

CS224n Fall 2014 Programming Assignment 3

SUNet ID: tzhang54, jiajihu

Name: Tong Zhang, Jiaji Hu

1 Naive Baselines

1.1 All Singleton

In the all singleton naive baseline implementation, we simply iterated over all mentions and created a singleton entity for each mention. Each mention was marked coreferent with its own singleton entity.

1.2 One Cluster

In the one cluster naive baseline implementation, we created an entity at the very beginning. When we iterated over the mentions, we marked each mention as coreferent with the same entity.

1.3 MUC Analysis

The all singleton baseline got 100% MUC precision but 0 MUC recall. On the other hand, the one cluster baseline got 100% MUC recall but a lower MUC precision. This behavior made sense according to the definition of MUC. In all singleton baseline, each mention was clustered to it self, so there was never a wrong clustering. However, since the actual clustering did not include singletons, it was not able to capture any of the correct ones either, leading to a zero recall. One cluster baseline clustered all mentions together, so all pairs of mentions were connected by some edges, leading to a 100% recall. The precision was low because some irrelevant mentions were clustered.

2 Better Baseline

Our baseline implementation was based on head words of the mentions. We decided to use head words because from the experiments we found that they were a very strong indicator of the clustering. In training, we iterated over all pairs of mentions that were clustered together and recorded their head words in a set. For test mentions, we checked if the pair of head words of two mentions had appeared in the training data. If the pair did appear in training, we would cluster the two mentions together.

3 Rule Based

3.1 Overview

In our rule-based implementation, there was no training process. All the clustering happened under our control flow. We first iterated over all mentions and created a `Set<Mention>` containing only the mention itself. Then we iterated over the sets multiple times. Each time we would apply a new rule. Based on the rule we would decide whether two sets should be merged. If yes, we combined the two sets into a larger set. The order we applied the rules was from the high-precision low-recall rules to the low-precision high-recall ones. At the end of the flow, we iterated over all the remaining set of mentions and clustered all mentions from the same set.

3.2 Rules

After experiments, the final rules we applied were the following (in order):

3.2.1 Pronouns Matching

In our first pass over all the mentions, we checked if the head word of a mention was a pronoun. If it was a pronoun, put the mention in a separate set. Our first rule was to cluster pronouns that possibly refer to the same noun. Given two pronouns, we first checked if the mentions were quoted. If only one of the mentions was quoted, then they were not clustered. If neither or both of them were quoted and had the same speaker, we proceeded to the next step. In the next step, we checked if the pronouns had the consistent gender, plurality, and speaker. Next, if the two pronouns were both first or second person pronouns, we clustered them together since it was likely that "I", "You" from the same document refer to the same person. If the pronouns were both third person, we looked at their sentence indices. If the difference of the sentence indices was within 2, we clustered the pronouns.

3.2.2 Exact String Match

For all the nominal mentions, we first performed an exact string match to cluster mentions that had exactly the same glosses.

3.2.3 Head Token Lemma Match

Our second pass on nominal mentions was the head token lemma match. We tried matching the head words originally, but after experiments we decided to use the head token lemma instead.

3.2.4 Noun Pronoun Match

Our last pass on the data was to match pronouns with the nouns they were referring to. To achieve this, for a given pronoun and noun, we checked if they had consistent plurality. We also added the constraint that the noun had to appear in the document before the pronoun, but they could not be too far away (more than 2 sentences apart). We only performed this match on third person pronouns.

3.2.5 Alternative Rules

The alternative rules that we tried but decided not to use in the end included head word matching, POS matching, and NER tag matching.

3.3 Error Analysis

In the rule based method, we were able to take advantage over classifier based method in the following two points:

First, we were able to encode complex rules that could not be easily represented as features. One rule that significantly helped our performance was the relative position of mentions. When we clustered pronouns together, we checked their sentence indices to make sure they appeared in the same or adjacent sentences. When we match a pronoun to a noun, we looked at their mention indices to make sure the noun appeared prior to the pronoun.

Second, we were able to easily make multiple passes over the data, clustering the mentions using different rules in descending precisions. This control flow enabled us to apply different language rules very conveniently without conflicts.

4 Classifier Based

4.1 Overview

The classifier based system uses a set of positive and negative examples of coreferent pairs to train a maxent classifier. Every training example consists of a pair of mentions and a label of whether they are coreferent. For this part, we went along with only using the most recent coreferent mention as a positive example, and using only intervening mentions as negative examples.

4.2 Feature Engineering

4.2.1 Feature Design Process and Error analysis

For this part, we looked into features that we thought were relevant to classification. After we implement and test a feature, we look at the results, do error analysis and make modifications or extensions to make it better. Our feature design process was as follows:

To start, we only had the exact match feature. We were obviously bound to miss a lot of information using only this feature.

