05.04 Prepositional Phrase Attachment 3/3

NLP

Parsing

Prepositional Phrase Attachment (3)

Algorithm 2a

```
If the preposition is "of", label the tuple as "low".
Else
    If the preposition is "to", label the tuple as "high".
    Else
        label the tuple as "high"
```

Some Observations

- First, even though the expected performance of rule 3 was 52%, its actual performance on the training set dropped to 39% after rules 1 and 2 were applied.
- In other words, these rules used up some of the information hidden in the data ahead of rule 3 and left it less useful information to rely upon.
- Even more, one can see that a better decision would have been to replace rule 3 with its exact opposite, label everything left at this stage as “high”, which would have boosted the combined performance.
- Algorithm 2a would achieve 5,527 + 2,172 + 7,714 = 15,413 correct decisions for an overall accuracy of 74% on the training set.
- Second, one cannot help but notice that Algorithms 2 and 2a each have only three rules. We can imagine a classifier with 20,801 rules, one per training example, each rule of the form “if the preposition is “of” and the nouns are such and such and the verbs are such and such, then classify the data point as the actual class observed in the training set”.

Some Observations

- Third, we so far reported performance on the training set.
- Can we project the performance on the training set to the test set?
- Let's start with Algorithms 1 and 3.
- Algorithm 1 labels everything as low attachment. It achieved 52% on the training set. We expect its performance on the test set to be similar. In fact it is 59% (1,826/3,097).
- This clearly demonstrates the variability of text across subsets of the data.
- In this case, this variability favors Algorithm 1 since its performance actually goes up when moving to the test set. In other cases (e.g., if we had swapped the training and test sets), its performance would have gone down.
- On average though, its performance on the test data is expected to vary around the performance on the training data.

Algorithm 3

If the preposition is “on” and the verb is “casting” and the first noun is
  “cloud” and the second noun is “economy”, label the phrase as “high”.

Else

  If the preposition is “of” and the verb is “opened” and the first noun
    is “can” and the second noun is “worms”, label the phrase as “low”.

  Else

   … 20799 more rules …

Some Observations 1/4

- Now, let's consider Algorithm 3. It achieved a very high performance on the training data (way above the "upper bound" achieved by humans).
- However, we will now see the meaning of the word *overfitting* in action.
- Algorithm 3 was so specific to the training data that most of the rules it learned don't apply at all in the test set.
- Only 117 combinations (out of 3032) of words in the test set match a combination previously seen in the training set.
- In other words, Algorithm 3 learned a lot of good rules, but it failed to learn many more. In fact, its accuracy on the test data is only around 4%.
- An alternative to Algorithm 3 would be to combine it with a default rule (just like rule 3 in Algorithm 2) that labels everything that Algorithm 3 missed as noun attachment.
- Unfortunately, even this algorithm (let's call it Algorithm 3a) would only achieve a performance slightly above the baseline (Algorithm 1) of 59% on the test data.
- The lesson to learn here is that, on unseen data, a simple algorithm (Algorithm 1) is much better than a really complicated one that overfits (Algorithm 3). Also, the combination of the two (overfitting + baseline) just barely outperforms the baseline itself and is nowhere close to competitive.

Some Observations 2/4

- Clearly this algorithm (Algorithm 3) would achieve close to 100% accuracy on the training set.
- Why “close to 100%” and not “100%”?
- It turns out that the training set there are mutually inconsistent labels for the same data point.
- For example, “won verdict in case” appears once as high and once as low attachment.
- There are a total of 56 such “discrepancies” in the training set.
- Some of them are caused by inconsistent annotators whereas others would require more context (e.g., the entire paragraph or document) to be correctly disambiguated.

Some Observations 3/4

- Next, let's see how algorithms 2 and 2a will fare on the test set.
- First, let's look at Algorithm 2.
- There are 3097 items to classify in the test set.
- Rule 1 correctly classifies 918 out of 926 instances of "of" (99% accuracy) while rule 2 gets 70% accuracy (234/332 correctly classified).
- Rule 3 achieves 810/1,839 = 44%.
- Overall the accuracy of Algorithm 2 on the test set is 63% (1,962/3,097).
- Again on the test data, Algorithm 2a outperforms Algorithm 2. Its Rule 3 gets 1,029/1,839 = 56% accuracy and the overall accuracy of Algorithm 2a on the test set is 70% (2,181/3,097).

Some Observations 4/4

- Let’s now summarize the performance of the five algorithms that we have looked at so far.

