Natural Language Processing: Assignment 2

Johannes Gontrum

---

1 POS-Tagging

1.1 Tagger tuning

1.1.1 Parameter optimization

I finetuned my tagger by using first a coarse-grained and eventually a fine-grained grid search [1] to optimize the parameters in the ranges $1 \leq t \leq 9$, $1 \leq e \leq 9$, $1 \leq f \leq 30$ and $1 \leq s \leq 30$. Though the solution is not guaranteed to be the global optimum, since not all combinations in the search space have been tried, I reach an accuracy of **93.18%** with the settings $t = 4$, $e = 3$, $f = 14$, and $s = 6$.

When analyzing the errors of the tagger on the given test set, I was amazed by the good performance. There are only a few words that are obviously tagged wrongly, most differences to the gold standard are subtle, ambiguous, or caused by a sentence from a non-English language.

1.1.2 Error analysis

A recurring error in the first sentences is the tagging of *associate* in *associate judge* as a noun, while the gold standard suggests an adjective here. In my opinion, this is an ambiguous case, since *associate judge* is a job title and I find it valid to describe both as nouns. There are also other mistakes that even humans struggle with like the distinction of adjectives and adverbs: *... the Sunni heartland is being done wrong ...*, where the gold standard marks *wrong* as an adverb, while the tagger sees it as an adjective.

In some cases, however, the tagger shows clearly a problem to distinct verbs from nouns, preferring verbs. As an example, the model falsely tagged *hope* as a verb in the context *... lost his patience and his hope in peace ...* or *move* again as a verb in *... this move was a bad, bad tradeoff ...*.

However, other cases are not so clearly: The tagger has problems to distinguish between normal nouns (*NOUN*) and proper nouns (*PROPN*), which are used for names of persons, companies, products etc. Again, this is a thin line where at times even human annotators do not necessarily agree. While I see it as a mistake that in the context *Court of Appeals* the model classifies *Appeals* as a noun and not a proper noun, I disagree with the gold standard in others cases. The *F* token in the plane type *F-16* should clearly be *PROPN* like it is tagged by the model. However, the gold standard sees it as *NOUN*. *israelians* on the other hand is set to be a proper noun by the annotators, while I would prefer the tagger's output *NOUN*.

Another interesting case is the usage of the part-of-speech tag *X* that serves as a bucket for everything that does not fit any another tag. Here, one would expect the tagger failing to classify these 'unknown' contexts since I assume it is hard to distinguish these cases. In one of the first sentences, we encounter exactly this issue: The french sentence *A la guerre c'est comme a la guerre!* is entirely marked as *X* in the gold corpus. The tagger, however, tries to tag these unknown words and fails miserably.

1.2 German tagsets

There exist multiple tag sets for the German language. Yet, the Stuttgart-Tübingen-TagSet (STTS) (see Schiller (1995)) is by far the most prominent one. It has also been the foundation for various modifications, like the one used in the TIGER treebank (Brants et al. (2002)). Less known are the annotations of the POS tagger developed by Xerox and described in Feldweg (1999) or the Münsteraner Tagset (Steiner (2003)) from the University of Münster.

1.2.1 Stuttgart-Tübingen-TagSet

Therefore I will describe the STTS tagset in more detail: All in all, the tagset is fine-grained as it contains 54 different tags. There are for example four distinct tags that describe variations of prepositions, depending on

[1] See `https://docs.scipy.org/doc/scipy/reference/optimize.html`

their position relative to the noun (*APPR*, *APPO*, *APZR*) or if they include a determiner (*APPRART*): 'in dem' (in the) ⇒ 'im'.

Also, different kinds of conjunctions are separated. Some conjunctions are used for comparison (*KOKOM*) others introduce subordinate clauses (*KOUI*, *KOUS*) which have their own word order in German and it might therefore relevant to treat them differently.

Most prominently, however, are the different types of verbs: Here, 12 tags model variations of verbs, some for auxiliary verbs, modal verbs and even four different types of infinitives.

Though admittedly this fine differentiation seemed very confusing to me at first, human annotators reach an inter-annotator agreement of more than 98% (see Brants et al. (2002)). Also despite the high number of different tags, Evert and Giesbrecht (2009) summarize that german POS tagger using the STTS tagset reach an accuracy between 94% and 97% on a newspaper corpus.

1.3 Tokenization preprocessing

Naturally, an HMM-based part-of-speech tagger requires a somewhat segmented text, as it is evaluating a sequence of tokens. But without tokenization the number of words would increase drastically:

In the following example, both occurrences of *cat* would be treated like completely different words:

The cat sleeps. The cat's fur is so soft.

