**Stanford CS224n – Project 3**

**By Patrick Manion & Jie Shen**

**1. Baseline Experiments**

For Coreference Resolution, we began by implementing two simple baseline models: a singleton model, which simply assigns every mention in the document to its own cluster, and the one cluster model, which assigns every mention to the same cluster, using the MUC and B3 metrics. The results indicate serious issues with MUC, which gives excessive high scores for one cluster and low scores for singleton. Both issues stem from MUC’s strategy of comparing links between the model and reference outputs, which are poorly defined when too few or too many links are present. B3 gives much more reasonable answers by evaluating the purity of each cluster and how well the clusters tie together the entities from the reference.

Two additional baselines were evaluated. The first was a simple Baseline model that merges mentions that are exact string matches of each other. The precision of this is much lower than might be expected driven primarily by pronouns, which are all clustered together despite potentially referencing numerous different entities. A BetterBaseline was also tested which combined entities that had exactly the same headword in the mention. This provides a sizeable boost

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MUC** | | | **B3** | | |
|  | **Precision** | **Recall** | **F1** | **Precision** | **Recall** | **F1** |
| **All Singleton (Dev)** | 100% | 0% | 0% | 100% | 25% | 40% |
| **OneCluster (Dev)** | 77% | 100% | 87% | 16% | 100% | 28% |
| **Baseline (Dev)** | 81% | 50% | 62% | 86% | 45% | 59% |
| **Baseline (Test)** | 81% | 39% | 53% | 91% | 44% | 59% |
| **BetterBaseline (Dev)** | 83% | 60% | 70% | 83% | 51% | 64% |
| **BetterBaseline (Test)** | 85% | 58% | 69% | 87% | 56% | 69% |

**2. Rule-based and Classifier-based Overview**

Expanding on the baselines, we first explored a rule-based model, which consists of a number of different rules that can build on each other to achieve the maximum performance. Generally, more precise rules are used early on to begin building meaningful clusters of entities and more lax rules are added at the end to take advantage of precise clusters. Lax rules are not added up-front as this can cascade errors through all the remaining steps and reduce performance.

The second model is a classifier-based model that uses a multiclass logistic regression model to learn weights to user-created features. The benefit of this approach is the model can help decide which features are important and include interactions between different features. However, the downside is it becomes more difficult to have features that build upon each other like a rule-based system, and it is also more difficult to include subject-matter expertise.

**3. Features**

For the rule-based system, we implemented a system similar in spirit to the multi-pass sieve Stanford system, which begins with high precision rules and moves to more relaxed ones later in the sieve.

We began our sieve with an exact string match, but learning from our baseline, we excluded any pronouns matches, which keeps precision very high. Afterwards, we implemented an acronym match that checks for the capitalized first-letters from one mention in another, which caught common errors in the training data like “the United Nations” and “UN” not being exact matches. After excluding any 1 letter “acronyms”, we found this rule gave high precision and a slight recall boost.

We then tried a simple rule to capture appositive constructs, but we found this actually reduced model performance due to many appositive-like constructs. For example, TV anchors often state their employer after their name like “Thelma Gutierrez, CNN,” and lists of items like “John, Tom, Mary, …”. Additional restrictions like requiring a named-entity match improved the rule but not sufficiently enough to include in the final model.

We then moved on to somewhat looser rules based around the head-words of the mention. First, we incorporated a match for mentions that had all the same words after excluding 25 of the most common stop-words. This helped capture obvious mistakes like “the Urban Institute” and “Urban Institute” not being merged. We also added a rule to merge clusters that had mentions with the same words *before* the head word. This captured a number of issues such as more descriptive early mentions like “the bill allowing food and medicine sales” not being matched with later mentions of the same topic like “the bill”, which occurs frequently in everyday discourse.

We then found that a rule combining mentions with the same headword boosted recall significantly and, despite a small decrease in precision, also boosted F1. Similar to the rule above, this helped capture mentions with more adjectives early on in the discourse that did not show up later like “the aged shuttle Discovery” being shortened to “the shuttle”. The F1 was further enhanced by matching headwords regardless of case, which helped solve errors in the text itself like (e.g. “the kursk” and “the Kursk”) as well as proper nouns being referred to later in the discourse as general nouns (e.g. “the Second World War” and later “the war”).

We then tried even more loose rules, but many did not work out as planned such a combining mentions with over 2/3 overlap in unigrams. Another example is a lemma match on the headwords, which ended up incorrectly combining a mention of a plural group with a mention of a single individual like “U.S. **officials**” with “One senior **official**”.