Algorithm	Number of rules	Training set accuracy	Test set accuracy
Algorithm 1: default	1	52%	59%
Algorithm 2: of/to + default	3	60%	63%
Algorithm 2a: of/to + better default	3	74%	70%
Algorithm 3: memorize everything	20801	near 100%	4%
Algorithm 3a: memorize + default	20802	near 100%	62%

What's Next?

- So far, so good. We have been able to go from 59% test set accuracy to 70% with two simple rules.
- What additional sources of information can we use to improve the algorithm?
- Here are some ideas:
  - use a few more *good* word features (e.g., more prepositions, perhaps some verb and nouns)
  - use clever ways to deal with missing information
  - use lexical semantic information (e.g., synonyms)
  - use additional context beyond the four feature types used so far.

05.04 Prepositional Phrase Attachment 3/3

PREP		
about	67	132
as	380	94
at	552	136
for	1136	1044
in	2251	1577
of	73	5534
on	666	550
to	2182	517
with	698	340
VERB		
bring	58	18
buy	217	126
cut	57	34
drop	84	15
follow	15	91
include	22	221
put	178	37

NOUN1		
company	61	33
director	6	51
increase	17	57
loan	23	9
rate	72	68
NOUN2		
asset	5	49
bank	33	31
board	22	23
client	23	9
company	68	204
day	57	14
year	427	106

What's Next?

- Let's first consider a combination of the first two ideas above: looking for ways to use all possible information that can be extracted from the training data.
- This is the approach that was used by Collins and Brooks (1995).
- Their method was based on a principle called backoff which is somewhat of a combination of all the algorithms used so far (e.g., Algorithms 1, 2, and 3).
- Backoff allows us to use the most specific evidence from the training data, when available but then make reasonable approximations for the missing evidence.

Collins and Brooks

- So what do we do? Collins and Brooks used the following algorithm.
- If a 4-tuple is available, use it.
- If not, combine the evidence from the triples that form the 4-tuple (looking only at the triples that include the preposition).
- If that is not available, look at the pairs, then the singletons, and finally use a default class.
- A 4-tuple is just a set of 4 features in a particular order, e.g., (verb, noun1, preposition, noun2).
- The matching term for 3 features is a triple; for 2 features it is a pair; and for 1 feature, the word singleton is used.

If the denominator of the next formula is 0, then use the classification for the 4-tuple

$$\hat{p}_4(H|v,n_1,p,n_2) = \frac{f(H,v,n_1,p,n_2)}{f(v,n_1,p,n_2)}$$

Else

If the denominator of the next formula > 0, then use the following estimate:

$$\hat{p}_3(H|v,n_1,p,n_2) = \frac{f(H,v,n_1,p) + f(H,v,p,n_2) + f(H,n_1,p,n_2)}{f(v,n_1,p) + f(v,p,n_2) + f(n_1,p,n_2)}$$

Else

If the denominator of the next formula > 0, then use the following estimate:

$$\hat{p}_2(H|v,n_1,p,n_2) = \frac{f(H,v,p) + f(H,p,n_2) + f(H,n_1,p)}{f(v,p) + f(p,n_2) + f(n_1,p)}$$

Else

If the denominator of the next formula > 0, then use the following estimate:

$$\hat{p}_1(H|v,n_1,p,n_2) = \frac{f(H,p)}{f(p)}$$

Else label as "low" (default):

$$\hat{p}_0(H|v,n_1,p,n_2) = 0$$

What's Next?

- The idea behind Algorithm 3 was quite reasonable – assume that if the same object appear again (as defined by the same set of four features), it will likely have the same tag.
- The problem with this approach is that there is not enough data in the training set to learn the likely classes of all possible combinations of features.
- Let's do the math. To cover all the data points in the test set, we'd need information in the training set for a total of 102,998,280,840 combinations (more than 100 Billion combinations)!
- How did we arrive at this number? It is simply the product of the numbers 1123, 1295, 52, and 1362, which are, respectively, the numbers of distinct verbs, noun1s, prepositions, and noun2s in the test set.
- It is impossible to label so much data and even if it could be done, there would be billions more combinations needed to cover a new test set.

Other Methods

- Zhao and Lin 2004 – nearest neighbors
- Find most similar examples – 86.5% best accuracy
- Similar to Zavrel, Daelemans, and Veenstra 1997 – memory-based learning
- Boosting – Abney et al. 1999
- Semantics – Stetina and Nagao 1997
- Graph-based method – Toutanova et al. 2004

Comparative Results

Algorithm	Test set accuracy
Algorithm 1: default	59%
Algorithm 2a: of/to + better default	70%
Best class per preposition	72%
Collins and Brooks	84%
K-nearest neighbors	80%
TUMBL	82%
Human average (4-tuples only)	88%
Human average (whole sentence)	93%

NLP