Information Extraction from Recipes

Rahul Agarwal and Kevin Miller

Introduction

The goal of this project was to extract information from cooking recipes into a machine-interpretable format. In particular we wanted to identify which actions are applied to which ingredients, and possibly identify which utensils are being used. One potential application could be for an application that has to answer questions about the nature of the preparation of the food. Another, more far-fetched application could be to have a program learn correlations between ingredients, the actions that are applied to them, and the sequence of those actions in order to be able to novel recipes that are plausible. In this project, our goal is to reduce the preparation steps of a recipe to ACTION, INGREDIENT, UTENSIL groups. For instance, the sentence "*Chop the beef and place it in a large skillet*" should be reduced to following two groups: CHOP(BEEF); PLACE(BEEF) (SKILLET). We use a combination of Named Entity Resolution (NER) and ideas from Semantic Role Labeling (SRL) in order to get this reduced form of the recipe.

We worked with semi-structured XML recipe data. Although certain classes of sentences were identified (ingredient descriptions vs. preparation steps), the majority of the data is still presented in natural language format. An example recipe is shown in the appendix.

There are many things that make working with recipe data difficult. One it is unclear which words relate to actions, foods, and other relevant categories. Although all actions are verbs not all verbs in the recipe directions are relevant actions that act on ingredients. For example _____. In addition, it's not clear which words are related to food items or not. For example, in the phrase "*½ cup chopped pecans*", only the word "*pecans*" is a relevant food word and "*chopped*" is a description of the state of the nuts. However, in the ingredient description "*1 cup sour cream,*" "*sour cream*" refers to a particular foodstuff, which is different from cream that is in the state of being sour. To solve this problem, we train an MEMM with specially labeled training data and domain-specific features to classify words into semantic categories.

Another challenge is that recipe data contains mostly imperative sentences. However, most of the available data used to train parsers rely on large treebanks which contain data from the Wall Street Journal and biomedical texts, which contain mostly declarative sentences. Therefore, syntactic and part of speech data obtained from these parsers is less reliable than for other types of text. We use the Stanford Parser in this project, which performs reasonably well for imperative sentences but still makes mistakes (see analysis below).

Recipes also present a set of interesting co-reference resolution problems. An ingredient may be referred to in multiple different ways, all using the same key ingredient word ("*1/2 c Dark seedless raisins*", "*raisins*", etc.). However, ingredients from the ingredient descriptions are often referred to by hypernyms of the words used in the descriptions (ex. "*1/4 cup chopped walnuts or pecans*" can be referred to later as "*nuts*"). Both of these cases are quite common in general coreference resolution problems [citation needed]. One challenge that is unique to recipe data is that entities such as ingredients are often grouped, mixed, or transformed together to create new entities. After this transformation, the group of ingredients is often referred to using the name of the new entity, without explicitly mentioning that it is a combination of previously mentioned entities. For example take the following recipe:

```
<RECIPE>
<TI>Empanadas De Fruta</TI>
<IN>6 c Flour</IN>
<IN>1 Tblsp Sugar</IN>
<IN>1 1/2 c Water</IN>
<IN>1 tsp Salt</IN>
<IN>3/4 c Shortening</IN>
<IN>20 oz Mixed dried fruit</IN>
<IN>3/4 c Sugar</IN>
<IN>1 tsp Cinnamon</IN>
<IN>1 c Pinon nuts(other nuts okay)</IN>
<IN>1/2 lb Raisins</IN>
<IN>1/2 tsp Cloves</IN>
<IN>1/2 tsp Nueg</IN>
<PR>Mix flour, salt and sugar; add shortening and mix well. Add beaten egg to water and then to
dry mixture. Dough should be soft but not sticky. Make small balls of dough. Roll out to 5 inches
diameter and 1/8" thick. Place a heaping teaspoon of filling on half of rolled out dough turning
other half of dough over and pressing edges together. Pinch edges between thumb and fore finger
giving the dough a halfturn. Fry in deep fat until golden brown. Drain. To make fruit filling, add
enough water to cover the dried fruit and raisins. Cook over low heat until tender. Add sugar, spices,
and nuts. Mix until well blended. A 21 oz. can of fruit pie filling can be substituted for the fruit
filling.</PR>
</RECIPE>
```

This example illustrates a number of the interesting coreference challenges present in recipes. One is the fact that the word "*dough*" actually refers to the set of {*"salt", "sugar", "shortening", "egg"*, and "*water"}*. It would be worthwhile for an application to be able to understand this in order to trace the transformation of ingredients through the cooking process. Another challenge is the presence of references such as "*spices,*" which refers to the group of ingredients {*"sugar", "cinnamon", "cloves", "nueg"},* which is difficult to determine without additional world knowledge of what foods can be considered spices. The last reference challenge presented in this example is the presence of sentence fragments that don't explicitly state the arguments to their actions, such as "*Cook over low heat until tender*". In this case, the arguments to "*Cook*" can be found in the previous sentence ("*dried fruit*" and *raisins*"), and therefore are implied in next sentence. The presence of these complicated reference situations makes it hard for programs to understand the sequence of events present in a recipe. We try to address some of these in this paper.

