CS5012: Practical 1
The size of the tagset in POS tagging

#160022011
School of Computer Science
University of St Andrews

1 Overview

During the practical a first-order Hidden-Markov-Model (HMM) tagger was developed which was trained with words from the Brown corpus. The tagger was capable of achieving an accuracy of 94.42% for an unknown test set of 500 sentences within the Brown corpus featuring the original tagset of 472 tags. With a reduced tagset of 76 tags, the tagger was even able to achieve an accuracy of 96.03% which corresponds to the *gold standard* for POS tagging. Furthermore, tests were conducted to further reduce the tagset as well as tests to determine the impact of the size of the training set. Also the tagger was applied to the Dutch Alpino corpus in `nltk` and the results were compared to the Brown corpus.

2 Training

The Hidden-Markov-Model was trained with the first 50.000 pre-tagged sentences from the Brown corpus. The training phase resulted in a relative frequency estimation for transition probabilities between two tags and emission probabilities between word-tag tuples within the training set. The relative frequency estimation was designed by creating two tables implemented by *Python* dictionaries with word-tag tuples and respectively tag-bigram tuples as a key and a probability as a value. During the training phase the model has been made robust by enabling it to deal with unknown words and by smoothing on the transition probabilities.

2.1 Unknown Words

In order to make the model capable of dealing with unknown words in test sets which have not occurred in the training sentences yet, the model needs to be capable of providing emission probabilities for unknown words as well. This was solved by replacing words that only occur once in the training set (so-called *hapax legomena*) with an artificial `UNK`: As soon as all sentences are imported from the corpus, the frequency distribution of all word tokens in the sentences is calculated and filtered for the words that only occur once in order to obtain the set of *hapax legomena*, which is defined as

$$H_L = \{\forall w_i \in W \mid C(w_i) = 1\}, \tag{1}$$

where W is the set of word types in the training set. The filtered words are saved in a *Python* set and in a second step each word token in each sentence of the training corpus is traversed and for each word token in the sentence it is asked if it is in H_L. If this is true, the token is replaced by the `UNK` string so that all subsequent operations are based on the new training sentences containing the `UNK` substitutions.

2.2 Relative Frequency Estimation

In order to prevent an underflow during training which may occur especially for large lexicon sizes and later during testing, as the probabilities are very small for long tag sequences, the model was extended by transferring the probabilities into the *log* domain.

As a first-order model was implemented, the *LaPlace*-smoothed transition probabilities between two tags are therefore calculated as

$$P_{log}(t_i \mid t_{i-1}) = log\left(\frac{C(t_{i-1}t_i) + 1}{C(t_{i-1}) + V_T}\right), \tag{2}$$

where V_T is the number of the tags in the tag set.

The emission probabilities for each word-tag pair were calculated as

$$P_{log}(w_i \mid t_i) = log\left(\frac{C(t_i, w_i)}{C(t_i)}\right). \tag{3}$$

To implement relative frequency estimation for both transitions and emissions the `nltk` functions `ngrams` and `FreqDist` were used. In a first step the frequency distribution of all tag unigrams in the training set was calculated which is needed for both transition and emission probabilities.

For the transition probability between two tags the frequency distribution for all tag-bigrams in the training set was calculated in a first step to obtain the absolute count of each bigram. The tag-bigram tuples were calculated by the `ngrams` function. A dictionary $T \times T$ was initialised for all tag-bigram combinations with the minimum log probability P_{log}^{min} =`log(float_info.min)` as $log(0)$ is undefined. It was then filled with probabilities for each tag bigram according to equation (2), which is the smoothed ratio between the bigram count and the unigram count of t_{i-1}.

It was proceeded accordingly for the emission probabilities. The word-tag tuples were extracted from the training sentences and a frequency distribution was calculated for the tuples. A dictionary of the size $W \times T$ for all word-tag tuples was then initialised with P_{log}^{min}. For each existing word-tag tuple in the `FreqDist` variable the value in the dictionary was updated by the calculation from equation (3), which is the ratio between the count of the word-tag tuple and the count of the tag-unigram.

