(Author: Marcus Walker)

**Hey, Wait!**

If you haven’t read README.txt, check that out first! This file refers to the content of README.txt, and may not make any sense at all if you haven’t already read that document!

**Input**

Input to the models was all the same: file.arff, which is a version of the original scene-delimited training data that has been subjected to our formatting, cleaning, and feature engineering scripts. It has 111,224 instances (13,486 of which are reference instances used to train and evaluate the model) and 18 features (10 of which we engineered. The features are:

* SeasonEpisode The season and episode this token is from
* Scene.ID The scene, within episode, this token is from
* Token.ID Unique identifier for token
* Word.Form Actual token from script
* POS.Tag Token’s part of speech
* Lemma Lemma of token
* Speaker Character who says token
* **Entity.ID** Referent of reference token **(Class Variable)**
* *Sentence* Unique identifier for sentence
* *Lemma\_Reference\_Uncertainty* Uncertainty value (1-3) of lemma
* *Speaker\_MFR* Speaker’s most frequent referent
* *Prev\_Speaker* Previous speaker
* *Next\_Speaker* Next speaker
* *Prev\_Lemma* Previous lemma
* *Second\_Prev\_Lemma* Lemma previous to previous lemma
* *Next\_Lemma* Next lemma
* *Second\_Next\_Lemma* Lemma after next lemma
* *Implied\_Gender* Implied gender of pronoun

(italicized features engineered by us)

**Model Results**

As mentioned in README.txt, our highest performing algorithm was the J48 Java implementation of the C4.5 decision tree algorithm (see *J48 Algorithm* section of README.txt). Our baseline for comparison to a simple standard was Naive Bayes. J48 and Naive Bayes summary results are compared below, and selected detailed output is provided for J48. Finally, a really terrible picture of the decision tree is included to provide small amounts of insight and convince the reader to look at the detailed output generated by runit.sh.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Precision | Recall | F-Measure | Kappa |
| J48 | 0.878 | 0.874 | 0.874 | 0.8432 |
| Naïve Bayes | 0.636 | 0.602 | 0.595 | 0.4811 |

**J48 Selected Detailed Output**

=== Summary ===

Correctly Classified Instances 11793 87.4462 %

Incorrectly Classified Instances 1693 12.5538 %

Kappa statistic 0.8432

Mean absolute error 0.0552

Root mean squared error 0.1706

Relative absolute error 23.8312 %

Root relative squared error 50.1361 %

Total Number of Instances 13486

Ignored Class Unknown Instances 97738

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Msr. MCC ROC Area PRC Area Class

