Inflector

Yating Jing

April 30, 2015

1 Summary

In this assignment, I tried incorporating part-of-speech tags, dependency tree information and building a bigram model over the inflected forms, and experimented with different back-off ways. The best result I got is 0.70.

Table 1: Inflection Accuracy

Method	Inflection Accuracy
Unigram + POS Tags	0.66
Unigram + DepTree	0.64
Unigram + POS Tags + DepTree	0.67
Bigram + POS Tags + DepTree	0.70
N-gram without backing off	0.67

2 POS Tags

Simply incorporating part-of-speech tags into the unigram model would largely increase the accuracy of the inflector up to $46807/70974 = 0.66$.

More specifically, here I maintained two kinds of counts in the LEMMA dictionary. Besides counting different lemmas for each training words, I also incorporated tuples (*lemma, tag*) into the unigram LEMMA dictionary as keys for different words/forms and accumulated corresponding counts.

Then during the inflection calculation, use the test POS tags to sort entries for each test word/form. If a key (*test_lemma, test_tag*) does not exist in the corresponding LEMMA dictionary, then **back off** to use the key *test_lemma* and then sort the entries for each words. Finally pick the lemma in the most frequent entry in the dictionary as the most desirable candidate for each test word/form.

3 Dependency Tree

Then I tried using the information from the dependency trees to improve the inflection accuracy up to $45193/70974 = 0.64$.

More specifically, according to the dependency tree files, each word/form is a node in the dependency tree corresponding to the sentence the word belongs to, I incorporated the the parent and label information from the dependency tree into a tuple (*parent_index, label*). Again here I maintained two kinds of counts in the LEMMA dictionary. Besides counting different lemmas for each training words, I also incorporated tuples (*lemma,* (*parent_index, label*)) into the unigram LEMMA dictionary as keys for different words/forms and accumulated corresponding counts.

During the inflection computation, use the tree information to sort entries for each test word/form. If one of the keys (*test_lemma,* (*parent_index, label*)) does not exist in the dictionary, then **back off** to use the key *test_lemma* and pick the lemma in the most frequent entry in the dictionary as the most desirable candidate for each test word/form.

4 POS Tag + Dependency Tree

Incorporate both POS tags and dependency tree information yields the inflection accuracy of 47323/70974 = 0.67.

Here maintain four kinds of keys for each word/form in the LEMMA dictionary during count accumulation on the training data.

1. key_1: (*lemma, tag,* (*parent_index, label*))
2. key_2: (*lemma, tag*)
3. key_3: (*lemma,* (*parent_index, label*))
4. key_4: (*lemma*)

During the inflection computation on the test data, **back off** sequence when sorting entries for each word/form is then $key_1 \rightarrow key_2 \rightarrow key_3 \rightarrow key_4$.

5 Bigram Model

Then based on the model in section 4, i built a bigram model on the inflected forms of the training data. The inflection accuracy is then 49588/70974 = 0.70.

Here maintain eight kinds of keys for each word/form in the LEMMA dictionary during count accumulation on the training data, where *history* denotes the preceding word/form.

1. key_1: (*lemma, tag, history*)
2. key_2: (*lemma, tag,* (*parent_index, label*) *, history*)
3. key_3: (*lemma, tag,* (*parent_index, label*))
4. key_4: (*lemma,* (*parent_index, label*) *, history*)
5. key_5: (*lemma, tag*)
6. key_6: (*lemma,* (*parent_index, label*))

7. key_7: (*lemma, history*)
8. key_8: (*lemma*)

During the inflection computation on the test data, **back off** sequence when sorting entries for each word/form is then $key_1 \rightarrow key_2 \rightarrow key_3 \rightarrow key_4 \rightarrow key_5 \rightarrow key_6 \rightarrow key_7 \rightarrow key_8$, which yields the largest number of correct predictions according to the experiments.

6 N-gram Model

I also extended the above model to N-gram model, without backing off to lower-gram model, where the history is extended to multiple preceding words. 3-gram and 4-gram both gives the accuracy of 47323/70974 = 0.67.

References

[1] Philipp Koehn. *Statistical machine translation.* 2009. Cambridge University Press.