As both tables are implemented as dictionaries, quick access to probability values of both transitions and emissions is possible by their keys which consist of a `(w,t)` or respectively `(t,t)` tuple.

3 Testing

In order to test the model the most likely tag sequence $\hat{t}$ was calculated for each sentence in the test set through

$$\hat{t}_1^n \approx \arg\max_{t_1^n} \left(\sum_{i=1}^{n} P_{log}(w_i \mid t_i) + P_{log}(t_i \mid t_{i-1}) \right) + P_{log}(\langle /s \rangle \mid t_n) \quad (4)$$

Algorithmically the calculation of $\hat{t}$ was generated by an implementation of a varation of the *Viterbi* algorithm. The algorithm was extended in two ways. First it was given the capability to cope with unknown words. For each word in the sentence, the algorithm asks if the word is in the lexicon of known words

$$L = \{W \backslash (H_L \cup unk)\}, \quad (5)$$

where W are the word types, H_L the *hapax legomena* and *unk* the `UNK` word. If a word cannot be found in L the algorithm looks up the emission probability for the current tag with the unknown word. Second, the algorithm works in *log* space and does not multiply but adds the probabilities P_{log} in order to prevent underflow during calculations. The model was tested on the 500 subsequent sentences after the training sentences and resulted in an accuracy of 94.42%. Even though a high overall probability was achieved, single tags were not determined with a high probability, as the bottom five tags in terms of accuracy show[1]:

NP-TL	0.2857
JJ-TL	0.6363
NP	0.6667
UH	0.72
QL	0.7381

The question arises, whether it makes more sense to merge tags in order to achieve a higher overall accuracy and a better accuracy for merged tags.

4 Tagset Experiments

In order to conduct experiments on the tagset a preprocessor has been developed using regular expressions which reduces the initial tagset according to given rules and subsequently re-tags the training sentences and test sentences.

[1]The probabilities were rounded to four digits after the comma in this document

4.1 Reduction of the tagset

In a first, intuitive step the tagset was reduced by merging all the compound tags in the initial tagset, e.g. `JJR-HL` to `JJR` and `FW-IN+NP-TL` to `FW`. This initial reduction of the tagset to 105 distinct tags led to a result of 94.91% which is slightly more accurate than the one of the full tagset. Looking at the bottom five probabilities per tag, it can be seen that the minimum accuracy is higher than in the first pass. However, tags that had a higher percentage before, lost some accuracy e.g. `BEG`:

BEG	0.5
QLP	0.6
FW	0.625
HVN	0.6667
AP	0.72

To see if the accuracy rises only by reducing certain tags, in a second step, additionally to the compound tags, negated `(*)` and non-negated tags as well as possessive `(\$)` and their non-possessive equivalents were merged together. It was found that even though the tagset was further reduced from 105 to 81 distinct tags, this did not lead to an improvement of the accuracy but rather stayed at 94.91%. It can be assumed that accuracy does not only depend on the pure size of the tagset but on smart merging on tags that belong closely together.

This assumption could be supported by an experiment, where the `C`-tags `(CC, CD, CS)` were naively merged together. This experiment resulted in a worse overall accuracy as if the tags were regarded separately. This observation may be explained as the `CD` tag which represents numerals is not closely related to the `CS` and `CC` tags which represent conjunctions. As the two types are used in different contexts other tags may be influenced negatively by the merge.