Related Work

A related project to ours is a CS224N project by Michael Gummelt and David Brody from 2009 entitled "Converting In-N-Out Orders into a Structured Form". That project used Named Entity Recognition and PCFG parsing, with an intermediate (and somewhat hand-coded) representation between NER and PCFG. The data involved in (Gummelt & Brody, 2009) – In-N-Out orders – was much more regular than our data, but what is important is that the data in both projects is relatively idiosyncratic and very domain-specific, to the point that hand-coding was a viable option.

Most of the literature on coreference resolution that we found was not nearly as domain-specific as our problem and therefore did not involve our problem's constraints or exploitable structure. For example, (Lappin, 1990) used a grammar called "slot grammar" to do heavily syntax-based coreference resolution; our project, unlike (Lappin, 1990), cannot rely heavily on the accurate syntactic parsing, since the parser used was not trained on many imperative sentences. At the same time, unlike (Lappen, 1990), our project involves very specific cases of coreference, so we can use domain specific methods, such as LESK where the ingredients list is used as a dictionary.

Methods

Maximum Entropy Classification of Named Entities

We use an MEMM classifier, similar to that of PA3, to identify key words in the ingredient lists and preparation directions that are important for understanding the semantics of the recipe. Specifically we define the following labels:

- **AMOUNT:** Defines the quantity of some ingredient. Does *not* refer to lengths of time, sizes of objects, etc.
- **UNIT :** Specifies the unit of measure of an ingredient. Examples include "cup", "tablespoons", as well as non-standard measures such as "pinch".
- **INGREDIENT:** The main food word of an item *that is mentioned in the ingredient list*. Groups or transformations of sets of ingredients (such as "dough") do not fall into this category
- **DESCRIPTION:** A word or phrase that modifies the type of food mentioned, such as the word "chopped"
- **ACTION:** The primary action verb that is being applied to a set of ingredients. Note that this did not include gerunds such as "stirring constantly…"
- **UTENSIL:** An auxiliary object used by the action, such as a "bowl" or a "pan"

- **MIXTURE/GROUP:** This label specifies any food within the recipe that refers to multiple ingredients (such as "spices" or "dry ingredients"), as well as those that are mixtures or bi-products of ingredients ("dough", "sauce").

An example is shown here:

- 24 lychees
- 1 pk unflavored gelatine
- 1/4 cup cold water
- 1/3 cup milk
- 1/2 cup sugar
- 1 cup half and half
- 1 teaspoon lemon juice

Peel and seed lychees. Squeeze lychees through 2 pieces of cheesecloth to obtain 1 cup of juice. Sprinkle gelatine over cold water and let stand for 5 minutes. Scald milk, add soaked gelatine, and stir until thoroughly dissolved. Add sugar, mixing well…

Many of these classes cannot be distinguished from morphological features alone, so we must also include syntactic features as well. Additionally, because of the semi structured nature of our data, we can construct features that relate the ingredients list and the preparation directions of the recipe. We use the following features for our MaxEnt classifier:

1. **Current Word**. The simplest feature is just to use the word itself. Many words in our training set only appear with a single label.
2. **Previous Word, Next word.** This feature includes a special start token <S> and end token </S> for the beginning and the end of sentences.
3. **Previous Label.**
4. **Parts of Speech for current word, previous word, and next word.** These parts of speech are obtained using the Stanford parser. The results from the parser are pretty good, but due to the nature of our data can sometimes be inaccurate (see below)
5. **The appearance of numbers [0-9]**. Typically the only words that contain numbers is the amounts, so this feature serves to increase the accuracy of classifying amount words.
6. **Ends with "-ed".** This feature helps to distinguish description words, which often take the form of past participles such as "chopped" or "sliced".
7. **No vowels**. This feature helps to identify units, which often come in the form of abbreviations such as "c" or "tbsp".
8. **If the word is not contained in the list of ingredients.** If a word is not contained in the list of ingredients, then it cannot, by definition, be labeled as an INGREDIENT because the set of ingredients for a recipe is defined as those foods that appear in the ingredient list.
9. **If the words is not in the ingredients AND it is a noun.** This feature specifically targets groups and utensils. These both are typically nominal nouns, but don't appear in the list of ingredients.

