COLUMBIA UNIVERSITY – CUNY

Homework 1 - NLP/ML/Web

Jessica Ouyang and Joe Ellis

October 7, 2013

1 TASK – CHEF RECIPE CLASSIFICATION

For this project we chose to attempt to identify Famous Chef's that appear on the FoodNetwork based on the characteristics of their recipes. FoodNetwork.com has many recipes from some of their most famous chefs available on their website, and with some effort we were able to extract around 200 of these recipes from various chefs automatically using web technologies. We then extracted the useful portions of the webpages, such as the ingredients, title, and instructions. Using this cleaned text we created different word representations based on these documents, and attempted to identify the author/chef for each recipe.

1.1 WEB SCRAPING

We have chosen to try to classify recipes on FoodNetwork.com based on which famous chef devised the paticular recipe. FoodNetwork is a great resource for many recipes and is widely utilized by many people.

However, there is no very easy way to gain access to each of the recipes, as they are not organized by chef pages. However, we can gain access to the recipes by using the website‘s recipe search ability, and then filtering the results by chef. The FoodNetwork also will only show 12 recipes per html page for a given search query, therefore we must iterate through the search results automatically, and extract the links to each recipe and then iterate through the results. We have created a command line program that takes as input the name of the chef to search and filter FoodNetwork.com for, and downloads each of the html files for each recipe into a directory. The url that is queried by the program can be seen in the “GetChefPages.py” fuction within our source code directory. Where the search finds the recipes created by “Guy

Fieri" and returns the 12th to 24th recipe in the list. We then parsed the html file to extract the links for each recipeâĂŹs html page.

After we had downloaded all of the recipes, we then moved on to extracting the recipe from the html page. We have created a program that systematically removes the ingredients and directions from each html file, and outputs them to a plain-text file with the same name as the html file, but a .txt file extension. To perform the parsing we used the python BeautifulSoup library.

1.2 Representation and Classification

Our goal was to classify recipes by chef. We downloaded 200 recipes for each of 5 Food Network chefs: Bobby Flay, Giada de Laurentiis, Guy Fieri, Paula Deen, and Rachel Ray.

We treated each recipe as three parts: the title, which usually lists the most important ingredients; the ingredients list, including ingredients that require preparation; and the instructions. We represented the title with a bag of words. The ingredients list was represented with features mapping ingredient names to the amounts used, as well as a single feature counting the number of ingredients that required preparation (for example, the ingredient "peach puree" came with the instructions "put peaches and ice in a blender, blend, etc"). These features did not change among experiments.

The three representations that we tested for the instructions were bag of words, a back-off bag of words (words appearing in fewer than 5 recipes were considered rare), and trigrams.

2 Results

We explored a mixture of different word representations for our testing, and looked to understand which portions of each document would be the most useful for chef classification. For each author we trained on 100 randomly selected recipes, from which we extracted the title, ingredients, and instructions. We then tested the learned classifier with 50 held-out test recipes from each author.

We ran all experiments with MaxEnt in Mallet. We chose MaxEnt because of a highly-cited paper by David Madigan on author identification, which we thought might be similar to chef identification, used logistic regression. Using all features, all three representations achieved 99.8% accuracy on the test set. We tried feature ablation and got the results displayed in the tables provided.

3 Linguistic Analysis of Word Representations

Document Classification is an interesting and widely studied task in the field of Natural Language Processing. It allows for unique challenges based on the type of classification that

Table 1: Results using all 3 feature representations

Features Used	Accuracy
title-only	95.4%
ingredients only	99.8%

Table 2: Results using 3 feature representations seperately on only instructions

Representation Used	Accuracy
BoW	31.2%
back-off	25.4%
trigram	28.6%

is attempted, and many open questions still exist. We will be discussing in this section different word representations, and how they effect the performance of given classification tasks. Document Classification is different than many computer vision tasks, because where as SIFT features are in many ways the one-feature-fits-all prototype for almost all computer vision recognition and classification tasks, different document classification tasks require very separate feature and word representations. For example, general document topic classification may be helped with the removal of stopwords, because they have very little to do with the topic of a given document. However, for author identification the stopwords can be useful grammatical choices that the author makes, and can therefore be very useful.

PRE-PROCESSING STEPS We will begin by discussing the topic of pre-processing steps, and how they can be useful for a given document classification tasks. One possible pre-processing step is known as word-stemming. This process takes words like “estimates”, “estimated”, and “estimate” stems them down to the same word-representation. This can be a useful process for dimensionality reduction of a given word representation for a document, and also helps to disambiguate similar words. However, there are also some disadvantages to using off the shelf stemming processes. Stemming is a very difficult problem, and do to the high variability in the English language all similar words are not stemmed down into the same representation. For example, the words “summarize” and “summary”, although simply they are same word in noun and verb form they are rarely stemmed down into the same word, this performance problem can be an issue in some instances. Another common pre-processing step in word representations is stopword removal. Many words such as “about”, “the”, “because”, etc. are widely used throughout the English language and give little to no topic based discrimination between documents. Therefore, these words are commonly removed before creating the word representation for many NLP classification tasks. However, for our task in particular which is chef recipe classification these stop-words may in face be highly useful. The choice between using different conjunctions, and other functional words are stylistic choices made by an author for rhetorical reasons known to them. Many authors also develop patterns within their writing style of using similar words and sentence structures to make their material accessible to a reader, and in this case including functional words in the representation

Table 3: Results using 3 feature representations seperately on title and instructions

Representation Used	Accuracy
BoW	96.2%
back-off	95.0%
trigram	95.6%

is very important. Pre-processing of the documents before choosing a word representation scheme can have great effect on the outcome of the results.

