Semantic Embeddings in IR

Text-Based Information Retrieval

Joren Verspeelt, Jochem Geussens, Vincent Tanghe

April 24, 2016

1 Introduction

In the field of Natural Language Processing (NLP) and Information Retrieval (IR), there is a large need for good presentations of text documents. Recently, a lot of researchers in the field are presenting dense word representations (also known as Word Embeddings, Neural Embeddings or Sementic Embeddings). For the Course of Text-Based Information Retrieval, we are given the opportunity to work with state-of-the-art algorithms. The goal of this assignment is to gain practical experience with these algorithms by comparing them and using them in an application. First we will implement a small analogy solver that guesses related words given a simple analogy. Then, we will discuss our implementation for a search engine to find pictures based on their description.

Contents

2 Part I (Warm-up)

2.1 Task description

In this part, we discuss our analogy solver that tries to guess the correct word, given simple analogy questions. E.g.:

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

Then, we will run several analogy solving models with several different representations on the benchmarking analogy dataset.

2.2 Setup

We compare the use of different word embeddings (GloVe and Word2Vec) towards different analogy models. To test the models, we used the Google's questions. [reference to link here]
We coded our solution in Python (version 2.7) using IPython Notebook. All experiments were run on a notebook with 8GB RAM memory, InteL(R) Core$\hat{\text{T}}$M i7-3610QM CPU, SSD hard disk and Windows 7 Operating System.
To run the different analogy models, a function per analogy model was created. These functions then use the Gensim library for Python to calculate the different vector distances.
We compare 2 analogy models: [Explanation about the analogy models]
We use freely available pretrained vector models for GloVe and Word2Vec. The GloVe models are almost the same format as the Word2Vec models. The only difference is that Word2Vec had a header with the dimensions. Once added, the GloVe model can also run with the default Gensim operations for Word2Vec.
If a word does not occur within the pretrained vector model, we generated two different recall numbers: One where the missing word is considered a failed for the analogy and one where the missing word is just ignored. We could also have done a third option, were we use the nearest word in the vector model instead of the missing word. However, we did not do this because this would take a lot more computing power and (as seen in the results below) we're not sure whether it would have made a difference.

2.3 Results

GloVe always had the same amount of skipped words (all words) independent of the dimension within a category. So we will not show the category results for these.

Category	**Recall**
Capital common countries	0.83202
Capital World	0.79134
Currency	0.35104
City in state	0.70896
Family	0.84585
Gram1 adjective to adverb	0.28528
Gram2 opposite	0.42734
Gram3 comparative	0.90841
Gram4 superlative	0.87344
Gram5 present participle	0.78125
Gram6 nationality adjective	0.89931
Gram7 past tense	0.65962
Gram8 plural	0.89865
Gram9 plural verbs	0.67931
Total	0.73588

Table 1: Word2Vec addition model (ran for 1h15)

Category	**Recall**
Capital common countries	0.85178
Capital World	0.80570
Currency	0.35450
City in state	0.71423
Family	0.84585
Gram1 adjective to adverb	0.31552
Gram2 opposite	0.42365
Gram3 comparative	0.90691
Gram4 superlative	0.91800
Gram5 present participle	0.80587
Gram6 nationality adjective	0.89368
Gram7 past tense	0.70641
Gram8 plural	0.89940
Gram9 plural verbs	0.73448
Total	0.75148

Table 2: Word2Vec multiplication model (ran for 3h25)

2.4 Discussion

We noticed that the GloVe model was not able to find all the words and this influences the results regarding execution times. The GloVe model seems to run a lot faster than the Word2Vec model because of our implementation strategy. If one word of the analogy is missing, the whole analogy is skipped resulting in smaller search space that needs to

Category	Recall
Family	0.85573
Gram1 adjective to adverb	0.25403
Gram2 opposite	0.22660
Gram3 comparative	0.86486
Gram4 superlative	0.69786
Gram5 present participle	0.68277
Gram7 past tense	0.60192
Gram8 plural	0.77402
Gram9 plural verbs	0.59770
Total	0.30778
Total no skipped	0.62774

Table 3: GloVe50d addition model (ran for 3 min)

Category	Recall
Family	0.62846
Gram1 adjective to adverb	0.09980
Gram2 opposite	0.04926
Gram3 comparative	0.41366
Gram4 superlative	0.17112
Gram5 present participle	0.29451
Gram7 past tense	0.31153
Gram8 plural	0.46021
Gram9 plural verbs	0.25862
Total	0.14506
Total no skipped	0.29587

Table 4: GloVe50d multiplication model (ran for 3 min)

be covered. But that aside the GloVe model has the edge with faster calculations due to its lower dimensionality.