That is, group nouns don't appear because they are combinations of the foods in the ingredients lists, and utensils, by the conventions of the recipe format, don't appear.

10. **Word Length**. The amount and unit words are typically small (< 4 characters), so using the length of the word gives some discriminative power for identifying these words.
11. **First occurrence**. This is a feature on the interval [0, 10] which measures the relative position within the recipe directions of the first time a particular word is seen. The value assigned to a word is just the rounded percentage of the total number of words in the directions that occur before the first occurrence of this word. For example, if the first mention of the word "*cake"* is 94% of the way through the text of the directions, it will be assigned a value of 9.

    The intuition behind adding this feature was that foods that are products of other words will most likely be mentioned later in the text, as they are constructed. Because this feature is targeted at utensils and group words, we only apply this feature to nouns that don't appear in the ingredients list.
12. **The appearance of "mix-" or "ingredient-".** Many of the words that word categorized as group words had the structure " ____ mixture" or "the ___ ingredients", where the blanks represent some kind of description of group (ex. "the beef mixture", "the dry ingredients"). We look for the occurrence of words with these prefixes in a radius of 3 words on either side of the current word.

We also tried to include syntactic information into features using the Stanford parser. However, information beyond part of speech tags, but this did not prove useful because of the unreliability of the parse trees we obtained (discussed below).

SRL:

We defined Semantic Role Labelling for our problem as follows: each predicate was an action taken in the recipe, and each predicate's argument would be a phrase spanned by a node in the parse tree and would be labelled as either a "Recipient" if it was one of the ingredients (or groups of ingredients) to which the action was applied or an "Agent" if it was one of the utensils used to carry out the action. For example, in the sentence "Poke the tomato with a fork", the correct Semantic Role Labelling would make "Poke" a Predicate that had "tomato" as a Recipient and "fork" as an Agent.

We chose to do SRL in this way for two reasons. First, this approach would help convert recipes (and perhaps other kinds of instructions) to a more machine-interpretable form – a set of actions , each one associated with a set of ingredients and utensils, would give a computer or robot most of the information it needed to carry the instructions described in the recipe. Second, grouping ingredients with action and utensils (and with each other) could be useful in resolving coreference; for example, such groupings would be useful in inferring that dough often refers to sets of ingredients that are mixed with beaters, that flour, butter, and water are often mixed with beaters, and therefore that flour, butter, and water are likely to be the ingredients jointly referred to by "dough". As another example, if "beef" and "meat" tended to co-occur most often with similar sets of utensils and other ingredients, one might infer that "beef" and "meat" likely referred to the same ingredient.

Our initial approach to SRL was to use a Maximum Entropy classifier, using features of both the NER labellings and the parse tree for a sentence, to classify candidate nodes as either having no role, having a reciepient role, or having an agent role, given a predicate in the sentence. We decided not to train our algorithm for filtering to get a set of candidate nodes or our algorithm for finding predicates; the first of these tasks is usually a pre-processing step, and the second can be done by simply selecting words (specifically preterminal, part-of-speech nodes in the parse tree) that NER labeled as actions. For candidate filtering, we only considered candidates that were POS-nodes or parents of at least one POS-node, since we knew that recipeints and agents would almost always be lower-level nodes in the parse tree.

One major hurtle for training SRL was a lack of annotated data; although we had many recipes, none of them had been previously given Semantic Role Labels, and we would not have the time or resources to do much labelling. Therefore, prior knowledge about SRL would have to be used in place of training data. One approach we briefly considered was to manually set the initial weights of the MaxEnt classifier used for SRL and then have it make guesses about SRL and ask a human (i.e. one of the authors) if those guesses were correct, and if not, what the correct role labellings were; as training data was accumulated in this way, an online version of MaxEnt would be used (specifically, counts would be stored and incremented, and the weights would be reoptimized on a regular basis). Hopefully, most of the guesses would be correct (due to a good choice for the initial weights), and so the human would not have to spend much time creating training data. Additionally, if time permitted, we could also do an online version of forward search by training several MaxEnt classifiers, each one with a different candidate feature added, and letting the one with the best F1 score make the guesses that would be displayed to the human.

