:
1. Is the choice of the analogy model important? Which representations work better with which analogy models?
2. Is dimensionality of the representation important when using GloVe vectors?
3. What is the computational complexity of the analogy models given the pre-trained vectors?
4. What are the typical errors?

3.2 PART II (only for 4 study points)

3.2.1 Retrieval Task: From Sentences to Images

Now that you are familiar with word embeddings from the warm-up task, we may actually use the embeddings in a real-life retrieval scenario.

We will use a subset of the data provided for the ImageCLEF challenge http://imageclef.org/2016/annotation. The data that you will work with correspond to $2,000$ images. Each image has been annotated with 5 to 51 textual descriptions (mean: 9.492, median: 8).

The data files are available here:
`https://people.cs.kuleuven.be/~susana.zoghbi/tbir_data/data_4stdpt.tar.gz`

We refer you to the README file accompanying the dataset for more info about the dataset structure and content.

Since the dataset contains sentence descriptions aligned to actual images, we may simulate a simple retrieval model that will retrieve images using purely textual queries. However, in this retrieval scenario we won't use the advantage of having the alignments to induce cross-modal semantic spaces, and we won't need the image representations at all, as the whole retrieval scenario will still be text-based.

There are two main files that you will work with:

a. target_collection: 17,784 documents (one document per line).
b. queries_val: 1,000 queries (one query per line) to be issued against the target collection. Each query is annotated with its corresponding image id, so you have ground truth.

The retrieval process is as follows:
1. Issue each query sentence, and rank all the sentences in the target collection according to their similarity to the query sentence.
2. Retrieve the ranked list of images based on the ranking of the target collection.
3. Evaluate your results.

You should report the results using *mean average precision* (MAP) with a cutoff of 1000 dcouments, and *Precision vs. Recall curves.*

Task and Questions.
(i) **(25 points)** As the first model, you may implement a basic yet quite powerful *simple unigram language model* that treats each word in the query sentence independently and treats test sentences as documents. See here for more info on the unigram language model for retrieval:
http://nlp.stanford.edu/IR-book/pdf/12lmodel.pdf

Report MAP and Precision vs. Recall curve using this model.

(ii) **(25 points)** Following that, try to include some semantics and learn structured representations for the query and each target sentence. You may rely on the word embeddings from the warm-up task (Word2Vec or GloVe embeddings). We will use a very simple compositional model for sentences relying on the additive vector-based model from [4]. The vector representation/embedding $\vec{s}$ for each sentence s is computed as follows:

$$\vec{s} = \vec{w_1} + \vec{w_2} + ... + \vec{w_{ns}} \tag{1}$$

where $\vec{w_1}$,...,$\vec{w_{ns}}$ are vectors of words constituting the sentence, and ns is the length of the sentence. In case you don't have a pre-trained vector for some words in the sentences, you may just neglect such words.

The similarity between two sentences may then be computed again on their embeddings using simple cosine similarity, and the ranked list may be obtained by ranking the target sentences according to the computed similarity scores.

Report MAP and Precision vs. Recall curve using this model. Can you think of a more intelligent way on how to compute the representations for sentences in the embedding space besides the simple additive model? What seems to be the major problem with the additive model? How do results change if you change the aggregation heuristic?

Choose your own method (25 points).
From the retrieval methods you have seen in class, you may choose one that you would like to try out for this task. Present the method you chose and report your results.

Further Evaluations (15 points).
On the day that you present your project, you will be given a new set of 200 queries. You will use your best model to evaluate its retrieval performance for these unseen sentences. Additinally, the team with the best MAP score (with cutoff of 1000 documents) will receive 15 extra points on the project.

To submit
- A report on each of the items above answering all the questions.
- Your working code.

3.3 PART II (only for the 6 study points)

3.3.1 Retrieval Task: From Sentences to Images

Now that you are familiar with word embeddings from the warm-up task, we may actually use the embeddings in a real-life retrieval scenario. In paticular, your assignment for this task will be to solve the Teaser 1 (Text illustration) from the 5th edition of the ImageCLEF challenge http://imageclef.org/2016/annotation. Since its start (in 2010), the aim of ImageCLEF competition has been the use of web data to improve image annotation. Thus, tasks such as automatic image description, localization of different objects in an image and generating scene descriptions are of interest in the ImageCLEF context. Creative ideas are welcome!

In the Teaser 1 task you will have to retrieve the correct image given a text. The ground-truth for the training and testing data sets is assumed to be the actual image already associated with a document.

The data for this task is available at
`https://people.cs.kuleuven.be/~susana.zoghbi/tbir_data/data_6stdpt.tar.gz`.

Note: the size of this file is 33Gb, as it contains a very large set of images.

The training data consists of 300,000 image-webpage pairs. In addition, a development set of 3,000 image-webpage pairs is also provided for parameter-tuning.

There exist a held-out test set of 200,000 image-webpage pairs. At test time, for a given text, you will have to choose the best illustration among the 200,000 available images.

Please refer to the readme file for more details.

All images have been pre-processed using a 16-layer Convolutional Neural Network (CNN), whose output for each image is 4096-dimensional vector corresponding to the activations of the relu7 layer of Oxford, extracted using the Berkeley Caffe library. More details can be found at https://github.com/BVLC/caffe/wiki/Model-Zoo. There is no need for you to do any image processing. Optionally, you may get acquainted with image feature extraction. Namely, to represent an image with a (usually high-dimensional) vector. Each component of such vectors can be interpreted as a "visual property." You can find a nice explaination of how to do it here http://www.marekrei.com/blog/transforming-images-to-feature-vectors/.

Training: In this task the idea is to create a cross-modal space where we can bridge the semantic gap between images and text. Given a textual snippet, you want to retrieve images that best illustrate the text. To accomplish this, there is a variety of methods that may be used. Here we suggest a few to get you started, but we would like for you to be creative.

1. Canonical Correlation Analysis
2. Latent Dirichlet Allocation (LDA) on concatenated image-text pairs.
3. Bilingual Latent Dirichlet Allocation (BiLDA) on aligned image-text pairs.
4. Cross-modal semantic embeddings.

Throughout the course you will be learning more about these methods and you can start trying them out.

Evaluation metric: The performance of your answers will be judged with the so-called Recall at the k-th position ($R@K$). $R@K$ is nothing but the proportion of queries for which the correct response was among the first k ranked results. Thus, in general, the larger the k, the larger the recall. For a more detailed explaination of this metric you might want to refer to [5]. In order to stick to the basis of the comptetition, the value of k will be that of ImageCLEF (which will be announced in http://imageclef.org/2016/annotation a few days before the submission deadline).

Teams with the best performance on the held-out 200,000 documents, will receive bonus points and may author a submission to the ImageCLEF challenge. So be creative!

To submit
- A report detailing your approach for this cross-modal retrieval task and your results.
- Your working code.

References

[1] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 27th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, pp. 3111–3119, 2013.

[3] 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)*, pp. 1532–1543, 2014.

[4] J. Mitchell and M. Lapata, "Vector-based models of semantic composition," in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 236–244, 2008.

[5] M. Hodosh, P. Young, and J. Hockenmaier, "Framing image description as a ranking task: Data, models and evaluation metrics," *Journal of Artificial Intelligence Research*, vol. 47, pp. 853–899, 2013.