[answer on the complexity question.. I dunno D: Also, we should probable make a graph showing all results next to each other]

If we look at the dimensionality of the GloVe representations, we see that the overall accuracy increases as the number of dimensions increases. This is a trade-off compared to the time needed to run the model, but in general it seems worthwhile. If we then look at the analogy model compared to the dimensionality, we see that the multiplication model performs better on higher dimensions while the addition model performs better as the dimension decrease. The overall accuracy decrease with a lower dimension is most likely the reason for this.

Category	Recall
Family	0.81621
Gram1 adjective to adverb	0.24395
Gram2 opposite	0.20074
Gram3 comparative	0.79129
Gram4 superlative	0.54278
Gram5 present participle	0.69508
Gram7 past tense	0.55448
Gram8 plural	0.71997
Gram9 plural verbs	0.58391
Total	0.28382
Total no skipped	0.57890

Table 5: GloVe100d addition model (ran for 7 min)

Category	Recall
Family	0.77866
Gram1 adjective to adverb	0.22883
Gram2 opposite	0.15764
Gram3 comparative	0.74850
Gram4 superlative	0.50624
Gram5 present participle	0.65057
Gram7 past tense	0.52821
Gram8 plural	0.66667
Gram9 plural verbs	0.57356
Total	0.26668
Total no skipped	0.54394

Table 6: GloVe100d multiplication model (ran for 7 min)

While running the experiment, we frequently got missing words (using the GloVe representation). It is still impractical to have a model representing every possible word that exists, so a trade-off is made towards a reasonable model size and the amount of represented words. We solved this error by either counting a missing words as a failed analogy or by skipping the sentence. However, it seems that the same word categories are missing in every dimension.

Category	**Recall**
Family	0.85573
Gram1 adjective to adverb	0.25403
Gram2 opposite	0.22660
Gram3 comparative	0.86486
Gram4 superlative	0.69786
Gram5 present participle	0.68277
Gram7 past tense	0.60192
Gram8 plural	0.77402
Gram9 plural verbs	0.59770
Total	0.30778
Total no skipped	0.62774

Table 7: GloVe200d addition model (ran for 10 min)

Category	**Recall**
Family	0.85178
Gram1 adjective to adverb	0.25302
Gram2 opposite	0.19581
Gram3 comparative	0.83784
Gram4 superlative	0.67162
Gram5 present participle	0.67045
Gram7 past tense	0.60321
Gram8 plural	0.76426
Gram9 plural verbs	0.65172
Total	0.03531
Total no skipped	0.62273

Table 8: GloVe200d multiplication model (ran for 10 min)

Category	**Recall**
Total	0.31304
Total no skipped	0.63849

Table 9: GloVe300d addition model (ran for 20 min)

Category	**Recall**
Total	0.32214
Total no skipped	0.65707

Table 10: GloVe300d multiplication model (ran for 20 min)

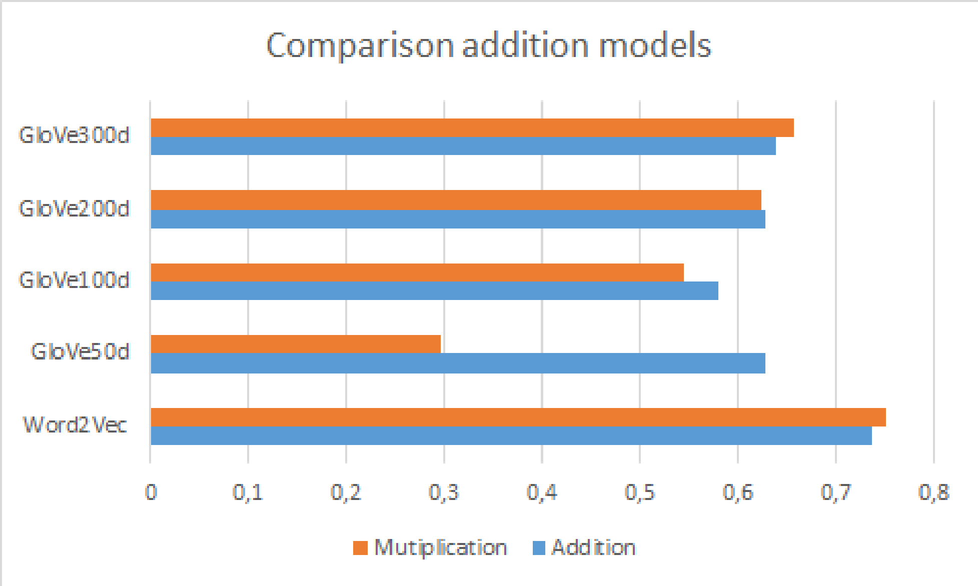

Fig. 1

3 Part II

3.1 Task description

3.2 Setup

3.3 Results

3.4 Discussion

4 Conclusion

[Strong points, Weak points, lessons learned]

5 References