

# 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, and the one cluster baseline got 100% MUC recall and 74% MUC precision. This made sense because in all singleton baseline, each mention was clustered to itself, so there was never a wrong clustering. However, since the actual clustering did not include singletons, get any cluster right, 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 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, which was inspired by the paper mentioned in Section 6.

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.

The reason we focus on pronouns is because we saw that many mentions in our data were pronouns, and that we were getting many errors in these cases. The rule designs come from our intuition about language structure.

### 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 match-

ing 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 a constraint that the noun had to appear in the document before the pronoun, but they were not 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 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 ap-

ply different language rules very conveniently without conflicts.

Looking at the output, we see that we do very well with cases covered by the rules, but fail when the rules don't cover the case. For example, **the nearby houses** and **nearby houses** and **the houses** don't even match, even though it seems like a simple case. If we had detailed labels of the named entity type, we can maybe check to see that they are all Locations, and put them in the same cluster.

Also, we did not go far into looking at the gender of pronouns. This would go along well with better entity type tags, since gender is most relevant if we can determine whether a mention is a person. Many useful rules can be done in this space.

## 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 the label. 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, as our score increased to 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.

Next, we note that our system had high precision but low recall, which means we 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 the mention distance alone did not improve performance. In retrospect, this may be because in our system, negative examples are all closer than the positive example for that mention. Therefore, the information is rather mixed.

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 men-

tion 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.

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} ... so {they}  
will ... for {them} that {they}...

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

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

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

Therefore, we added gender, plural and speaker features for names, nouns and pronouns. There was also a decision of whether we want a positive indicator for the two being compatible, or a negative indicator that the two are incompatible. Intuitively, the positive feature is good for increasing recall, and the negative is good for improving precision.

As it turns out, both are good features. In addition, aggregating the three negative indicator features into one 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 string match (case insensitive)

- (2) Headword is same (case insensitive)
- (3) Pair(Mention dist, Fixed is pronoun),  
Pair(Mention dist, Cand. is pronoun)
- (4) Name-Pronoun gender incompatible,  
Name-Pronoun plural incompatible,  
Noun-Pronoun plural incompatible
- (5) Pronoun-Pronoun gender compatible,  
plural compatible, speaker compatible
- (6) 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	0.755	0.679
1,2,3,5	0.740	0.678
1,2,3,6	0.743	0.684
All	0.792	0.721

Table 1: Results over feature combinations

#### 4.2.3 Final errors and future improvements

From the output of our system, we can still see a few errors that can potentially be fixed.

For example, {this organization} --> !{Hezbollah} ... shows that we are just not getting that “Hezbollah” is an organization – something that NER should be able to pick up. If we incorporated named entity tagging to make new features, mistakes like this should be avoided, and we could get substantially improved recall. A similar error

case that would benefit from this is {Yemen's President} with {him}; {he}; {his};.

Also, we still miss a lot of {this} references, so incorporating knowledge from the parse tree will definitely help in those situations, which is something we can improve on.

Finally, there are some errors that are quite understandable. For example, on the test set, our system was unable to realize that "The USS Cole" was a carrier, which is a ship. Things like this would benefit from world knowledge that the system obviously doesn't have. Incorporating this knowledge seems quite difficult for our framework.

## 5 Results

The results for this assignment are shown in Table 2 and Table 3.

We also see that the results on the test set is slightly worse than those on the dev set, which is expected, since we are testing and modifying our features to maximize performance on the dev set, and have never seen the test data set.

Algorithm	MUC			B3		
	Prec	Recall	F1	Prec	Recall	F1
Better Baseline	0.728	0.553	0.590	0.594	0.592	0.591
Rule-based	0.847	0.774	0.809	0.786	0.702	0.741
Classifier-based	0.862	0.733	0.792	0.825	0.640	0.721

Table 2: Dev results for both systems

Algorithm	MUC			B3		
	Prec	Recall	F1	Prec	Recall	F1
Better Baseline	0.708	0.440	0.543	0.676	0.545	0.604
Rule-based	0.829	0.708	0.764	0.800	0.676	0.733
Classifier-based	0.854	0.664	0.747	0.866	0.624	0.725

Table 3: Test results for both systems

## 6 Extra Credit

### 6.1 Multi-Pass Sieve Coreference System

When developing the rule-based system, we read *Stanfords Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task* and used their idea of running multiple passes over the data, each time using a different rule in descending precision order.

We implemented their control flow and applied our own rules, which gave us improved performance. Our rules for each pass were:

1. Exact String Matching
2. Head Token Lemma Matching
3. Pronoun Matching
4. Noun and Pronoun Matching

From the table, we could see in each pass, we got an improved score from a new rule.

Rules	MUC F1	B3 F1
1	0.560	0.605
1,2	0.723	0.706
1,2,3	0.733	0.720
1,2,3,4	0.761	0.728

Table 4: Performance for each pass