Statistical Machine Translation

Homework 5

To be sent (pdf, Zip) to `waszczuk@phil.hhu.de`.
Deadline: 18.01.2019

Important: you need to solve the theoretical exercise first (Exercise 2)!

Exercise 1 - Practice

This week we implement the training of a phrase-based translation model, in particular the process of learning the phrase translation table. Download the code (Eclipse project as a zip file) and save it locally. Start Eclipse and import the zip file as existing project. At the end of the session, you can export your project as zip archive and keep a copy of it (email, USB).

Make yourself familiar with the code. The main class is `de.hhu.phil.smt.pb.Uebung5`. It already contains code and comments. Your tasks are located in `PhraseExtractor` and `PhraseScorer`.

`PhraseExtractor` should extract all phrase pairs that are consistent with the given alignments and which are not longer than a given maximum phrase length, and write them to a file. Large part is already implemented. You only need to implement the core extraction algorithm.

This file is then sorted. Then `phrase Scorer` is called to calculate the phrase translation probabilities based on this sorted file. You need to implement this part. Some help is given in the comments. Please start by solving the theoretical task A2 (2).

First, use the sample data `michael.de`, `michael.en` and `michael.de-en.align`. These correspond to task A2 (1), so you can check whether your implementation is working properly. Then please use the data in `europarl-v7.de-en.30.h5000.*`. It consists of 5000 pairs with a maximum length of 30 words. It could be the case that your computer needs some time to process it.

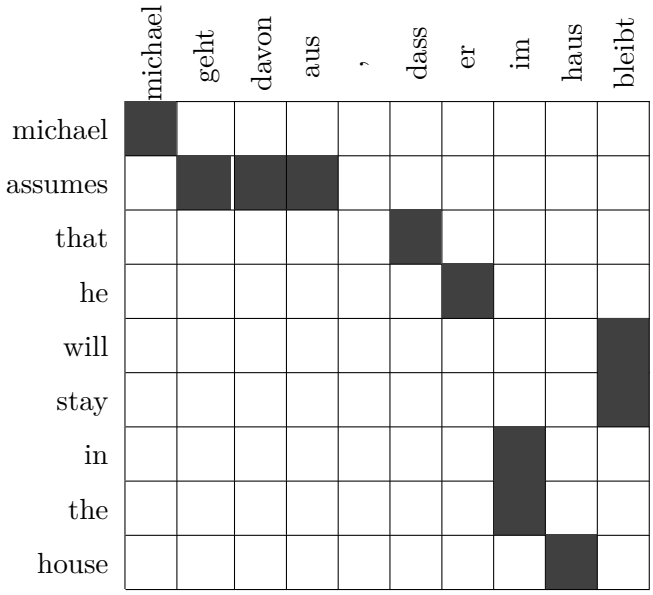
According to your translation model $\hat{P}(\bar{e}|\bar{d})$ in `Europarl-v7.de-en.30.h5000.trans`: What translations are available for the German phrase `wie ich meine ,`? And how likely are these?

General note: You do not have to stick exactly to the proposed structure of the code. If you are not sure, feel free to ask, but in any case, make sure that anyone can understand your code, for example by using comments. If the original entry point of the program no longer works, please add a readme file that specifies how the code should be compiled and executed.

Exercise 2 - Theory

1

Given a sentence pair with the following word alignment:



(a) List all phrase pairs that are consistent with this alignment.

(b) Suppose that your corpus D only consists of the single sentence pair given above. Estimate the phrase translation probabilities $\hat{P}(\bar{e} \mid \bar{d})$ for all phrase pairs $(\bar{d}, \bar{e})$ in the corpus. As a reminder:

$$\hat{P}(\bar{e} \mid \bar{d}) = \frac{C(\bar{d} \rightarrow \bar{e}; D)}{\sum_{\bar{e}' \in R_E(\bar{f})} C(\bar{d} \rightarrow \bar{e}'; D)}$$

where $R_E(\bar{f}) \subset V_E^*$ is the set of English phrases to which $\bar{f}$ can translate.

2

With a " normal sized " parallel corpus (> 1 million pairs), a practical problem arises when estimating the translation probabilities: it is not possible to have all phrase pairs

simultaneously in memory. Therefore one can not calculate $C((\bar{d}, \bar{e}), D)$ naively.

Instead, one writes gradually all phrases couples extracted from D in a large text file (which is located on the hard disk, not in memory, to avoid the problem). This would look for example like this:

```
michael ||| michael
geht davon aus ||| assumes
...
michael ||| michael
...
michael ||| our michael
...
```

The question is to know how you can, starting from this file, calculate $\hat{P}(\bar{e}|\bar{d})$ without reading the entire file at once? Tip: there are efficient sorting algorithms that can sort this large file line by line (without having it completely in memory, for example, mergesort). Starting from a sorted file (all phrase pairs sorted, for example on the German side), how can one calculate $\hat{P}(\bar{e}|\bar{d})$ (without reading the complete file at once)? Sketch the algorithm as pseudo-code.

3

Concerning the number of consistent phrase pairs with a given word alignment, the words themselves play no role. It is enough to have the alignment relation (with positions represented as numbers). Your task is, given an alignment relation, to return *how many* well formed phrases there are for the sentence pair. The lengths of the associated sentences correspond to the highest numbers occuring in the corresponding components of the relation, i.e. for a) below we have sentences of length 7 and 6, for b) of length 7 and 8, and for c) of length n and n.

a) $A_1 := \{(1,1), (2,2), (3,2), (4,4), (2,3), (5,4), (5,6), (6,3), (7,2)\}$

b) $A_2 := \{(7,8)\}$

c) (bonus exercise) $A_3 := \{(1,1), (2,2), (3,3), ..., (n,n)\}$, for any $n \in \mathbb{N}$. The solution is of course not a number but a monotonically increasing sequence (function $\mathbb{N} \to \mathbb{N}$), such as $4n + 3$. Try to illustrate graphically the growth of the possibilities.