Katholieke
Universiteit
Leuven

Department of
Computer Science

Using Semantic Embeddings in Information Retrieval

Assignment 2015-2016

Text Based Information Retrieval (B-KUL-H02C8A)

Jakub Macina (r0641432)
Tao Chen (r0606207)
Xueying Deng (r0601458)

Academic year 2015–2016

Contents

1 Warm-Up Task: Solving Analogies using Semantic Embeddings

Table 1: Results of using word2vec pre-trained word vectors [1].

Number of base vectors	Model	Question type	R@1
30000	Addition	Gram 1- adjective - to adverb	0.3044
		Gram2-opposite	0.3411
		Gram3-comparative	0.9174
		Gram4-superlative	0.7005
		Gram5-present-participle	0.7335
		Gram6-nationality-adjective	0.8524
		Gram7-past-tense	0.6603
		Gram8-plural	0.7741
		Gram9-plural-verbs	0.6195
		Overall	0.6559
60000	Addition	Gram 1- adjective - to adverb	0.3085
		Gram2-opposite	0.4409
		Gram3-comparative	0.9174
		Gram4-superlative	0.8405
		Gram5-present-participle	0.7841
		Gram6-nationality-adjective	0.9037
		Gram7-past-tense	0.6596
		Gram8-plural	0.8544
		Gram9-plural-verbs	0.6494
		Overall	0.6722
60000	Direction	Gram 1- adjective - to adverb	0.0373
		Gram2-opposite	0.0837
		Gram3-comparative	0.5676
		Gram4-superlative	0.4082
		Gram5-present-participle	0.2812
		Gram6-nationality-adjective	0.8480
		Gram7-past-tense	0.1981
		Gram8-plural	0.2170
		Gram9-plural-verbs	0.2483
		Overall	0.3210
60000	Multiplication	Gram 1- adjective - to adverb	0.3427
		Gram2-opposite	0.4397
		Gram3-comparative	0.9182
		Gram4-superlative	0.8806
		Gram5-present-participle	0.8087
		Gram6-nationality-adjective	0.8999
		Gram7-past-tense	0.7091
		Gram8-plural	0.8649
		Gram9-plural-verbs	0.7011
		Overall	0.7294

Table 1 and 2 show the results of warm up part if solving analogies using word embeddings. In the following paragraphs, the discussion is based on these results.

Table 2: Results of Glove pre-trained word vectors [2].

Number of base vectors	Model	Dimension of vectors	R@1
30000	Addition	50	0.4738
		100	0.6371
		300	0.7132
30000	Direction	50	0.3112
		100	0.4083
		300	0.5272
30000	Multiplication	50	0.3662
		100	0.5932
		300	0.7199