We very briefly experimented with a classifier where the weights were manually set but realized that it would be better to hard-code rules for SRL in a more direct way than weights. With weights, certain rules, such as “no two labeled nodes can be ancestors of each other”, would be very difficult to express. Furthermore, SRL could easily be done with a few hard-coded rules as long as the syntactic parsing and NER-labelings were sufficiently accurate, and as NER continued to improve with more features (some of them syntactic), we decided that it would be best to give NER the entire burden of learning from features and use its (hopefully sufficiently accurate) results with those of parsing to get accurate SRL. The following SRL rules were used (in the form of a depth-first tree-traversal algorithm):

-for a given node:

--if n isn't at least as close to the predicate in question as it is to all other predicates, don't consider this node or any of its descendants (here “closeness” to another node is defined as the number of edges between node and the lowest common ancestor of node and the other node)

--if all of leaves spanned by the node have NER label “O”, don't consider this node or any of its descendants

--otherwise:

---if node is not a POS node or a parent of a POS node, then recursively call this procedure on all of node's children

---otherwise:

----if all leaves spanned by node have NER label “FOOD” or “GRP”, label node as Recipient and don't consider any of its descendants

----if all leaves spanned by node have NER label “UTENSIL”, label node as Agent and don't consider any of its descendants

----if neither of the above are satisfied, recusively call this procedure on all of node's children

Coreference Resolution:

To address coreference resolution between named ingredients (e.g. “raisins” and “dark seedless raisin”), we used LESK's algorithm on each argument from SRL, using the ingredients list as the dictionary.

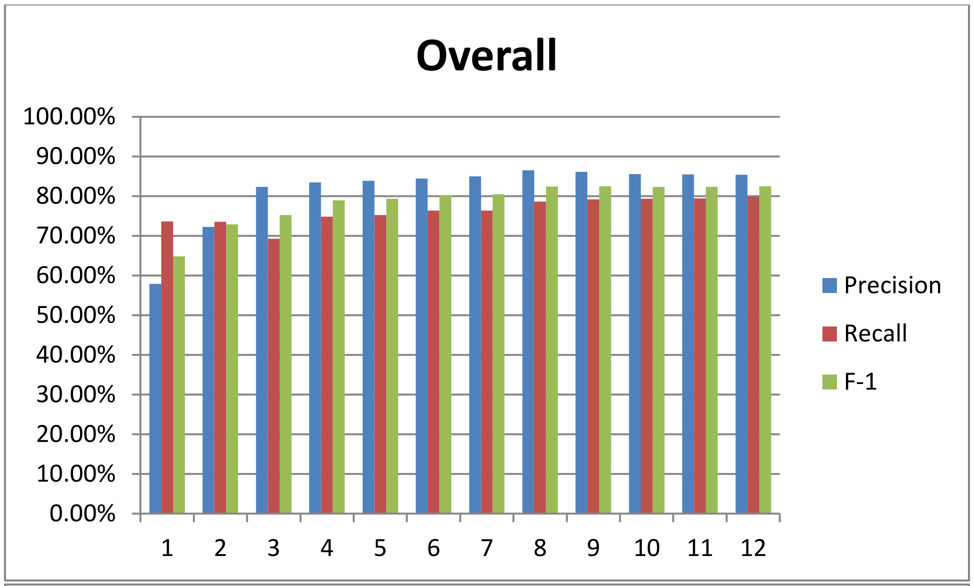
We also came up with a method to address coreference resolution between ingredients and groups containing those ingredients (e.g. “flour, water” and “dough”). We decided to approach this problem as a machine translation problem, where each set of predicates and arguments being equivalent to a sentence.

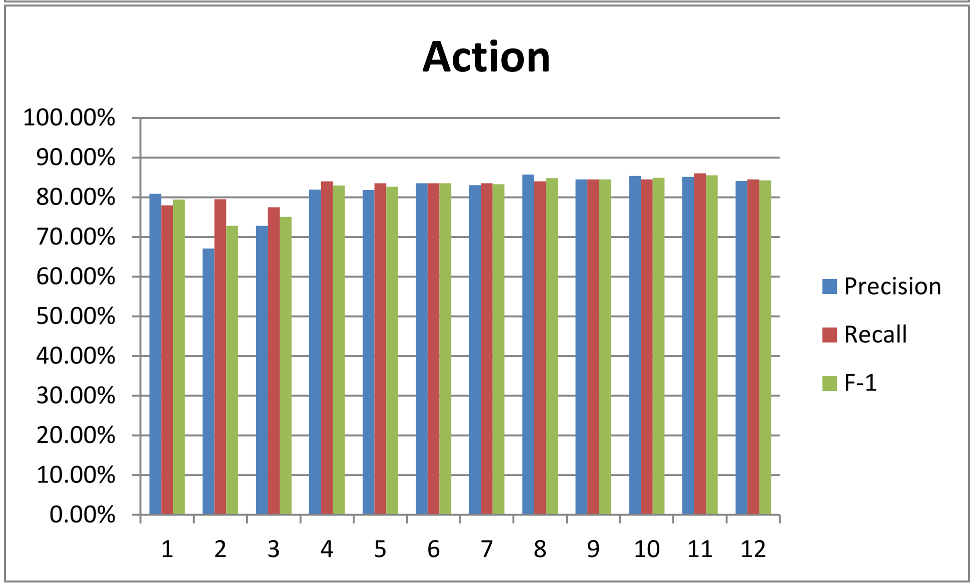
A similar model could be used to address coreference between ingredients and predicates that have no arguments but are applied to those ingredients. Unfortunately, we did not have time to implement this form of coreference resolution.