BAG OF WORDS REPRESENTATION After pre-processing we then must choose a particular document representation for the task at hand. The simplest and one of the most transmittable word representations that can be used is the "bag-of-words" feature representation. This idea has been used widely in a variety of different fields, and is not limited to simply document classification, but is also widely used in Computer Vision as well. "Bag-of-Words" (BoW) simply searches through a document and denotes whether a word is present within the document that is within the word dictionary. This representation has some very nice features, which for one is that it is very intuitive and easy to understand. This is one reason that is has had such high adoption for representation in other fields as well. The BoW features are made up solely of binary values for each word, because they simply denote whether or not a word is present in a document. This allows them to be highly compact and efficiently stored in memory, if memory consumption is a prevalent issue within the system. They are also geometrically very convenient, because each of the dimensions within the vector have the same range. This allows for us to calculate useful distance measures between different points with little to no need for normalization beforehand. Many Machine Learning tasks can be highly dependent on the normalization scheme performed on the data before classification, and therefore using BoW can alleviate some of the dependencies on extra processes. However, one issue that arises with BoW is that this feature representation effectively throws away the count information for the words within each document. For instance, if the word "rice" appears 10 times in a very short recipe it would be safely presumed, that rice is a very important ingredient for this particular recipe. However, using BoW "rice" would have the same numerical value in our document representation as ingredients that only appear one time. By throwing this count data away the BoW representation may be losing valuable information that can be useful in classification.

TERM VECTOR REPRESENTATION Instead of simply denoting if a word was present in a document, we could then instead count the number of times that each word appears in a document and use this integer valued number as our word representation. This representation is known as the term-vector representation, and allows us to capture the frequency data that is lost within BoW representation. However, it does have some disadvantages when compared to BoW. Probably the most obvious disadvantage is that term vector values are integer values, and therefore each value takes up more space in memory than a dimension in the BoW representation. Another possible problem with the term vector representation is that the range of

the values within the representation can be very large, and therefore two different representations can be very far away in Euclidean space making classification difficult. One solution is to normalize the term vector in some way, possibly divide each dimension by the sum of all the values in the term vector to normalize every value by the total length of the document. Other normalization schemes exist, and it was probably trial and error could be used to find effective normalization techniques for a given task.

TF-IDF REPRESENTATION One very useful normalization procedure is known as TF-IDF, and can be used on BoW or term vector representations. TF-IDF is a measure of how important and discriminant each word within a dictionary is. To perform TF-IDF normalization we count the number of documents within our collection, and then divide by the number of documents that the word that which we are trying to normalize appears in. We then finally take the logarithm of this value to obtain the IDF score. Then to gain the TF-IDF vector we multiply IDF by whatever the TF (word-representation) for that word may be. The TF could be BoW, Term Vector, or some other type of representation. This representation allows for us to prioritize the words that are discriminant across the document sets, and then downgrade those words that appear in every document. It is a very useful strategy, and this technique is also widely used in the field of Computer Vision. I have personally often used TF-IDF in CV tasks to great effect, and I believe that this representation is one of the most useful and intuitive document representations available to us.

4 IDEAL REPRESENTATION AND RESULTS DISCUSSION

We expected the ingredients to be the most informative features, and we were right. Our best feature, ingredients and their counts, alongwith the number of ingredients that need preparation, achieved 99.8% accuracy, which is very close to ideal.

The second best feature, the bag of words from the recipe title,achieved 95.4% accuracy, which is also very high. This makes sense. The title of the recipe generally consists of the main ingredients and the cooking technique used. For example, the title“baked mashed potatoes with parmesan cheese and bread crumb” names the main ingredients (potatoes, parmesan, and bread crumbs) and the cooking technique (baking). The other ingredients (milk, butter, mozzarella, salt, and pepper) are not included. While ingredients like, salt, and pepper, might not be very useful in distinguishing between recipes because they are so common, ingredients like butter and mozzarella are useful. The title of a recipe omits some important ingredients.

In addition, the title of a recipe does not indicate how much of each ingredient is used. Our ingredient features each consisted of an ingredient name and the amount used. This is important information because different chefs favor different ingredients – Paula Deen, for example, uses a lot of butter in her recipes, while Rachel Ray uses smaller amounts, often mixed with olive oil.

However, our ingredient features could have been better. Ingredients are usually listed in the format <number> <unit> <ingredient> (1 tablespoon butter). Our ingredient feature would treat the ingredient name as "tablespoon butter" and the amount as "1". A better representation would be to convert all the ingredient amounts to a common unit so that we could distinguish between 1 tablespoon of butter and 1 pound of butter. However, this would be difficult to implement because converting weight to volume requires world knowledge. For example, a human who has seen the wrapper on a stick of butter knows that 8 tablespoons is equivalent to 1/4 pound, but it would be difficult to code this information into a program.

An additional problem with our ingredient representation is due to the way we process the list of ingredients. We tokenize the list by commas, but some ingredients are followed by instructions for how to prepare them. For example, the ingredient "peach puree" might be followed by the parenthetical "(blend peaches and ice)". For the most part, this is not a problem: we extract the parenthetical instructions and add them to the instructions for preparing the dish itself to produce the instruction features. However, some of the parenthetical instructions contain commas: "blend peaches, ice, and ice cream". We do not detect this, so we treat this as three ingredients: "peach puree", with the accompanying instruction "blend peaches"; "ice"; and "and ice cream". While ice can be considered an igredient in this recipe, "and ice cream" is a nonsensical ingredient name.

Thus an ideal representation for classifying recipes by chef would be an mapping from ingredients to the amounts, where the amounts were all given in terms of a common unit of measurement, and where parenthetical instructions could be completely removed.