University of Heidelberg,
Faculty of Modern Languages,
Department of Computational Linguistics

Comparing Semantic Changes of Anglicisms in German – using „CrOssinG“

Motivation

Our project's goal was to analyze how much an anglicism's meaning changes during its transition from English to German.

For this task, we built two vector space models based on the English and German Wikipedia (1) by using dictionary data extracted from **dict.cc**. (2)

We applied our tool **„CrOssInG“** on this data to analyze which anglicisms displayed a significant change in meaning. (3)

1. We used **word2vec** to create two vector space models on the English and German Wikipedia. Every word's meaning could then be represented by a word vector of the following form:

```
katze    0.006136 -0.052587 0.012688 -0.01403 -0.046991 0.042845 -0.023529 -0.001199 . . .
cat     -0.067114  0.033746 0.020565  0.032246 0.113999 0.016741 -0.021005  0.043264 . . .
```

2. The next step was to acquire a suitable translation into English for every German word. **dict.cc** provided a dictionary in the following format:

```
German {Num./Gen.} Addition <Abbr.>/[com.] English <Abbr.>/[com.] word type
```

An example entry might look like this:

```
platituedenhaft   [alt.] [geh.]               platitudinous         adj
```

Having removed stop words, ambiguous translations and other low-priority words, we extracted this format for every respective translation, and created a bijective mapping from a German word to its English translation.

3. From the previous steps, we established a mapping from each German word *v* to an English translation: *translation(v) = w*.

Our next task was to find the transformation matrix *T* that would map v to w as close as possible:

$$Tv = w' \approx w.$$

From the difference between *w'* and *w* (calculated by cosine similarity), we could deduct how close an anglicism is to its original meaning.

Evaluation

During the process described in (3), we created several transformation matrices based on different linear regression methods (*ridge, elastic net* and *lasso*) using the Python package **scikit-learn**. *S* denotes the set of those matrices.

Given a set of false friends pairs *F*, the most accurate mapping *T** could then be found using the following formula:

$$T^* = \min_{T \in S} \frac{\sum_{(f,f')\in F} cosine\_similarity(Tf, f')}{|F|}$$

Results

	Highest similarity pairs	similarity
1	peeling - exfoliation	81.80%
2	body - onesie	77.97%
3	spray - spray	76.75%
4	aftershave - aftershave	76.59%
5	t-shirt - t-shirt	76.09%

	Lowest similarity pairs	similarity
1	city - city	46.83 %
2	team - team	47.65%
3	campus - campus	48.57%
4	in - in	48.87%
5	golden_goal - golden_goal	49.27%

Conclusion

Comparing different combinations of models and alpha values, we found that an elastic net model with an alpha value of α = 0.1 performed best using our method of evaluation.

Whilst the top ten of highest similarities seem promising at first glance, many of the results of the lowest-similarity top ten seem to be beyond the usual amount of freak values and therefore let us question our approach regarding the following points:

1. The vector data obtained from word2vec (small values between 0.1 and 0.01)
2. Similarly to 1., the performance of scikit-learn on that vector data.
3. Calculating the precision of our transformation matrices by evaluating false friends pairs. Evaluating on German and English words with a similar meaning might have been more precise.

Dennis Ulmer, Sebastian Spaar
https://github.com/Eroica/crossing

Tomas Mikolov, Quoc V. Le, Ilya Sutskever: *„Exploiting Similarities among Languages for Machine Translation“*