1.1 Is the choice of the analogy model important? Which representation work better with which analogy models?

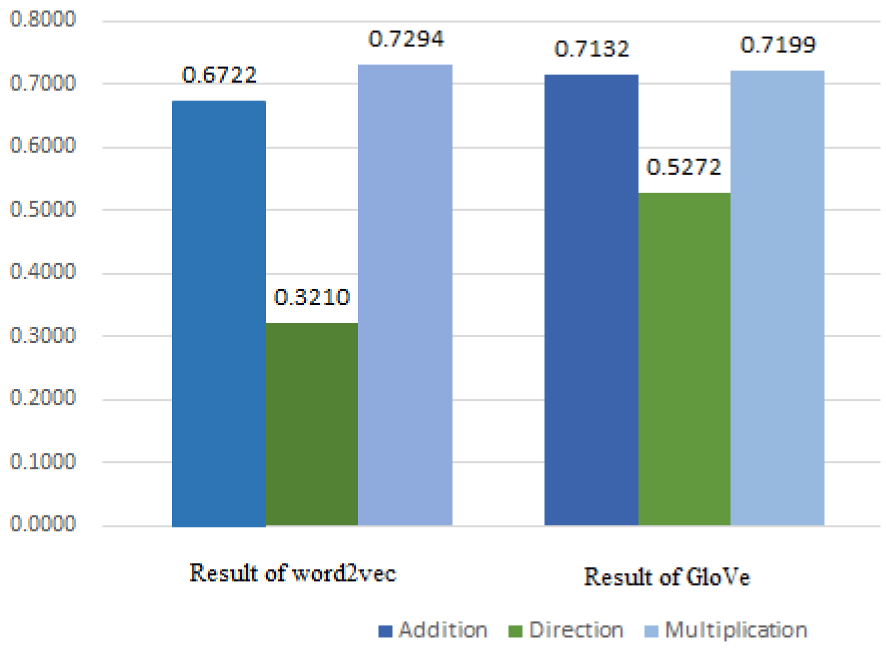

Figure 1: Comparison of analogy models for word2vec and glove vectors.

Yes, the analogy models are quite important. According to Figure 1 we can find that,the results of two representation ways are similar. Direction model performed worst in both representations, the results of addition model and multiplication model are almost equal with the recall@1 around 0.7. Problem with the direction model is that this model only take into account direction of the step and ignore the magnitude of the step.

1.2 Is dimensionality of the representation important when using GloVe vectors?

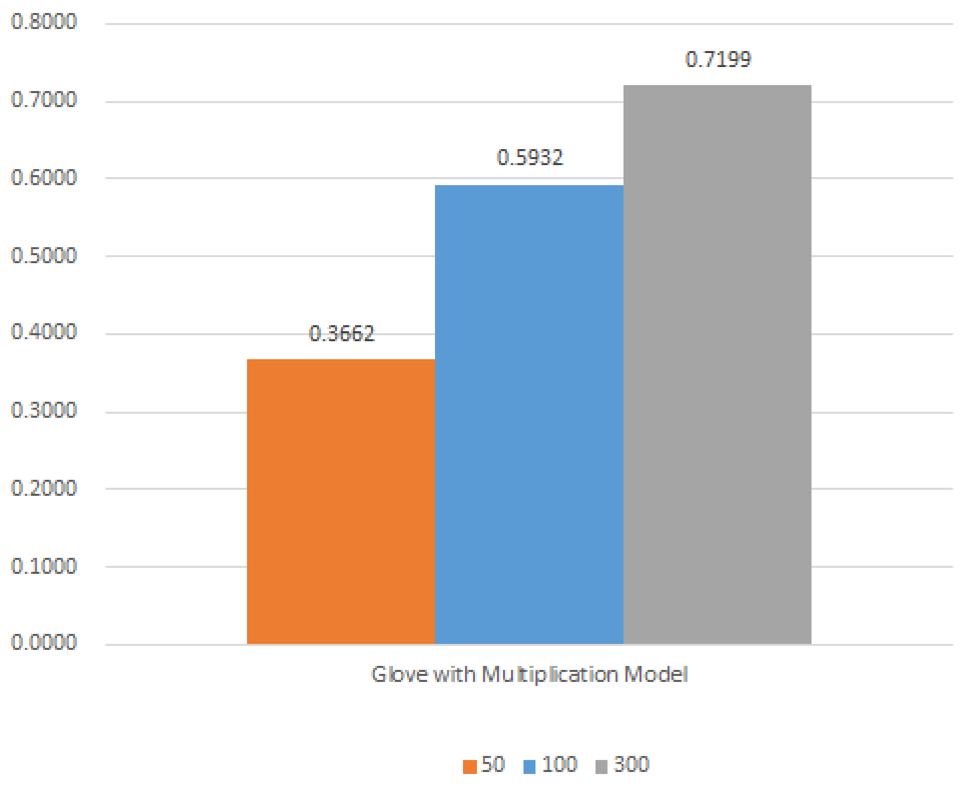

Figure 2: Different models of Glove vectors with different dimensions.

Figure 2 shows the comparison of GloVe representation with different dimensions when using multiplication model. From the figure, we can clearly find that with higher dimension representation, we can get higher recall. Thus we conclude that, the dimension of a representation highly influence the recall. Rationale for this result is that computing cosine similarity for vectors with higher dimensionality can better distinguish the similar words, thus the recall is higher.