4.2 Finding a better tagset

In order to find a tagset that leads to better per-tag accuracy and subsequently to better overall accuracy, an approach was chosen to merge some of the tags together, that did not achieve 100%. Since after the test with 105 tags it was found that several types of verbs, adverbs, pronouns, nouns and adjectives were not fully recognised an approach was made to merge these different subtypes into their super classes leading to following combinations:

Class	Tags before merge	after
Adjectives	JJ, JJR, JJS, JJT	JJ, JJ$
Nouns	NN, NN$, NNS, NNS$, NP, NP$, NPS, NPS$, NR	N, N$
Pronouns	PN, PN$, PP$, PP$$, PPL, PPLS, PPO, PPS, PPSS, PRP, PRP$	P, P$
Adverbs	RB, RBR, RBT, RN, RP	R
Verbs	VB, VBD, VBG, VBN, VBP, VBZ	VB

This reduced set comprising 79 tags was able to achieve an accuracy of 96.03% which corresponds with the *gold standard*. Especially verbs achieved a high percentage of 96.99% in comparison to the individual verb sub tags before. Even though, the percentage is quite high already, an attempt was made to further improve accuracy by merging the tags `N` with `N$` and `P$` with `P`.

This approach led to a tagset of 76 tags and an accuracy of 96.10%. It was found that by merging tags where the accuracy is below 100% with grammatically related tags into a super class, the overall accuracy may be improved. The downside of the approach is that it is not possible anymore to distinguish between the sub tags. Therefore, it has to be weighed up if accuracy is more important than the diversity of different tags.

Even though a high accuracy was achieved, there are still tags which have quite low recognition rates, e.g. tags for *"having"* and *"being"* as well as qualifier tags. Also foreign words only had an accuracy rate of 70%. For foreign words there is no intuitive way to improve this rate, as tags of foreign words cannot be merged with other POS tags due to their differentness in comparison to the other tags.

BEG	0.5
HVN	0.5
QLP	0.6
FW	0.7
QL	0.7381

In order to improve recognition rates of `BEG` and `HVN` it was attempted to merge `BEG` with `BE`-like tags and `HVN` respectively with `HV`-like tags. However, the result stayed at 96.10%, but the number of tags decreased to 57, so a lot of diversity was lost during this step without any visible accuracy yield. As the accuracy did not increase significantly, it is not reasonable to replace the former tagset with the smaller one.

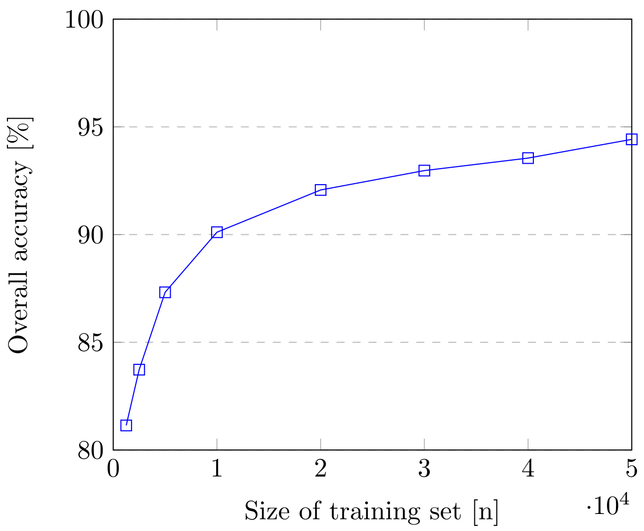
4.3 Impact of merging on other tags

Additionally to the size of the tagset, it was tested, what impact the merging of tags has on other tags. A test was conducted to determine the impact of merging all adjective tags to `JJ` on noun tags. The results show, that the reduction of the adjective tags leads to slightly higher overall accuracy but this comes to a price of weaker recognition rates for most nouns.

Tag	before	after	diff
overall	0.9442	0.9448	+0.0006
NN	0.9170	0.9188	+0.0002
NN$	1.0	1.0	
NNS	0.9586	0.9586	
NNS-TL	1.0	1.0	
NN-TL	0.8966	0.8387	-0.0579
NP	0.6667	0.6667	
NP$	0.8571	0.7778	-0.0793
NP$-TL	1.0	1.0	
NP-TL	0.4	0.2857	-0.1143
NR	1.0	1.0	
NR-TL	1.0	1.0	

5 Training Set Sizes

Different training set sizes were tested for the Brown corpus in order to determine their influence on the overall accuracy. Training sets with a sizes from 1250 to 50.000 were tested on the sentences 50.000 – 50.500. It can be seen that the accuracy rises quickly for smaller training set sizes but finally converges to $\sim 95\%$.



