Text Based Information Retrieval (B-KUL-H02C8A)
Using Semantic Embeddings in Information Retrieval
Assignment 2014-2015

Ivan Vulić, Susana Zoghbi, and Marie-Francine Moens

For the 4 study points: Choose paper assignment <u>OR</u> programming assignment.
The assignment is due April 30, 2011.
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 30, 2015, the second part May 22, 2015.
Graded for 1/3 of points.

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:ivan.vulic@cs.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. What is the role of context in all these models? Which models rely only on local contexts (local contexts refer to only a small set of neighboring words given the pivot word) and which models rely on global contexts (i.e., contexts are defined at the document level)? What are the properties of models relying on local window-based contexts, and what are the properties of models relying on global document-based contexts? Is there a way to consolidate the two approaches into a unified framework?

2. Since the majority of work on text-only semantic embeddings focuses on single words, how to build embeddings beyond the level of words, and represent phrases, sentences or documents as real-valued dense vectors, that is, semantic embeddings?

3. How to exploit semantic embeddings in information retrieval models? Why are such semantically-aware structured text representations important in retrieval tasks?

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:ivan.vulic@cs.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 assignent 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

(You will start working on this task already as part of the exercise session on semantic embeddings in the week of March 23rd. 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

$$\begin{aligned} a : b &= c :? \\ \mathit{Bratislava} : \mathit{Slovakia} &= \mathit{Bishkek} :? \\ \mathit{ate} : \mathit{eat} &= \mathit{found} :? \end{aligned}$$

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:
https://code.google.com/p/word2vec/source/browse/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) [Mikolov et al.2013a, Mikolov et al.2013b] - 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 [Pennington et al.2014] - 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. 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?

3.1.2 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 the Flickr8K dataset that contains $8,000$ Flickr images annotated with short human-generated captions (5 captions/descriptions for each image). The set of images is available here:
http://nlp.cs.illinois.edu/HockenmaierGroup/Framing_Image_Description/Flickr8k_Dataset.zip
Moreover, textual descriptions of images are available here:
http://nlp.cs.illinois.edu/HockenmaierGroup/Framing_Image_Description/Flickr8k_text.zip
We refer you to the README file accompanying the dataset for more info about the dataset structure and content. All experiments in this part, as well as in subsequent parts of the assignment will be conducted only on the test subset (TEST) of the whole corpus containing $1,000$ images with their descriptions (see the dataset). Other two subsets of the dataset are $1,000$ images used as the development set (DEV), and the rest is used for training (TRAIN), but you will not need the other 2 datasets (for now).

Since the Flickr8K 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.

Since each image in TEST has 5 sentence descriptions associated with it, we will simulate the retrieval process as follows:
1. For each test image, randomly sample one sentence that will serve as a query sentence, and remove it from the list of descriptions (now, you have 1 sentence acting as a query and 4 sentences associated with each test image). By this procedure you immediately have the ground truth established, as you know the image associated with the query sentence from the original test set.
2. Issue the query sentence, and rank all the remaining sentences in your test data according to their similarity to the query sentence.
3. Retrieve the ranked list of images based on the ranking of sentence descriptions from the test dataset.

You should report the results using *mean reciprocal rank* (MRR) or *Recall@N* (R@N) (where *Recall@N* actually measures the proportion of test queries for which the retrieval system returned the single correct image in top N candidates, where $N = 1, 5, 10, 20, ...$).

Task and Questions. (i) 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 MRR and R@1 scores using this model. Note that you may provide the ranking of images by aggreggating the results for sentences in different ways, e.g., you may take the best scoring sentence associated with each image and neglect all the other lower-scoring sentences for that particular image, or you may aggregate all the results and rerank images based on the average score for each sentence associated with the image. Is there some other aggreggation heuristic besides the two mentioned here? How do results change if you change the aggreggation heuristic in the step that replaces sentences with the actual images?

(ii) 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 [Mitchell and Lapata2008]. 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 MRR and R@1 scores 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 aggreggation heuristic?

BONUS QUESTIONS. (for extra points) (iii) Now, try to combine the two models (i.e., the simple unigram language model and the embedding-based IR model) into a combined model using a simple combination process as follows:
1. Compute ranked lists using single models first.
2. Normalize the ranked lists to the interval [0,1]
3. Combine the scores from the two ranked lists using linear interpolation with parameter λ. Each score is computed as:

$$score_{comb}(s_i, s_j) = \lambda \cdot score_{unigram}(s_i, s_j) + (1 - \lambda) \cdot score_{embeddings}(s_i, s_j) \tag{2}$$

4. Build a new combined ranked list according to the combined scores from the previous step. Experiment with three different λ values: $\lambda = 0.3$ (more weight given to the embedding-based IR model), $\lambda = 0.5$ (equal weights), $\lambda = 0.7$ (more weight given to the simple unigram model).

Report MRR and R@1 scores using this combined model. Do you see any difference compared to the scores obtained by the single models? What seems to cause the difference? How does the combined model behave with respect to parameter λ?

(iv) Now, try to perform the retrieval over the entire Flickr8K dataset (TRAIN+DEV+TEST) using the same set of queries as before (one sentence for each TEST image sampled in the previous step). What can you expect if you expand your search space, that is, your target collection? Report the difference in results.

PART II (TRAILER). Moving from these simple retrieval and similarity models (which still do not exploit the inherent multimodality and training data) towards the models that exploit a joint multi-modal vision-language space for cross-modal retrieval (e.g., again retrieving relevant images given query sentences) will be the subject of the extension to this programming assignment in Part II (for 6 study points). You may already consult the literature and get acquianted with the ideas from recent state-of-the-art papers on cross-modal information retrieval [Frome et al.2013, Karpathy et al.2014]. The extension will again rely on word embeddings from the warm-up task, as well as the Flickr8K dataset needed for the sentences-to-images retrieval task. Visual features will be also coded as vector representations (i.e., visual embeddings), and will be provided to you.

3.2 PART II

To be continued...

References

[Frome et al.2013] Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M., and Mikolov, T. (2013). Devise: A deep visual-semantic embedding model. In *Proceedings of the 27th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, pp. 2121–2129.

[Karpathy and Fei-Fei2014] Karpathy, A. and Fei-Fei, L. (2014). Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306.

[Karpathy et al.2014] Karpathy, A., Joulin, A., and Fei-Fei, L. (2014). Deep fragment embeddings for bidirectional image sentence mapping. *CoRR*, abs/1406.5679.

[Kiros et al.2014] Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2014). Multimodal neural language models. In *Proceedings of the 31th International Conference on Machine Learning (ICML)*, pp. 595–603.

[Mikolov et al.2013a] Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[Mikolov et al.2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). 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.

[Mitchell and Lapata2008] Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 236–244.

[Pennington et al.2014] Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

[Socher et al.2014] Socher, R., Karpathy, A., Le, Q. V., Manning, C. D., and Ng, A. Y. (2014). Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2:207–218.