**Report Group 06**

**Exercise 2**

WS 2021 - 188.977 Grundlagen des Information Retrieval

**Part1: Warmup**

|  |  |  |
| --- | --- | --- |
| **Word1** | **Word2** | **Cosine Similarity (~)** |
| cat | dog | 0.75 |
| cat | Vienna | 0.17 |
| Vienna | Austria | 0.78 |
| Austria | dog | 0.22 |

* cat and dog are antonyms, thus they relate to each other
* cat and Vienna get a low score since they are also semantically not related but might have been mentioned on the same article
* Vienna and Austria usually are mentioned in a lot of articles together since Vienna is the capital city of Austria
* Austria and dog has in this case the same relationship as cat and Vienna.

|  |  |  |  |
| --- | --- | --- | --- |
| **Word** | **Top-1** | **Top-2** | **Top-3** |
| Vienna | Salzburg (0.78) | Austria (0.78) | Prague (0.77) |
| Austria | Austria (0.81) | Vienna (0.78) | German-Austria (0.76) |
| cat | cats (0.84) | housecat (0.77) | -cat (0.76) |

For the word “Vienna” the model has successfully established a relationship between geographical location and political relation with the results. The model does, of course, not know this but depending on the dataset that was used to train it, we manage to get “Salzburg” as the first result which makes sense, because Vienna and Salzburg are both cities of Austria. This is also the reason why Austria comes in second. While the third result “Prague” might not seem that obvious it also makes sense, because Vienna and Prague are geographically close to each other (they might have been mentioned in the dataset as being close to each other) and they were also part of the same empire, meaning they share history together, indicating that the dataset could’ve had historical articles in it.

For the word Austria we get Austria as the first result, Vienna as the second since Austria also came up as a result for Vienna and then “German-Austria”. In the third case Austria was part of the multi word entity.

For the word “cat” we get only words where the word “cat” is part of the substring in all of the computed results.

**Part2: Short-Text Similarity**

|  |  |  |
| --- | --- | --- |
| **Method** | **Preprocessing** | **Pearson Correlation** |
| Vector Space Model (from sklearn library) | Lowercasing+Stopword | 0.71 |
| Average Word Embedding | Lowercasing+Stopword | 0.58 |
| IDF Weighted Agg. Word Embedding | Lowercasing+Stopword | 0.58 |
| Vector Space Model (from sklearn library) | Lowercasing | 0.69 |
| Average Word Embedding | Lowercasing | 0.56 |
| IDF Weighted Agg. Word Embedding | Lowercasing | 0.56 |

The methods with stopword removal obviously worked better because stopwords add more noise to weighting, thus making the scores less accurate. The TfIDF method worked better than the other methods since its strategy for determining weights includes better estimators with smoothing, compared to just taking the mean or just dividing by the idf for an average.

**Part3: Training new language models**

|  |  |  |  |
| --- | --- | --- | --- |
| **Word (of your choice)** | **Top-1** | **Top-2** | **Top-3** |
| spiel | game (0.74) | heimspiel (0.71) | gegentor (0.71) |
| polizei | polizisten (0.7) | täter (0.63) | kriminalpolizei (0.62) |
| berlin | berliner (0.76) | hamburg (0.63) | münchen (0.63) |

We decided to use a dataset containing some 50 thousand german tweets. As for the results:

* For the word “spiel” we see in the first result its translation in english. Since Twitter is a platform that is mostly used by Millenials and younger generation, they tend to use English words in a mix with the german ones. The second result is a compound word which contains spiel. Both the second and third result come from the Football jargon meaning the dataset was heavily influenced by football.
* For the word “polizei” we see that the first and third results contain polizei in them and the second result is semantically related to “polizei” since the police is working against criminals (täter)
* For “berlin” we see in the first result the denonym and the other results are other influential cities in Germany.

**NOTE:**

Please keep in mind that for part 3 a dataset of ~780MB was used that generates a model with size 2.84GB. The training of the model took a little bit more than 1 hour and 10 minutes on a MacBookPro 13” 2017 (2,3 GHz Dual-Core Intel Core i5)