Semantic Embeddings in IR

Text-Based Information Retrieval

Joren Verspeelt, Jochem Geussens, Vincent Tanghe

April 17, 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$^{TM} i7-3610 QMCPU, SSDharddiskandWindows7OperatingSystem.$

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

Category — Recall ——————

Category	Recall
Capital common countries	0.83202
Capital World	0.79134
Currency	sdqs
City in state	qsdqs
Family	0.83202
Gram1 adjective to adverb	0.83202
Gram2 opposite	0.83202
Gram3 comparative	0.83202
Gram4 superlative	0.83202
Gram5 present participle	0.83202
Gram6 nationality adjective	0.83202
Gram7 past tense	0.83202
Gram8 plural	0.83202
Gram9 plural verbs	0.83202
Total	0.83202

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

2.4 Discussion

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