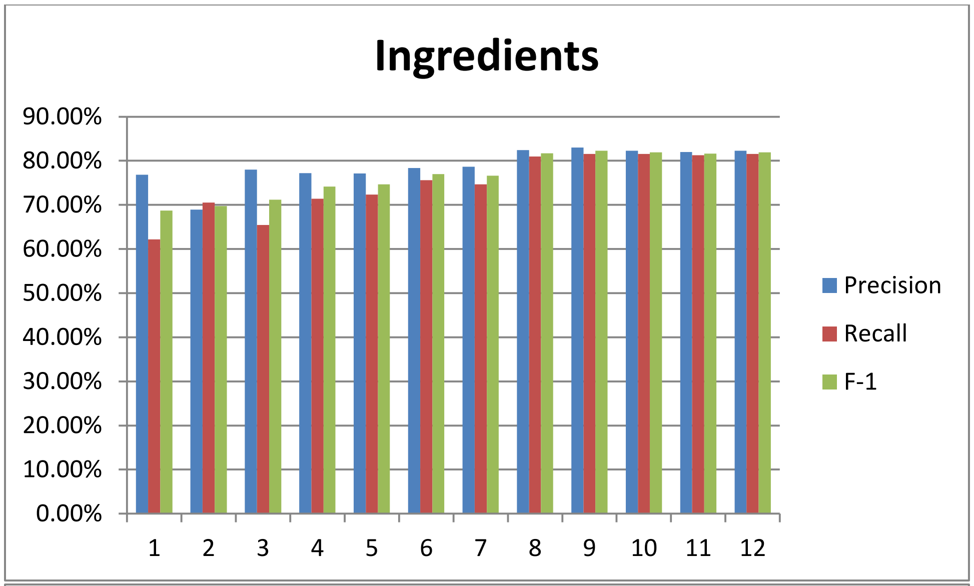
Results and Discussion

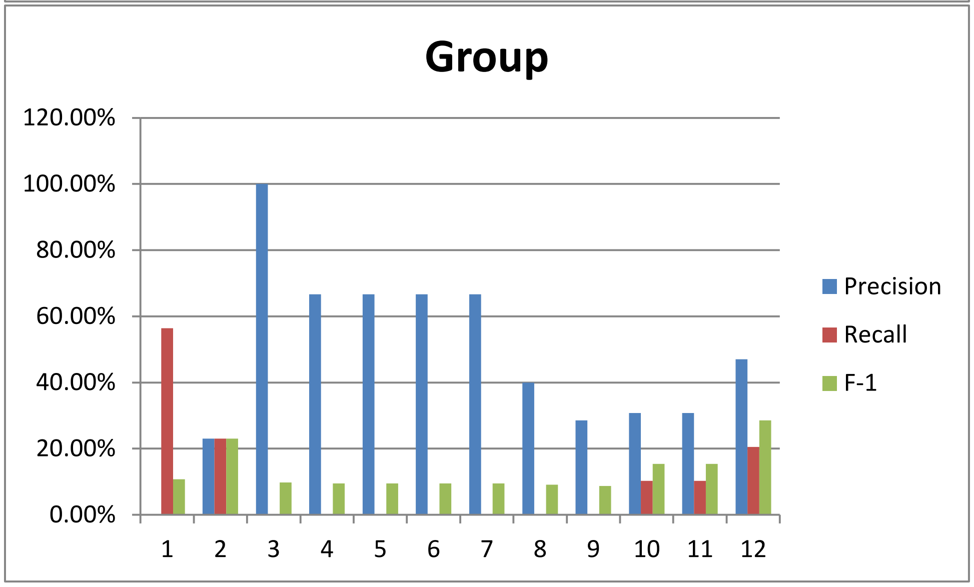
NER

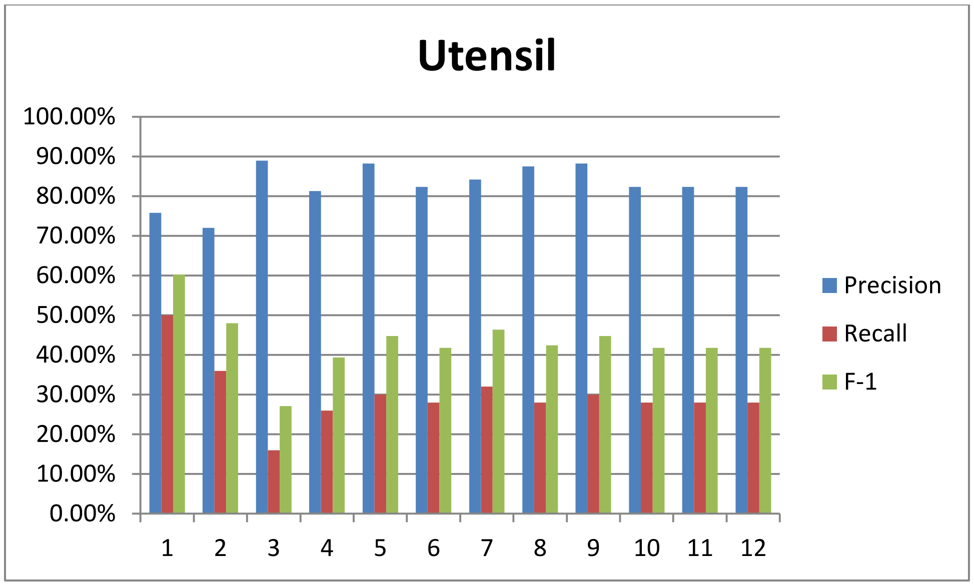
Below we show the contribution of each feature to precision, recall, and F-score as features are added incrementally. In addition to showing statistics for overall classification, we show it for the labels that are particularly important for our task of information extraction: INGREDIENT, ACTION, GROUP, and UTENSIL. These were computed for a fixed training set of 19 recipes (sentences) and 12 testing recipes. Each recipe contained about 5-10 ingredients, and approximately 10 sentences for the directions.


Overall
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
1
2
3
4
5
6
7
8
9
10
11
12
Precision
Recall
F-1


Action
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
1
2
3
4
5
6
7
8
9
10
11
12
Precision
Recall
F-1


Ingredients
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
1
2
3
4
5
6
7
8
9
10
11
12
Precision
Recall
F-1


Group
120.00%
100.00%
80.00%
60.00%
40.00%
20.00%
0.00%
1
2
3
4
5
6
7
8
9
10
11
12
Precision
Recall
F-1