0.838 0.014 0.887 0.838 0.862 0.845 0.903 0.170 Chandler Bing

0.829 0.006 0.933 0.829 0.878 0.868 0.907 0.220 Joey Tribbiani

0.871 0.014 0.882 0.871 0.877 0.862 0.947 0.749 Monica Geller

0.935 0.096 0.834 0.935 0.882 0.818 0.911 0.469 Other

0.845 0.005 0.942 0.845 0.891 0.883 0.941 0.680 Phoebe Buffay

0.852 0.015 0.885 0.852 0.868 0.850 0.882 0.159 Rachel Green

0.828 0.017 0.888 0.828 0.857 0.836 0.869 0.166 Ross Geller

Weighted Avg. 0.874 0.041 0.878 0.874 0.874 0.843 0.907 0.380

=== Confusion Matrix ===

a b c d e f g <-- classified as

1282 4 16 146 13 28 41 | a = Chandler Bing

30 1049 10 133 0 21 22 | b = Joey Tribbiani

11 14 1235 94 13 22 29 | c = Monica Geller

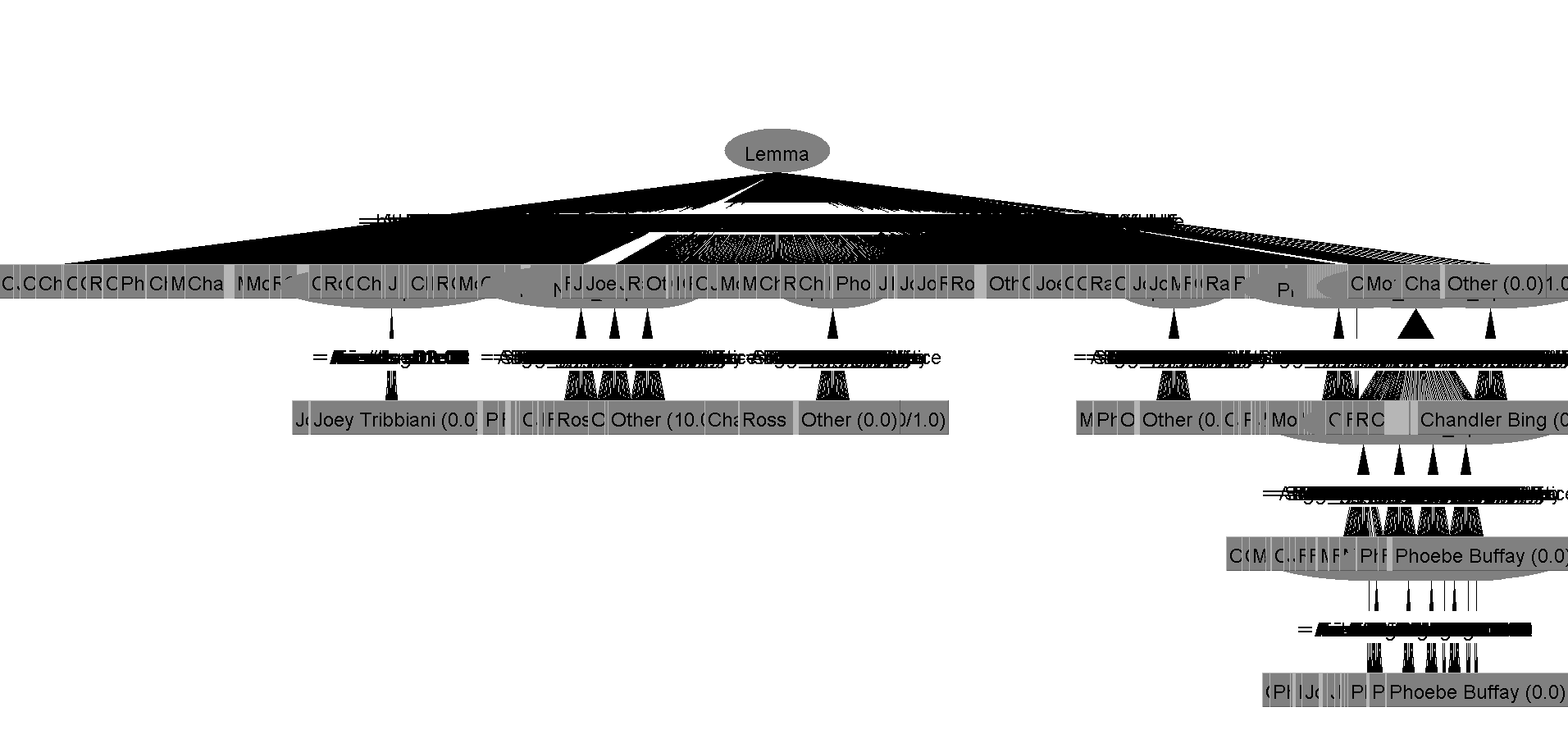
53 30 73 4283 22 70 51 | d = Other

17 9 26 89 991 26 15 | e = Phoebe Buffay

20 4 21 154 10 1410 36 | f = Rachel Green

32 14 19 236 3 16 1543 | g = Ross Geller

**Really Terrible Picture of the Decision Tree**



(7307 leaves. That’s worth at least one “yike”.)

**Results Analysis**

*(Note: some analysis refers to the full J48 output generated by Weka. For reasons that are probably clear, the unabridged content that 173 page document is not included here in full.)*

The task required that we classify each reference instance into one of seven categories: one each for the six major characters (Chandler Bing, Joey Tribbiani, Monica Geller, Phoebe Buffay, Rachel Green, Ross Geller) and a seventh, “Other” category for all other characters. That’s exactly what these results suggest we’ve done! Hooray!

Comparison between J48 and Naive Bayes shows that we are far outperforming a relatively simple baseline. The difference in F-Measures suggests that our model is doing more than just capitalizing on the distribution of the class variable: J48 beats Naive Bayes significantly in both precision and recall across classes. Clearly, we’re capturing patterns in the data that allow us to make strong predictions.

It can also be seen that the model is discerning: the average True Positive rate is 87.4%, while the average False Positive rate is 4.1%. This is expanded by the area under the ROC (receiver operating characteristic) curve, which measures the false positive rate against the true positive rate as the discrimination threshold is varied. A high area, such as the 90.7% we see in this model, suggests that the model’s discernment is not just a product of its discrimination threshold, but that it is fundamentally good at avoiding false positive classification as a result of the ruleset generated by the decision tree.

Looking more closely at the decision tree itself (see the J48 output generated by runit.sh for more detail), we can get some idea of why the model is performing the way it is. The most influential feature is at the root of the tree: Lemma. This is as we expected; different words should be narrowed down in different ways, and the lemma is a low level hypernym that is more likely to be discriminating along this axis than the raw word form.