Looking at the actual output of the system, things were as expected. For example, all mentions “Hezbollah” were put in the same cluster due to the exact match feature. However, the system was making mistakes even where the mentions were pretty much the same, but had just a little difference like an extra “the”. For example: `the strikes --> !strikes; !these strikes`

From that, we had the idea to use the head word of the mentions. This turned out to be a high impact feature. By adding an indicator to see if the head words were a match, we immediately got a significant increase in score (MUC 0.70, B3 0.64). We also looked at

the lemma of the headword, which was also a good feature, but it was overlapped with the headword feature.

With that feature, we saw that `the strikes --> strikes; these strikes`, so that error case was solved.

However, there was still much to improve. We note that our system had 0.80+ precision but recall only around 0.6, which means we still don’t have enough features to identify good coreference pairs. We thought that intuitively, coreferent mentions should be closer rather than further away. For example, there was a sentence: `since {the majority of strikers} are jobless and {they} have found...`, and we were not putting the two mentions in the same cluster. To try to solve this, we added mention distance as a feature.

However, it turned out that neither the distance in terms of mentions or in terms of sentences improved the performance. In retrospect, this may be because of the way we are creating our examples. Our negative examples are all ones that are closer than the positive example for that mention. Therefore, the information is rather minimal.

To improve on this, we observe that in many of these cases, one of the mentions is a pronoun. It makes sense that if one mention is a pronoun (such as “they” in our error example), mention distance may be more relevant. By pairing up the mention distance and the pronoun indicators, we got a slight increase in F1 score.

After that, as suggested by the guidelines, we investigated the named entity type of the fixed and candidate mentions. After experimenting with the type, or whether they are the same, we concluded that this was not an impactful feature.

At this point, we still had the problem of

high precision and low recall – we weren’t extracting enough information.

We came across some interesting error cases that suggested that we could look at gender, speaker and plural compatibility:

are {five members} , for example ,
so {they} will ... for {them} that
{they} would not get

For this example, our system clusters the two “they”s, but misses the other two. However, we see that these are all plural.

part of {Hezbollah} and {its}
supporters ... strength of {this
organization}

For this example, all three were put in different clusters. However, we see that the plural and speaker features would have helped here.

Therefore, we added gender and plural features for the case of one mention being a name or noun and the other being a pronoun, or both mentions being pronouns. For the speaker, we only look at pronouns. Also, there is a decision of whether we want a positive indicator for the two mentions being compatible, or a negative indicator for the two mentions being incompatible. Intuitively, the positive feature is good for increasing recall, and the negative feature is good for improving precision.

As it turns out, both are good features, and their effects do not overlap. In addition, aggregating the three negative indicator features into one feature gave the same performance.

With the plural, gender and speaker features, our system was able to get right all four mentions in the first error example. It was also able to correctly cluster {its} with {Hezbollah} for the second error example, so our features really had the expected effect on improving performance.

In the end, our system uses the following:

- (1) Exact match (case insensitive)
- (2) Headword is same
- (3) Pair(Mention dist, Fixed is pronoun)
- (4) Pair(Mention dist, Cand. is pronoun)
- (5) Name-Pronoun gender incompatible
- (6) Name-Pronoun plural incompatible
- (7) Noun-Pronoun incompatible
- (8) Pronoun-Pronoun gender compatible
- (9) Pronoun-Pronoun plural compatible
- (10) Pronoun-Pronoun speaker compatible
- (11) Pronoun-Pronoun incompatible

4.2.2 Statistics

Table 1 shows some statistics of the classifier-based system on the development set using different feature combinations:

| Features | MUC F1 | B3 F1 |
|--------------|--------|-------|
| 1 | 0.642 | 0.601 |
| 1,2 | 0.707 | 0.645 |
| 1,2,3,4,5-7 | 0.755 | 0.679 |
| 1,2,3,4,8-10 | 0.740 | 0.678 |
| 1,2,3,4,11 | 0.743 | 0.684 |
| All | 0.792 | 0.721 |

Table 1: Results over feature combinations

4.2.3 Final errors and future improvements

characterizing systematic mistakes the system makes. You should describe how these features would be fixed, either with more features (and if so, what would they be?), or note

that they cannot be easily captured in our framework.

5 Results

The results for this assignment are shown in Table 2. As you can see,

6 Extra Credit

6.1 Multi-Pass Sieve Coreference System

When developing the rule-based system, we read the paper *Stanfords Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task*

7 Collaboration

| Algorithm | MUC | | | B3 | | |
|------------------|------|--------|----|------|--------|----|
| | Prec | Recall | F1 | Prec | Recall | F1 |
| Rule-based | 0 | 0 | 0 | 0 | 0 | 0 |
| Classifier-based | 0 | 0 | 0 | 0 | 0 | 0 |

Table 2: Results for all systems