Natural Language Processing: Assignment 2

Shifei Chen

---

1 POS-Tagging

My tagger achieved an accuracy of 93.10% by using parameters `-t 3 -f 5 -s 5`. Here are some errors my tagger made.

1. "From the AP comes this story."

"AP" was mistagged as a NOUN rather than a PRON. This is pretty obvious since AP is the name of a news agency.

2. "The sheikh in wheel-chair has been attacked with a F-16-launched bomb."

"F" was tagged as a NOUN in the gold standard and a PRON in my result. I think both of the tags are not correct. Here "F-16-launched" should be treated as a single adjective. If we have to separate them, I would believe that "F" is a pronoun as it is a part of the model name of a plane.

3. "US troops there clashed with guerrillas in a fight that left one Iraqi dead."

My tagger believed the word "Iraqi" is an ADJ rather than a PRON. An English speaker shouldn't make that mistake since he can see clearly that the clause "left one Iraqi dead" is the consequnce of the fight and because of that, "one Iraqi" is a person and "dead" is an adjective describing him. My tagger might not be able to understand the meaning of the sentence. It might simply looked at the words around "Iraqi" and believed this fit the structure of num. + adj. + noun.

4. "Xinhua alleged that 'Many of the Iraqis, who suffer ...'"

The word "that" should be a SCONJ instead of a DET because of the word "alleged" and the fact that everything inside the quote after "that", from a semantic prospective, is the object of Xinhua agency's allegation.

5. "2015 is going to rock!"

My tagger tagged "to rock" as a preposition and a noun, respectivelly. However in the golden standard they are a particle and a verb. We know "rock" can be a noun or a verb but in this case, a verb makes more sense semantically. Hence the word "to" should be a particle rather than a preposition.

In general, I think in my tagger only part of the mistakes are considered to be genuinely ambiguous, for example ADP vs. PART. There are still plenty of cases that are not ambiguous to human beings at all. The tagger is restricted to look forward or backward only N words (specified by the parameter `-t`) therefore it cannot have a good understanding of the overall context, especially in sentences consist of clauses.

Another issue is the lack of semantics understanding. There is a sentence in our corpus goes as below and it made my tagger confused.

... they hear a company who's stated goals include "Don't be evil," ...

My tagger tagged "evil" as a noun while our golden standard marked it as an adjective. Both of them make sense, although personally I would choose ADJ because I believe that company is Google. In senteces like "I'm going to work.", "to" being a particle or a preposition makes sense in either way since "work" can mean the action (verb) or the place (pronoun/noun).

However our golden standard is not prefect as well. For example,

... the price would be too high for investors to make a real profit.

In this sentence ”for” should be a prepostion, not a subordinating conjuction. There is no clause in this sentence.

I found several tagsets for my mother language, Chinese: the Chinese Penn Treebank POS tagset[1], an SVMTool-Based Chinese POS Tagger[2], FudanNLP[3], etc. Here we take a closer look at the Chinese Penn Treebank tagset. Comparing to its English siblings, the Chinese tagset has 33 tags. The word “把”, which means ”make sth. to do” and the word “被”, which marks a passive voice, are separated from other verbs and prepostions since their identities are still highly controversial. Another intresting thing I have noticed is that “的”, “地” and “得”, which are the three most common particles, are categorised into DEC, DEG, DER and DEV. This is a reasonable choice as their appearence usually gives people hints about the part-of-speeches of words around them, like “得” usually indicates the word it follows is always a verb and the word after it is usually an adverb.

Therefore I hold the idea that tokenization doesn't necessarily has to be done before tagging in some languages like Chinese, though most tagging algorithms assume that a process of tokenization has been applied to the tags[4, p. 157]. Of course tagging could benefit from tokenization because it splits words and gives information about word boundaries, sentence boundaries, average word length, etc. On the other hand, tagging could also help improve tokenization as different part-of-speeches will have impacts on tokenizaiton, for example distinguishing ”.” as a punctuation from a symbol of abbreviation. Another example is in languages like Japanese, its particles usually contain rich information about the sentence structure and semantics. In sentence “好きだ”, when “だ” appears the end of the sentence usually it means it is an auxiliary verb and we can therefore separate it from the other parts of the sentence. It also tells us that “好き” here should be a noun or a na-adjective (It actually means ”like” and in Japanese ”like” is a na-adjective). In conclusion, tagging does not need to be done after tokenization and I believe it is better if they can be carried out simultaneously.

Finally in Lab 6 we did an invesitigation on key sequences and predicted words in mobile phone inputs. By definition HMM models should always have a sequence of T observations (or signals) $O_1, O_2, O_3, ..., Q_T$, and a set of N states $Q_1, Q_2, Q_3, ..., Q_N$[4, p. 177]. If we observe the sequences of number keys then our states will be the letters of the words because they are what we see from the surface. Our signals will be the numbers as they are not what we can directly observe and they are hidden behind the predicted words. On the other hand, when we apply HMM models to POS tagging, what we see is a sequence of words and what we need to find out is the sequence of tags corrsponding to them. Words are the signals and tags are the hidden states.