The two classes that were the hardest to identify were the group words and the utensils. The poor performance of group words is particularly concerning because without a correct label, the co-reference resolution module has no chance of resolving these words. This is because, although the food words are identifiable because they are located in the ingredient list, there is nothing in the structure of the recipe that identifies groups and utensils from other nouns in the sentence. We can see the feature that increased the F-score for these classes by the most was addition of feature 8, which indicates if the word is a noun that is not present in the ingredients list. While this is one of the most distinguishing features about these words, there are many other nouns that activate this feature that are neither utensils nor food related group words (*examples)*. Another problem arises from the fact that many imperative nouns are also tagged as nouns by the Stanford parser, which introduces more noise into the evaluation of the feature.

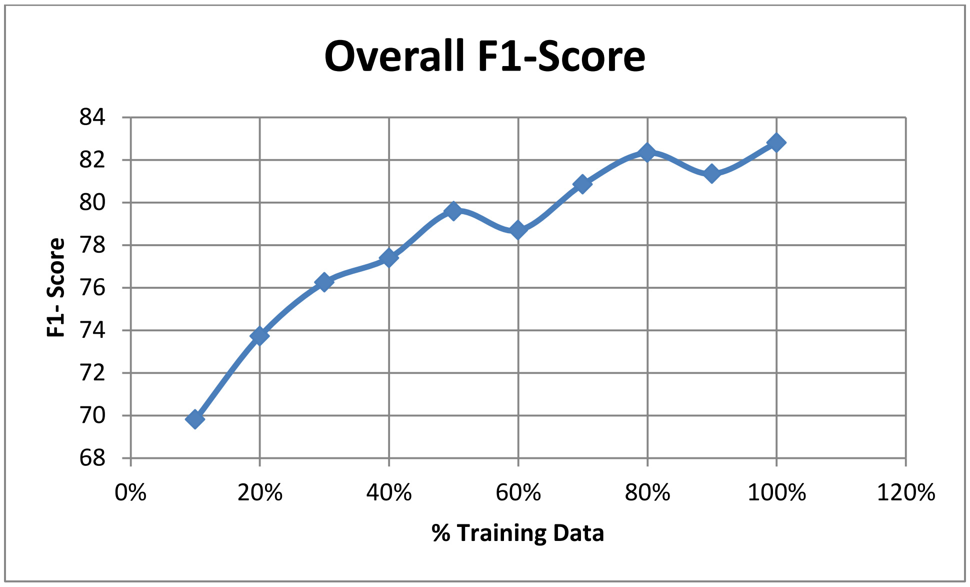
It is interesting to note that adding the feature that looks for the prefix “mix” or the word “ingredient” also helped improve the performance on GROUP nouns. However, in practice, although the classifier seems to identify the word “*mixture*” itself correctly, it frequently doesn’t label the preceding word.