The tagger sees the tokens *cat* and *cat's* and therefore needs a larger amount of text to estimate good probability values for both of them. The same is true for stripping punctuation characters from tokens. Not tokenization a text would drastically increase the number of unknown words, which are more tricky to deal with than well-observed tokens.

Well tokenized and lowercased, the amount of types in the sentence is reduced to $\frac{9}{12}$ from $\frac{8}{9}$ in the first example:

the cat sleeps . the cat 's fur is so soft .

However, for languages like Chinese, a part-of-speech tagger might perform better if it is character based. Tokenization is a harder problem for this type of languages and there are approaches that skip the tokenization step completely as demonstrated by Ng and Low (2004).

1.4 Part-of-speech tagging with HMMs

In the lab, we saw that the sequence of keystrokes (digits) are the observed emissions generated by the hidden states - the characters a-z. We could, therefore, calculate the most likely sequence of characters given an observed sequence of keystrokes.

When using a Hidden Markov Model (HMM) for part-of-speech tagging, we regard the words of the text we want to tag as the emissions of the underlying, hidden states. Therefore, the hidden states are the POS tags and the tokens the observations.

This is motivated by the assumption that language is produced by a generative, random process of the model. This is obviously not true since it implies that we write and speak by first randomly picking part-of-speech tags and then let them generate the actual words given their probability distribution. Nevertheless, we can use an HMM under this assumption to calculate the most probable sequence of POS tags given their observed emission of words.

2 Lemmatisation

2.1 Lemmatizer tuning

The best version of my lemmatizer reaches an accuracy of 97.76%, though most of the improvement is achieved by adding rules for pronouns and irregular verbs. My approach for adding rules only took the last one or two characters of the stem into account. Retrospectively, this caused a poor performance and in the future, I would use longer suffixes to form rules like done by the Porter Stemmer[2].

[2] See `https://tartarus.org/martin/PorterStemmer/`

2.1.1 Problem: Past tense verbs

The most common mistake of my lemmatizer is adding or removing a trailing ”e”. This is very obvious for verbs in the past tense. Here, ”promised” is turned into ”promis”, ”referred”, however, becomes ”referre”. Like mentioned before, I assume that my rules are too simple for this case since they only take the length of the word (short words are always assigned an ”e”) and the last letter into account. For example, ”nominated” $\rightarrow$ ”nominat” $\rightarrow$ Rule: last letter is t, g, r or c $\rightarrow$ ”nominate”. Though this rule increases the accuracy by 1.25 percentage points, it is clearly too short-sighted. The Porter Stemmer, for example, takes longer sequences of characters into account to make better decisions.

2.1.2 Problem: Verbs ending in *-ing*

A very similar error occurs for verbs ending in ”ing” because I took a similar approach, where I added an ”e” depending on the last two characters. Words like ”including” are reduced to ”includ”, while my rules perform correctly e.g. for ”retiring” which is lemmatized to ”retire”. My mistake was to optimize my rules for certain words, while not observing the overall performance. Here a better approach might be to find concrete consonant pairs in which the words are ending (like ”AT”, ”BL”, ”IZ” in the Porter Stemmer) and only add the ”e” to these words. Unsurprisingly, my attempt only adds 0.91 percentage points to the accuracy.

2.1.3 Problem: Singular nouns ending in *-s*

Another case that is more difficult are singular nouns ending in ”s”. Generally, the rule is to strip the ”s” of nouns since it is the plural suffix most of the time. Words like ”focus”, ”troops”[3] or ”bus” on the other hand will be falsy lemmatized to ”focu”, ”troop” or ”bu”. I have tested the Porter Stemmer[4], and surprisingly it makes the same mistakes. Interestingly, however, the snowball stemmer[5] (or Porter 2) performs correctly on these examples. I can only assume that words like ”focus” are treated by a hand-crafted word list since I can not imagine a rule that could model this.

2.1.4 Problem: *ie* $\rightarrow$ *y*

In many words, like ”cities” or ”cried”, the vowels ”ie” will turn into a ”y” in the lemma. As one of the last rules in my pipeline and after all specific rules are executed I perform the rule ”ie $\rightarrow$ y”. It works well in most cases, but also falsy turns ”veggie” into ”veggy”. I would improve my rule to only change words that have been altered by previous rules. ”cities” for example has been modified to ”citie”, ”cried” to ”cri”. ”veggie” on the other hand remained the same.

2.1.5 Problem: Irregular words

Eventually, a big issue are irregular words like ”came” $\rightarrow$ ”come”. Per definition, these words do not change according to rules thus we must define the lemma specifically per word. I suspect that industry-level lemmatizers use a large dictionary to cover most of these cases. Even in my lemmatizer, adding rules for the irregular word ”be” improved the accuracy by about 7 percentage points.