2 Lemmatisation

2.1 Lemmatizer Analysis

After tunning my lemmatizer to achieve a score of 94.71%, here are five of the remaining issues.

1. Irregular Nouns/Verbs

Some nouns and verbs do not follow the inflection rules and because of that I have to use the method of exhaustion to tackle them. I have built a dictionary in my lemmatizer but it is still far from compelete. And as the lab instruction and the textbook had pointed out[4, p. 55], a lexicon should always exist in a more sophisticated lemmatizer.

2. Overlemmatisation

Some words look very much like they are already in a inflection form but actually they are not. For example ”bring” was mistaken by my lemmatizer to believe it is in its progressive form. Again a comprehesive lexicon will help us solve this problem. But it could also be beneficial if we provide more information by tagging different forms of verbs/nouns with different tags, like VB/VBG/VBD in the Penn Treebank tagset.

3. Constant Doubling

For words like ”cut”, ”big” we have to repeat the final constant before make any conjugation. The general rule is that if a single vowel is surrounded by two constants then the last constant should be repeatted. Any modern lemmatizers should have this rule built in.

4. Different Meanings with Singluar and Plural Form

Some nouns can have slightly different menaings with their singular and plural forms, e.g. ”troop” means a small group of soilders while ”troops” usually means the army in general. In lemmatisation

it is hard to tell the difference bewteen these two words as we usually work with one single word at a time. We lack the context to judge its semantic. One possbile solution I can think of is that we can mark their with different tags, e.g. NN and NNS, just like the overlemmatisation problem.

5. The Hidden "e"

The is the most common problem in my lemmatizer and it relates to pronounciation as well. The "e" in words like "take" changes the pronounciation of the vowel before it hence it is necessary to be restored during lemmatisation. But when I was thinking about how to program this rule to restore the hidden "e" I found that it is exactly the same rule to deal with a closed heavy syllable at the end of the word–a vowel surrounded by two constants. The hidden "e" case usually happens in longer words (more than 4 letters) and we can use that feature to distinguish them together with a big lexicon.

2.2 FST Diagram

My FST diagram goes as the picture below.

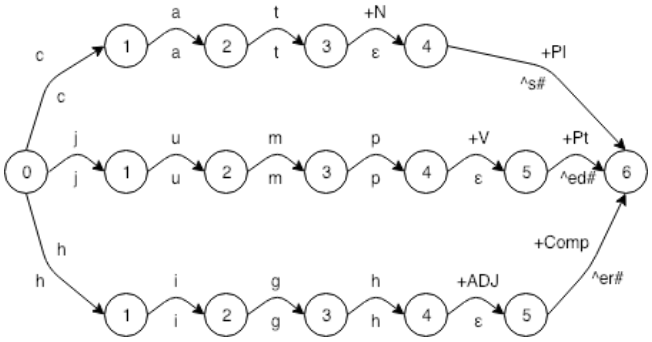

Figure 1: FST for "cats", "jumped" and "higher"

We have three words "cats", "jumped" and "higher" on the surface level. Their stems are "cat", "jump" and "high", respectivelly. We also need to know their lexical levels to draw this diagram. For example for "jumped" it is "jump +V +Pt(Past tense)". Then I just expand the Fig 3.13 on the textbook[4, p. 61] with some necessary changes to fit both verbs and adjectives and I got the diagram above.

2.3 Tagging in Morphologically Rich Languages

English is not a morphologically rich language, therefore it has two advantages compared with other highly inflectional languages like Finnish. First, the roles in the sentence of each word is much less, which means they usually carry fewer tags than the ones in morphologically rich languages. As a result we can see that English tagsets have way less tags[4, p. 162]. Another issus is that, during tagging process different inflections of the same stem will be treated as different words, hence highly inflectional languages will have a higher precentage of unknow words. If we recall that an HMM tagging algorithm using trigram tags will often has problems with data sparsity[4, p. 149], then here we can see that we will be in a deeper trouble as we now have not only unseen tag combinations in our test set but also way more unseen words. Moreover as modern taggers tend to give as many morphological tags as possbile and treat them as a single complex tag when tagging a morphologically rich language, there might be problems with unseen tags too. We need good interpolation algorithms to solve this problem.

References

[1] Fei Xia. "The part-of-speech tagging guidelines for the Penn Chinese Treebank (3.0)". In: *IRCS Technical Reports Series* (2000), p. 38.

[2] 王丽杰, 车万翔, and 刘挺. "基于 SVMTool 的中文词性标注". In: 中文信息学报 23.4 (2009), pp. 16–22.

[3] Xipeng Qiu, Qi Zhang, and Xuanjing Huang. “Fudannlp: A toolkit for chinese natural language processing”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2013, pp. 49–54.

[4] Daniel Jurafsky and James H. Martin. *Speech and Language Processing, 2nd Edition*. 2nd. Prentice Hall, May 2008. ISBN: 9780131873216. URL: `http://amazon.com/o/ASIN/0131873210/`.