6 Tagging in other languages

An experiment has been conducted in order to test and compare POS tagging in different languages. In this case, POS tagging in Dutch was compared to English, using the Alpino corpus in `nltk`. In order to compare the two languages, measures had to be undertaken in order to make the two corpora compatible. As the Dutch Alpino corpus only consists of 7136 sentences, both the Alpino and the Brown corpus were therefore trained on the first 7000 sentences and tested on the subsequent 136 sentences.

Additionally, the size of the tagsets had to be adjusted, as the Alpino corpus has a tagset of only 16 tags. The 472 tags of the Brown corpus were first narrowed down to 81 non-compound tags without `$` and `*`. Then the different tags were merged together in order to adjust the dimensions of the tagsets. This task was particularly challenging, as the different tags in the Alpino tagset cannot be exactly assigned to a tag or group of tags in the Brown tagset. For example Alpino has an own tag for the comparative Dutch words *"als"* and *"dan"* whereas the English word *"than"* is tagged either as `CS` or `IN` in the Brown tagset. Also Alpino has an own tag for names while in Brown names are in `NP` together with proper noun phrases. On the other hand Brown has an own tag for the negation *"not"* (`*`) which does not exist in the Alpino corpus.

Therefore, some of the tags that could not be transferred were left unchanged. However, the Brown tagset could be narrowed down to 16 tags and Alpino was narrowed down to 14 tags as the tags `noun` and `name` were merged to achieve compatibility with the Brown corpus and the `pron` tag, which was only assigned to two words in the whole corpus was converted to `det`, as this seemed like an error in the Alpino corpus.[2]

The test revealed, that with the Alpino corpus an accuracy of 88.67% could be achieved, while with the Brown corpus even 93.31% of the words were tagged right. Through the result it may be assumed that due to the higher simplicity of English, a higher accuracy could be achieved.

7 Conclusion

During the practical a POS tagger with an underlying first-order HMM was developed. From the tagset experiments it can be concluded that finding a good size for a tagset is a complex task not only because some of the diversity gets lost due to reductions or some tags may be merged better together than others but also because merging certain tags may have unpredictable consequences for the accuracy of other tags.

However, in section 5 it was shown, that the size of the training set has a significant impact on the accuracy. Nonetheless, the size of the training set is no cure-all as for larger training sets the accuracy does increase significantly anymore and finally converges to a certain maximum.

In section 6 an experiment was conducted which showed that for an English test set a higher accuracy could be achieved than for a Dutch test set with similarly sized training sets and tagsets. However, an experiment with two identical tagset for both languages could produce an even more evident result.

[2]The conversion table of the tagsets can be found in the appendix

A Conversion list

from	to
*	*
ABL	ab
ABN	ab
ABX	ab
JJ	adj
JJR	adj
JJS	adj
JJT	adj
QL	adv
RB	adv
RBR	adv
RBT	adv
TO	comp
AP	det
AT	det
DT	det
DTI	det
DTS	det
WDT	det
WPO	det
WPS	det
WRB	det
FW	FW
NN	noun
NNS	noun
NP	noun
NPS	noun
NR	noun
CD	num
OD	num
RP	part
IN	prep
PN	pron
PP	pron
PPL	pron
PPLS	pron
PPO	pron
PPS	pron
PPSS	pron
’	punct
”	punct
–	punct
(	punct
)	punct
,	punct
.	punct
:	punct
“	punct
EX	tag
BE	verb
BED	verb
BEDZ	verb
BEG	verb
BEN	verb
BER	verb
BEZ	verb
DO	verb
DOD	verb
DOZ	verb
HV	verb
HVD	verb
HVN	verb
HVZ	verb
MD	verb
VB	verb
VBD	verb
VBG	verb
VBN	verb
VBZ	verb
CC	vg
CS	vg