2.2 Finite State Transducers

The finite-state transducer (FST) for our basic lemmatizer accepts an input in the form ”WORD TAG” and outputs the lemma for the given word. As illustrated in Figure 1, its starting state is q_0, while the only final state is q_7. The FST must read the input and reach the final state for the output to be returned. Transitions from one state to the other are in the form x, y where x stands for the input and y for the output. Please note that sequential transitions for a string are collapsed for reasons of clarity and comprehensibility: Instead of going from q_0 to q_1 with ”cat”, an FST would actually go character by character (c:c $\rightarrow$ a:a $\rightarrow$ t:t).

The general idea behind the FST is to use different routes for different part-of-speech tags, so we can handle their suffixes accordingly. Since our lemmatizer only strips suffixes, we can start by recognizing the stem of

[3] I would argue that the lemma of ”troops” is indeed ”troop”, but the gold standard disagrees.

[4] JS Porter Stemmer: `http://9ol.es/porter_js_demo.html`

[5] See `http://proiot.ru/jssnowball/`

the words which we also output. If we use *higher ADJ* as an example input, we are now in state q_5: *higher ADJ*. The next transition describes the possible suffixes for words with this tags. Following our example, we can go to the next state q_6 either with "er" and" est". Since they are not part of the lemma, we output ϵ, the empty string: *higher ADJ*. Eventually, we ensure that "high" is the lemma of "higher" only if it is an adjective. This might be an obvious case but it allows us to distinguish words with ambiguous tags like "lead NOUN" and "lead VERB". We continue by recognizing the tags preceded by a space character. Again, we output ϵ as the tag is obviously not part of the lemma. After the tag, we arrived in the final state q_7: *higher ADJ*.

2.3 Tagging morphologically rich languages

Morphologically rich languages encode grammatical information into the words. Consequently, there are many different words derived from the same lemma, most of which have the same part-of-speech tag. In German, for example, we append the person, tense and number information as suffixes to the stem of a verb.

Part-of-speech taggers perform better the more often they encounter a token during training because they learn more accurate probabilities this way. When using a tagger in a morphologically rich language, however, it treads different forms of the same lemma as completely different words. Given the example of the lemma "schreiben" ("write") in Figure 2, a tagger cannot see that "schreibt" (3rd person, sg, present) and "schrieb" (3rd person, sg, past) both are verbs, even with the same person and number. Hence, both words will occur in very similar contexts, but the tagger will be unable to see a connection between them. This will lead to a high number of out-of-vocabulary words during the tagging since it is not guaranteed that all forms of a word will be seen during the training. Therefore, it might be a good idea to lemmatize a text before using it with a part-of-speech tagger, though this approach could also lead to confusion with words that have word forms in with different POS tags (e.g. "Schreiber" ("writer")).

2.3.1 Experiment: Training a tagger on lemmatized data

I was curious about this hypothesis and decided to conduct an experiment myself, in which I compare the performance of the *hunpos* part-of-speech tagger on a corpus of either tokens or lemmas. The German TIGER corpus (release 2) contains not only syntactic information but also the lemma and part-of-speech tag for each token, so I chose it.

From the corpus I extracted two initial files: One with the token and the POS tag and one with lemma and tag. In the training, I was restricted to the first 200.000 words, since *hunpos-train* malfunctioned with larger files. As the test set, I used the last 20.000 words, making sure not to overlap the training set. I trained a model with the default parameters of *hunpos-train* for both settings and evaluated the outcome with our `score.py` script.

While the token-based model performed well with an accuracy of 95.23%, the lemma-based approach only reached a score of 87.35%. This contradicts my hypothesis that lemmatized data would improve a tagger in a morphologically rich language. However, during my analysis of the used STTS tagset, I realized that there exist several tags e.g. for verbs, which heavily depend on the form of the token. By replacing it with its lemma, this information is lost.

I still believe that languages with an even richer morphology like Turkish or Finnish will benefit from careful lemmatization as a preprocessing step to tagging and I would be excited to repeat my experiment with a corpus in one of these languages.