By the nature of the problem, the vast majority of lemmas are never used to refer to a character. (For example, the lemmas *compensate* and *a-gogo* never refer to a character, because that wouldn’t make any sense!) Each of these lemmas is given the naive rule “classify as *Other*”, because *Other* is the most frequent class and the lemmas have to be given some kind of rule. These rules are never used, but they’re super good at filling 173 pages of output.

Of the remaining lemmas, many of them are also given classification rules straight away. For example, the lemma *Gunther* is given the rule “classify as *Other*” because *Gunther* is the name of a minor character and the model always sees *Gunther* paired with the classification *Other*. Interestingly, in this type of case, the model is doing more work than it needs to: since any instance not classified as one of the six major characters with either be classified as *Other* or not classified at all, it could restrict itself to classifying the major six and throw everything else indiscriminately into the *Other* pool. Interesting to consider, though since the actual modeling step takes 1.64 seconds in total, we don’t begrudge it the little extra effort it expends here.

Similarly, the model is eager to jump to conclusions when it sees lemmas such as *Joe* or *Joey*. When the names of the six major characters come up, the model generates very simple one-step classification rules that are rarely wrong. (One example is the lemma *Joseph*, which the model hastily classifies as *Joey Tribbiani* in all cases. This is the correct classification 18 of the 20 times *Joseph* appears, and incorrect twice- when referring to Joseph Stalin!)

The real work is done in just a handful of interesting lemmas. The best example is the lemma *you* (and related lemmas *ya* and *yourself*). If the lemma is *you*, the model always checks who the next speaker is. In many cases it is confident making a classification based only on the next speaker- for example, if the lemma is *you* and the next speaker is *Amber*, that’s enough information to (mostly correctly) determine that the referent is *Joey Tribbiani*.

If the next speaker is *Chandler Bing*, for example, the model will then go on to check who the previous speaker was. Here again, in most cases, the model will make a classification at this point. (Lemma = *you*, Next Speaker = *Chandler Bing*, Previous Speaker = *Monica Geller*) results in a classification of *Monica Geller*.

However, in some cases it will go further. For (Lemma = *you*, Next Speaker = *Chandler Bing*, Previous Speaker = *Rachel Green*), the model will check the season and episode that this line of dialogue is from. This would seem to reflect sensitivity to a difference in distribution based on which characters are more active or present in a particular episode, and seems like a totally reasonable way to make a distinction at this level of resolution and uncertainty. Nice going, model!

As mentioned before, though, the model does this work separately for *you*, *ya*, and *yourself*, though it is clear to a human observer that these lemmas (or at least, *you* and *ya*) should be considered to be the same. In a less relevant example, there are separate lemmas for *hm*, *hm.*, *hmm*, *Hmm*, *hmm.*, *Hmm.*, *Hmmm*, and *hmmm.* Clearly, there is some combining of lemmas that could be done, which suggests that the marginal return on an additional layer of hypernym discovery might be high, and could potentially eke out another percent or two in overall model accuracy.

The other place where our model fails most clearly is in how it handles certain pronouns that usually refer to *Other* characters. The pronoun *he*, for example, is so frequently seen to refer to *Other* characters that the system classifies any instance of *he* as *Other*. This classification is correct 773 times, but incorrect 203 times. This seems like a good ratio if all you’re looking at is the lemma *he*, but it accounts for approximately 12% of the incorrect classifications the system makes.

**Conclusions?**

We identify a few key takeaways from these results.

1. A decision tree system works well as a strategy for dealing with a small amount of training data with this kind of task. An 87.4% average F-Measure is pretty good, and beats a baseline like Naive Bayes handily.
2. The lemma of the token, when trying to identify referents, is super important. Investment of time and effort into getting lemmas bundled together properly is likely to pay off, and there are probably a few different ways you could do some hypernym discovery to improve final results.
3. The audience matters! After the lemma, the most influential features were the previous and next speakers, while the previous n and next n lemmas weren’t considered in the final tree at all. Even the current speaker was less influential overall than the previous and next speakers, suggesting an “eye of the storm” effect where there is more information contained in who a speaker’s words might reach than in the identity of the speaker themself. (Caveat: obviously not true for lemmas such as *I*  or *me*)
4. You can build a strong model with a very small number of engineered features. If the only features in this model had been Lemma, Previous Speaker, Next Speaker, and Current Speaker, relatively little accuracy would have been lost. After resolving differences based on Lemma, the system of further resolution is approximately Markovian at the level of Speaker.
5. Super wide decision trees are not made to be visualized on standard, letter-oriented pages.