1.3 What is the computational complexity of the analogy models given the pre-trained vectors?

The computational complexity of analogy models given the pre-trained vectors are: O(M(3N + D + DN)) $\approx$ O(MDN), where M is the number of analogies/queries we need to solve, D is the dimensionality of pre-trained vectors (300 for word2vec, 50, 100 and 300 for GloVe), and N is the number of pre-trained vectors (word corpus). Taking the "addition" method for example, for each analogy, it takes 3N computations to get the vectors of the first three words in the analogy, D computations to get predicted vector ($\vec{c}$-$\vec{a}$+$\vec{b}$), and $D \times N$ computations to get the best vector in the vocabulary (D computations for computing the cosine similarity for each vector of N vectors). The three different methods, addition, direction and multiplication have the same overall computational complexity, O(MDN), though there are slight differences when computing the vector representations.

1.4 What are the typical errors?

We analyzed results for word2vec vector representation with multiplication analogy model with limit of 60 000 most frequent words. Our findings are presented below with R@1 for every category.

Model performed well for capital countries, nationalities (0.89), family (0.81), comparative (0.91), superlative (0.88), present participle (0.80), past tense (0.70) and plurals (0.86) questions. However, it performed bad on currency (0.24) questions, where the typical error was predicting the currency symbol instead of the currency name. One of the reason is that the word2vec vector representation is trained on the corpus from Google News and in the articles currency sign is more commonly used. Not very good results was obtained for adjective-to-adverb questions (0.34), where one of the reasons might be the limitation to 60 000 most frequent words and some of the adverbs could be less frequent. Moreover, synonyms to ground truth were frequently predicted, e.g. sometimes-infrequently, finely-precisely or occasional-infrequently.

Other common error is that the most similar word predicted is one of the three words that are an input. It is also essential to lowercase the input and the predicted word. These types of error is prevented in our implementation.

In conclusion, number and type of errors is dependent on the training corpus and context size. When searching for the best analogies, it depends on the constraint on the number of frequent words. From the results, one can notice that news training corpus is not suitable for adjective to adverbs task, while it is suitable for predicting nationality and comparative questions.

2 Part II Retrieval Task: From Sentences to Images

2.1 LDA model

The most important thing for achieving retrieval from sentence is solving the gap between text representation and image features. Text is represented as a bag of the words with weights. Images were preprocessed by convolutional neural network and resulting image features are in the form of a vector, where each component of the vector represents some visual property. Our approach is based on Latent Dirichlet Allocation (LDA) on concatenated image-text pairs. In the following part, the detail of idea of LDA will be discussed.

2.2 Creating dictionary

In order to process the words by a model, preprocessing is needed. At first, we build dictionary of all words that appear in the training textual data corpus. Dictionary contains mapping between words and their corresponding ids and vice versa. Using words ids has advantage in reducing the memory complexity.

Each word is converted to lowercase and stemmed by a Porter stemmer. To delete words with an apostrophe, tokenization was used.

In the last step we filter out stop words and words that appear only once in the corpus, because these words are not useful for the LDA model. As image features can be seen as a words, these words in form of $feature_{id}$ were also added to the dictionary. After this preprocessing, dictionary contains 285 324 words.

2.3 LDA training

In the next step, for each textual input, the corresponding image features are found and concatenated together, as shown in figure 3. Textual features are preprocessed as bag-of-words by the same method as described in previous section. Since there are at most 100 words with highest weights, and the sum of these weights are at most 10000, we normalized the weights by 10000. Then the weights of textual words are all between 0 and 1. For the corresponding image, top 100 features with highest weights are chosen and normalized to unit vector, since we want to balance the significances of textual (word) features and visual (image) features in the concatenated vector.

In the last step, preprocessed textual features and image features are concatenated into one vector with unit length. Concatenated vectors might have various length because number of words in training examples varies. To reduce the memory complexity, sparse matrix representation of concatenated features is used. Concatenated feature has form $(word_{id}, weight)$, where $word_{id}$ is id of the word from the dictionary and $weight$ is a weight of the word/image feature.