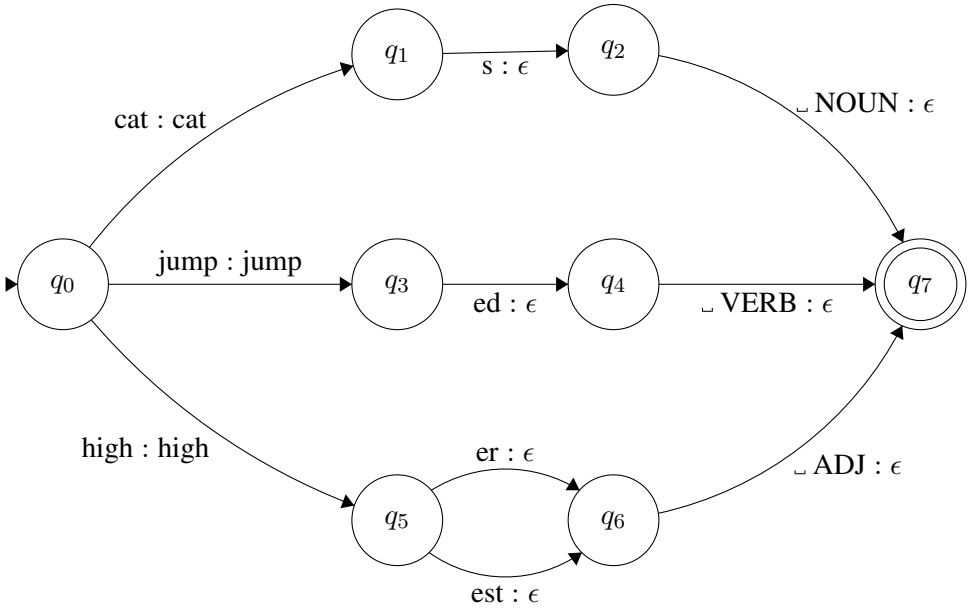

Figure 1: A *Final-State Transducer* reading the strings ”cat NOUN”, ”jump VERB”, ”higher ADJ”, ”highest ADJ” and returning the lemma of the word in the input. The starting state is q_0, the only final state q_7. The transitions are in the form x, y where x is the input and y the output. The empty string is denoted as ϵ.

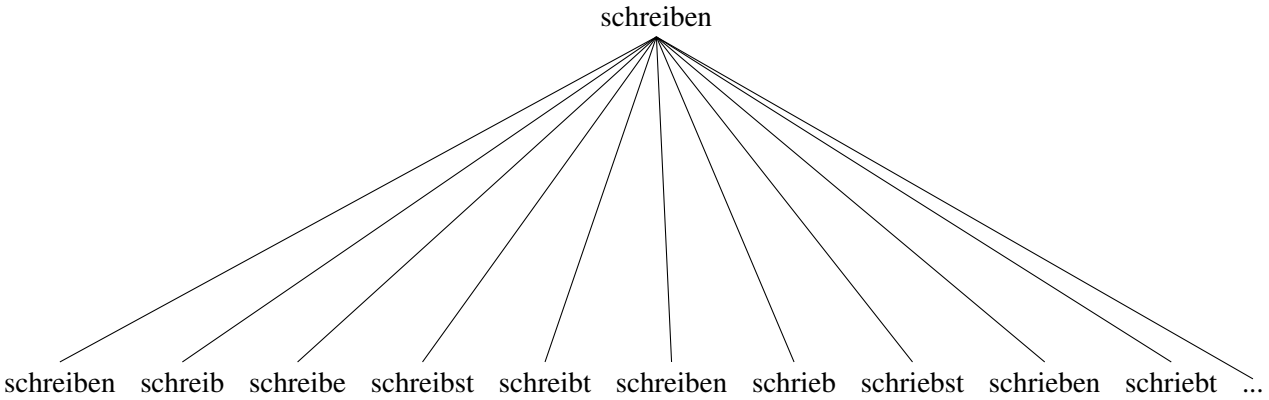

Figure 2: Example of German verb morphology: The lemma ”schreiben” (”write”) with a subset of its different forms.

References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168, 2002.

S Evert and E Giesbrecht. Part-of-speech tagging–a solved task? an evaluation of pos taggers for the web as corpus. In *Proceedings of the 5th Web as Corpus Workshop (WAC5), San Sebastian, Spain*, 2009.

H. Feldweg. *Implementation and Evaluation of a German HMM for POS Disambiguation*, pages 1–12. Springer Netherlands, Dordrecht, 1999. ISBN 978-94-017-2390-9. doi: 10.1007/978-94-017-2390-9_1. URL https://doi.org/10.1007/978-94-017-2390-9_1.

Hwee Tou Ng and Jin Kiat Low. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *EMNLP*, pages 277–284, 2004.

Anne Schiller. Vorlaufige guidelines fur das tagging deutscher textcorpora mit stts. In *Seminar fur Sprachwissenschaft, 1995*, 1995.

Petra Steiner. Das revidierte münsteraner tagset/deutsch (mt/d). beschreibung, anwendung, beispiele und problemfälle, 2003.