However,

Two additional features were surprisingly

U.S. officials in { 'U.S. officials' }

One senior official in { 'One senior official' }

At this point, we tried several looser rules that did not successfully improve performance including a match on the headword’s lemma to solve issues like

**4. Results**

Two primary models were evaluated as part of the project. The first is a simple PCFG with no parent information but keeping track of all prior sibling tags (infinite horizontal Markovization). The second model adds 2nd order vertical Markovization so each tag retains its parent relationship (e.g. S🡪NP becomes S^ROOT🡪NP^S). The results on the test data are shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1** | **EX** |
| **Base PCFG** | 81.44% | 75.70% | 78.46% | 20.65% |
| **+2nd Order Vertical Markov** | 83.71% | 81.33% | 82.50% | 32.26% |

The results show that the base PCFG already does fairly well, but adding 2nd order Markovization provides a significant boost of over 4% in F1.

Looking at the errors, one of the most obvious issues is that seemingly small differences in attachment can cause a huge drop in F1 due to how the metrics are calculated *especially* for very long sentences. For example, the sentence *“Stock-index futures contracts settled at much lower prices than indexes of the stock market itself.”* is parsed almost entirely correctly except for the final word *“itself”*, which is incorrectly labeled as an NP-PRP attached as the object of *“settled”*. This results in an F1 of 59.26% for this sentence. This would hopefully be an example where additional vertical context would help, but the same error exists in both models.

Another example of a common error is long prepositional phrases accidentally being broken up or wrongly attached by the model. For example, the sentence *“The balance is supplied by a host of smaller exporters, such as Australia and Venezuela”* has a long prepositional phrase starting with *“by”*. The base model, which has no vertical context, assumes that *“by a host”* and *“of smaller exporters”* are separate PP’s attached to *“supplied”* and that *“such as Australia and Venezuela”* is an adjective phrase after a linking verb. Luckily, in this case, the 2nd order Markovization captures the context and correctly nests everything under one prepositional phrase, which improves the F1 from 60.87 in the base model to 86.96.

The base PCFG ran through the test set in 6m 26s while the 2nd Order Markov model ran in 11m 31s, which are well below the guidance limits of 10 minutes and 20 minutes as a result of the optimizations mentioned in Section 1.

**3. Improvement Opportunities**

One key area for improvement is the storage of probabilities. Rather than the double[][][] array approach, small list could be used to store separate double[][] arrays for each row, which could eliminate the wasted space from the lower-left triangle that is never filled. If memory was more of a constraint, the Map approach could also be sped up using custom hashing functions rather than the default Java HashMap.

Another opportunity to improve both the memory required and runtime is to add a search heuristic such as beam search. The probabilities for many cells are so small that it’s very unlikely they will end up being the best at the end, so they could likely be dropped with only a small impact on overall performance. Careful tuning of this approach would be required as it may depend on the specific test dataset.

Another opportunity is to optimize or remove the best tree information stored in the three integer arrays. Similar to the probabilities, more careful data structure can eliminate a lot of wasted space in the lower left triangle. Second, the three numbers could be more carefully stored in a single array using bit-level encodings or even by hashing the numbers into buckets and evaluating the small number of collision at the end. Similarly, it’s also possible to store some subset of the three matrices or even the hash-code of the best rule itself and doing a small number of calculations at the end depending on the specific implementation.

**4. Model Extensions**

Several extensions to the base models were implemented. First, an additional level of 3rd order vertical Markovization was tried, which showed a significant improvement in performance. In addition, a test was conducted on whether to include the pre-terminal tags in the Markovization as this seems a potentially easier task for the parser. The results are shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1** | **EX** |
| **Base PCFG** | 81.44% | 75.70% | 78.46% | 20.65% |
| **2-Vertical** | 83.71% | 81.33% | 82.50% | 32.26% |
| **2-Vertical**  (excluding Pre-Terminals) | 82.34% | 79.61% | 80.95% | 27.10% |
| **3-Vertical** | 83.52% | 84.41% | 83.96% | 37.42% |
| **3-Vertical**  (excluding Pre-Terminals) | 82.85% | 82.16% | 82.50% | 27.10% |

The results show that 3rd order vertical Markovization provides a material boost even above the 2nd order vertical Markovization – boosting F1 by almost 1.5%. The results also indicate that the pre-terminal vertical Markovization is very important, providing almost half the boost from the Base PCFG to the 2nd order vertical Markov model. Interestingly, including the pre-terminals in the vertical Markovization add 1.55% in the 2-Vertical model and 1.46% in the 3-Vertical model, which suggests that most of the benefit from the 3rd order vertical Markovization comes from splits above the pre-terminal. This aligns with the intuition that this is a somewhat easier part of the parsing.

Finally, different settings for the horizontal Markovization were tried comparing the default ∞-Horizontal approach with versions that discarded some of the prior tag information if it became too distant.