We used LDA implementation from Gensim library [3]. Right number of topics and Dirichlet priors was estimated empirically by the performance on the validation data set. Training corpus for LDA is not loaded into memory and concept of lazy loading is implemented, where only data that are needed for computation are loaded from disc and stored in the memory. Lazy loading approach is used in our implementation to reduce the memory requirements.

2.4 Multimodal retrieval: from text to images

After training of the LDA model, previously unseen images or text representations can be presented to the model for inference of topic distribution $\theta = P(z|d)$ using Gibbs sampling. Each topic is then represented by textual topic-word distribution $\phi = P(w^{text}|z)$ and image topic-word distribution $\phi = P(w^{img}|z)$.
For a retrieval scenario, for every 200 011 testing images that can be used for prediction, image topic distribution $\theta^{img} = P(z|d^{img})$ by trained LDA model inference was computed and stored on the disc. After that, given a textual query d^{text_query} one by a time from a total of 189 888 testing queries, textual query topic distribution $\theta^{text\_query} = P(z|d^{text\_query})$ for each query was estimated the same way using LDA inference. For a given query topic distribution, similar images are found by cosine similarity with images topic distribution:

$$sim(d^{text\_query}, d^{img}) = cos(\theta) = \frac{P(z|d^{text\_query}) \bullet P(z|d^{img})}{\| P(z|d^{text\_query}) \| \| P(z|d^{img}) \|} \quad (1)$$

Cosine similarity was chosen as a ranking method because it's computation is fast and it can be used efficiently by using matrices multiplication. We also experimented with Hellinger distance metric, which is suitable for comparing the similarity between probability distribution. However, cosine similarity outperforms Hellinger distance on the validation set, therefore we used cosine similarity in the final approach.

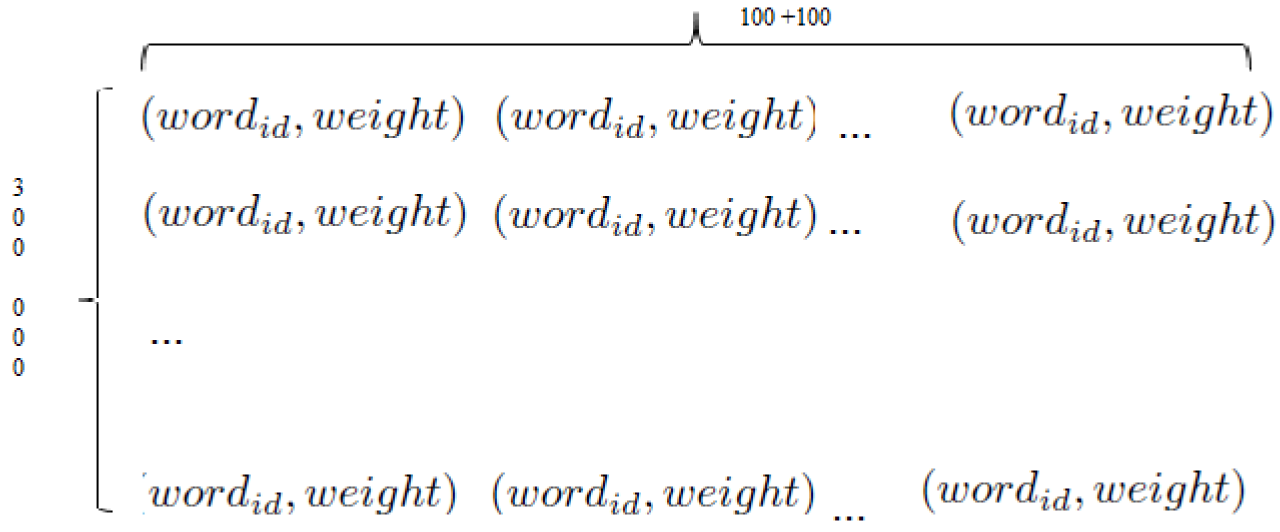

Figure 3: Visualization of concatenated textual image pairs.

2.5 Considered alternatives

Alternatives for dictionary: Currently by using bag-of-words model we built dictionary for the input to the LDA model. Several other alternatives for dictionary creation were considered.