For example:

stir	ACT	
cheese	FOOD	
and	O	
reserved	O	O
stuffing	O	
mixture	GRP	
into	O	
beef	FOOD	
mixture	GRP	
.	O	

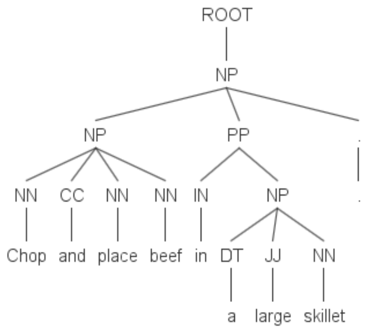
Although the words mixture is correctly classified twice, both the preceding words are incorrectly labeled. They should be labeled as GROUP words.

We varied the number of training recipes to get a learning curve of our classifier. As we can see, even with our full set of training data, we were still in the regime where more data would make significant improvements in labeling performance. However, because of the expensive nature of hand-labeling data, we were unable to obtain more.



Some classes, such as AMOUNTS and UNITS and were identified with very high accuracy (F = 91.69 for AMOUNTS, F = 84.98for UNITS). However, this was usually because AMOUNTs have very distinct morphological features because of the inclusion of numerical characters, and UNITs have a high probability of occurring directly after AMOUNTs.

Many of our classes require good part of speech information to be identified. For example, ACTIONS are always verbs, so the part of speech can be a very discriminative feature for identifying ACTIONs. However, because the majority of our sentences are either fragments, as is the case for those in the ingredients list, or imperative sentences, on which the Stanford parser performs particularly poorly on because of the lack of such training instances. One of the most frequent errors is that imperative verbs are classified as nouns. One such erroneous parse is shown below.



Because of the unreliable sentence parsing, it was difficult to incorporate features that incorporated more global syntactic information. One proposed feature was to incorporate whether or not a word is in a prepositional phrase, under the assumption that UTENSILs often occur after words such as "with" or "in" or "into". However, we found this feature to actually hurt performance because of errors in the sentence parsing. However, this syntactic information was still useful during SRL in order to assign arguments and agents to predicates.

SRL

Using the hand-coded SRL rules on 12 recipes, which had a total of roughly 150 predicates (found by the predicate-finding rule), SRL was done and a human (one of the authors) was asked for each predicate whether the correct arguments had been correctly labeled for that predicate, and the percentage of predicates for which the human said that SRL was correct was calculated. Admittedly, this score had two major flaws. First, it did not give any "partial credit" to answers that were mostly but not completely right, as often occurred in long sentences with many arguments per predicate. Second, it measured precision but not recall, meaning that it might favor NER labeling models that caused fewer predicates to be chosen (and chosen in such a way that SRL was likely to get a large proportion of the ones chosen correct – for example, the precision could be increased if NER assigned the "ACT" label only to words in short sentences, since SRL tended to do better on shorter sentences). For this reason, comparisons using this score were only made between models with the same NER part, and in general, this score was treated only as a rough lower bound on the quality of a model, and qualitative analysis played a much bigger role in decisions about SRL rules. The only reason this score was used was so that we could get an overall idea of how SRL was doing with different rules, which would be difficult if we only used qualitative analysis.

The absolute precision for SRL over the testing set was 0.53, which was much better than chance. Among the errors observed, two types stood out as problems that might be solved by modifying the SRL rules. First, Recipients that had multiple (consecutive) "FOOD" and/or "GRP"-labeled words were sometimes mistakenly split up. For example, in the sentence:

"add remaining broth , corn and worcestershire sauce ."

the predicate "add" was correctly given a recipient "broth" but incorrectly given two recipients, "worcestershire" and "sauce", rather than the single recipient "worcestershire sauce". This happened despite the fact that both "worcestershire" and "sauce" were labeled by NER as "FOOD". "worcestershire" and "sauce" were siblings in the parse tree, but the tree structure was very flat, and the lowest node that spanned both "worcestershire" and "sauce" spanned the phrase "broth , corn and worcestershire sauce", so the parse tree was the cause of this error. Since we had less control over the parser's behavior (compared with NER and SRL), and since we noticed that "FOOD" and "GRP" words appearing consecutively (without commas in between) almost always formed a single argument together, we decided to add a “recipient-chunking” rule that would take the output of SRL and coalesce any consecutive Recipient arguments.