- `Using bag-of-words`: Currently, the method we are using is the improvement of bag-of-words. First immature idea of using bag-of-words is calculating all unique words in the textual training examples, which is around 800 000. Then create a 800 000 dimensions vector for every document in the textual training set.

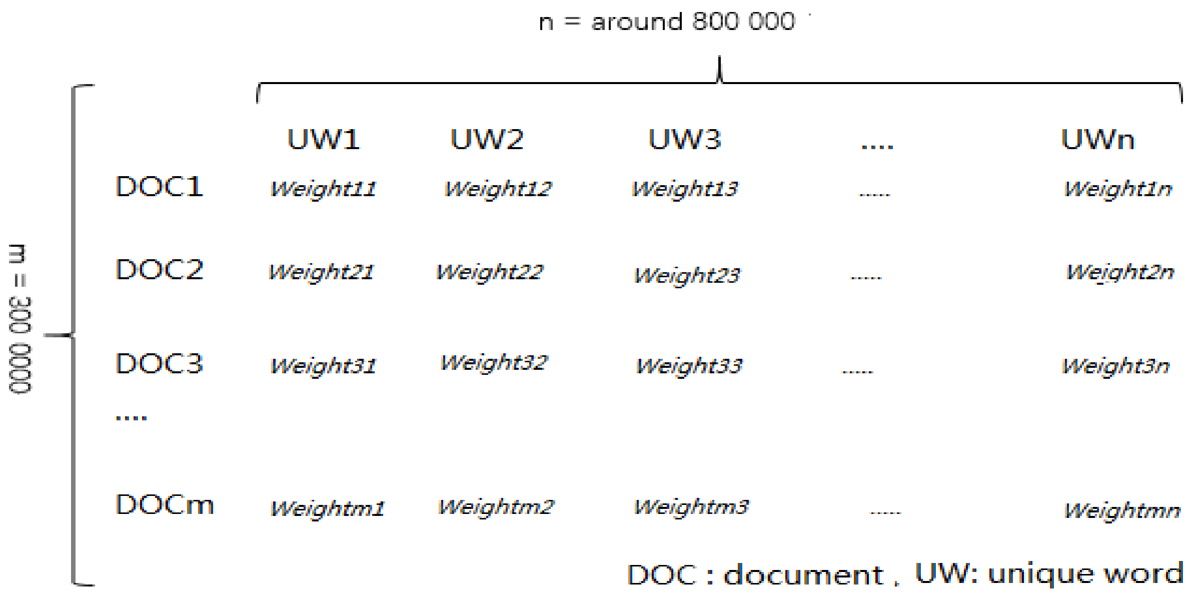

Figure 4: The alternative matrix representation of bag-of-words from full dictionary for all training examples. The drawback of this method is consuming too much memory, in order to store this matrix needs around 22.3 * 4 GB, which is impossible for processing.

- `Using word2vec` : By using inference of the vectors binary document of word2vec from warming up, we try to transfer every word to the 300 dimensions vectors and multiply the vector with the weight of the word in the document and sum all vector of words together to represent the document.

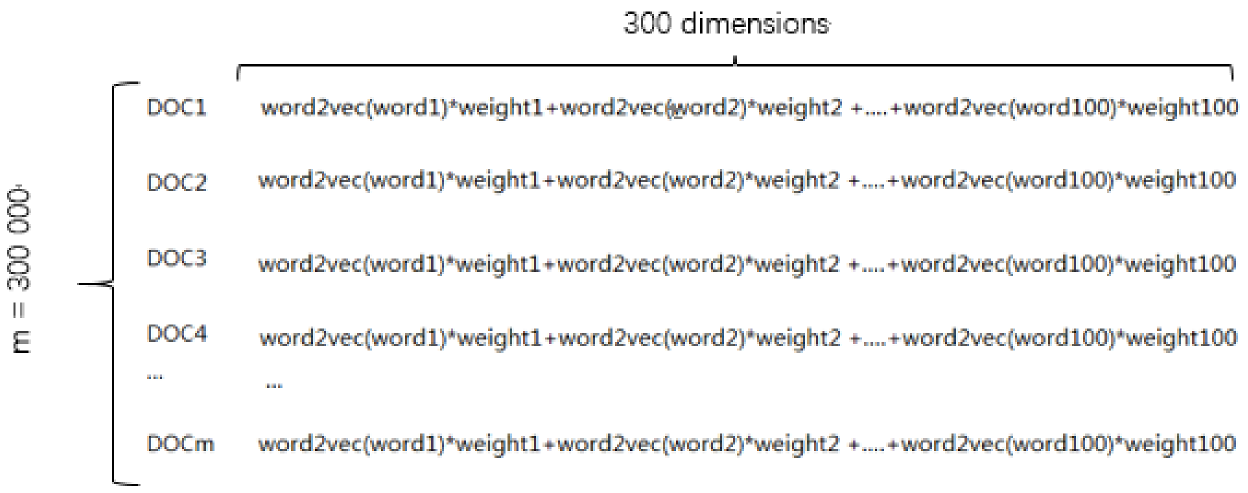
300 dimensions
m = 300 000
DOC1
DOC2
DOC3
DOC4
...
DOCm
word2vec(word1)*weight1+word2vec(word2)*weight2 +....+word2vec(word100)*weight100
...