A second type of error we noticed was a grouping of arguments with the wrong predicate. For example, in the sentence:

“cut the mushrooms into thin slices ; or use 1/2 oz dried mushrooms and soak them in 1/2 cup lukewarm water for 15 minutes .”

The predicate “cut” was correctly given recipient “mushrooms” (first occurrence) but incorrectly given recipients “mushrooms” (second occurrence) and “water” (second occurrence). Again, NER labelled “cut” “soak” “mushrooms” (both occurrences), and “water” correctly, so the problem must have been related to how the sentence was parsed. We figured that modifying the definition of “closeness” between two nodes as the sum of the number of steps from each node to their lowest common ancestor might help prevent mismatch errors such as these.

Taking human input over the entire testing set took very long, so for subsequent tests, human input was requested for only about ¼ of the data (chosen at random). When chunking of recipients was added, the absolute precision was 0.40. It is important to note that this score came from only [L] data-points, so variation would be high. Besides, recipient-chunking seemed to be having an overall positive effect; for example the in the Worcestershire sauce example, “worcestershire sauce” was given to “add” as a single Recipient. No instances were observed where recipient chunking led to an error (note that the chunking was done after the rest of the SRL algorithm, so incorrect chunking would be very easy to pinpoint as a source of error). Another run of recipient-chunking SRL with human queries on a random ¼ of the data was done; this time, the absolute precision was 0.57, confirming suspicions that there was high variance due to the low number of data-points used. Thus, recipient-chunking was kept as a rule.

The next rule modification to consider was to redefine “closeness” between two nodes as the sum of the number of steps from each node to their lowest common ancestor . The absolute precision was 0.60 – not much effect (remember, this was calculated from a small number of data-points). Unfortunately, on the troublesome sentence “cut the mushrooms into thin slices ; or us ½ oz dried mushrooms and soak them in ½ cup lukewarm water for 15 minutes”, the predicate “cut” was given no arguments. Fortunately, with the original definition of “closeness”, SRL did not often group arguments with the wrong predicates when more than one predicate was present. Thus, in the end we decided to use SRL with Recipient-chunking and the original definition of “closeness” of nodes.

The biggest source of error for SRL was error in NER labelling; if a node's leaves were not all labelled as “FOOD” or “GRP”, SRL would have no reason to consider that node for labelling as a Recipient; likewise, if a node was accidentally labelled “ACT” by NER (as was the word “olives” in one example), SRL would think “olives” was a predicate.

One interesting behavior of SRL in some cases was that it did the job of coreference resolution for argument-free predicates that most likely operated on mixtures. For example:

In “add chilli powder ; cook for 5 minutes more .”, “chilli powder” is grouped with “cook”

In “add wine , stir , cover , and simmer for 10 minutes .”, “wine” is grouped with “add”, “stir”, “cover”, and “simmer”

This was considered an error when computing the absolute precision score, since it would not be coreference resolution if, for example, “add wine” had been replaced with “boil off alcohol”; however, this behavior could also be thought of as an advantage because it would take care of some of the coreference resolution.

Future Work:

As previously mentioned, our training curve did not plateau for the amount of training data we used, which suggests that more training data would improve our NER's F1 score. Since NER was a major source of error for SRL, which in turn would cause errors in coreference resolution, making NER more accurate would help make practically all other pieces of our project more accurate.

Another way to improve our algorithm would be to train a parser on imperative data (ideally data from recipes or other instructions). In some cases, the Stanford Parser would misclassify the imperative verb as a noun because it was used to seeing nouns at the beginning of sentences; fortunately, the parser we used had been trained on some imperative data, but more imperative data would lead to more accurate parses, and more accurate parses would lead to more accurate SRL and coreference resolution.

Some interesting cases that our model did not account for in SRL were sentences such as “lightly grease a hot cast iron griddle”, “prepare a 9x5" loaf tin by greasing it with butter and lining it with butter waxed paper”, “grease a 9 inch pie plate”. Here, the utensil is the recipient, and sometimes the ingredient is the agent. It would be interesting to see whether our methods could be applied to these steps by simply treating utensils as ingredients and ingredients as agents, and what extra information could be gleaned from these steps to aid coreference in the case where ingredients are recipients and utensils are agents.

Finally, an interesting direction would be to try to detect timing-related data; in particular ordering of actions and events – for example, “Cook until lightly browned, but do not remove until popping has subsided” - would be useful but challenging.

Contributions:

Rahul programmed the NER and co-reference resolution modules, and Kevin programmed the SRL component.