Figure 5: The alternative of using word2vec vectors for training.

Figure 5 shows idea of the matrix of using word2vec. However, not all words are presented in the vectors corpus. Becayse the raw text of website for this task is not available, it is not possible to train vector model by word2vec. Furthermore, it might take much more time and effort then the retrieval task, therefore this alternative is not prospective.

2.6 Evaluation

- Number of topics: At the time when are going to train the model of LDA, we were wondering how many topics we should set for training. According to the size of our training data and number of our features, model with more than 500 topics seems most reasonable. However, it is very time and memory consuming to train model with large number of topics. Machine with at least 25GB memory might be needed for the training. Thus we trained the LDA model on 50, 150 and 200 topics.

  According to the result on the 3337 validation examples presented in table 3, there are no significant difference between these results which is mainly due to our training data requires much more topics to learn more detail to distinguish between two similar images, only by using 50, 150 or 200 topics is very hard to find slightly individual differences between the images.

  There exist a method called HDP (Hierarchical Dirichlet Process) LDA [4] which automatically determine a suitable number of topics. It is possible to use it in the future work.

- Reduce usage of memory: Struggle with the memory is what we did most of the project, then we *sacrifice time complexity to reduce the space complexity* which makes the program able to run even on less powerful machine. The idea is we increasing the frequency of hard disc I/O, in order to reduce the usage of memory.

Table 3: Results on validation set for different number of LDA topics.

Number of topics	Similarity measure	Constraint to top 100 image features	R@100
50	cosine	yes	0.0734
50	cosine	no	0.0761
50	Hellinger	yes	0.0261
150	cosine	no	**0.0791**
200	cosine	no	0.0725

2.7 Conclusion

At the beginning of the project, we tried a lot to save the memory in order to load all data into the memory and figure out how to use all files that we have. It consumed too much time and therefore no time was left for implementing another model for comparison, e.g. CCA or cross model embedding.

About CCA model, actually we have already held the whole idea how to implement it. First, construct vectors for all text documents by using sparse vector, use text matrix and image matrix to get u and v value. Third, transform query vector and all possible image vectors in to same space by using u and v value and then calculate their similarity. We did not implement this model is due to lack of time, but it is a possibility for a future work.

To conclude, our approach show promising results. We designed and implemented the pipeline from textual representation to image representation space. Our results can be further improved by implementing proper BiLDA with sampling from distinct topic-word distribution for text and topic-word distribution for image features.

For workload in total, i.e. time for searching materials, configuring IDE and server, implementing logic and testing the code took us more then 200 hours. However, we tried a lot and learnt about the field of semantic embeddings and probabilistic topic models.

References

[1] word2vec project , available at `https://code.google.com/archive/p/word2vec/`.

[2] Global Vectors for Word Representation, available at `http://nlp.stanford.edu/projects/glove/`.

[3] Gensim library, available at `http://radimrehurek.com/gensim/`.

[4] Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2012). Hierarchical dirichlet processes. Journal of the american statistical association.