Recipe Similarity Using Ingredient Extraction and Latent Semantic Analysis

Benjamin Holtz
Department of Computer Science
Stanford University
Stanford, CA 94305
bholtz@stanford.edu

Xiangnong Wang
Public Policy Program
Stanford University
Stanford, CA 94305
gxwang@stanford.edu

Abstract

In this report, we present an approach to determining similarity between recipes using information extraction and latent semantic analysis techniques. Recipe similarity can be used to gain a functional understanding of what distinguishes different types of cuisine. Leveraging recipe data from the WikiBooks Cookbook[1], we can identify natural recipe classes using high-level user generated tags (e.g. "Indian Cuisine"). This report will describe various approaches to improving upon a baseline bag-of-words model.

1 Notes

This report was developed as part of the final project for Stanford University's CS 224U: Natural Language Understanding taught by Bill MacCartney and Chris Potts. The source code can be downloaded from the project's GitHub page[2].

2 Introduction

Food often characterizes culture, and indeed, some cultures characterize themselves by their food. Anecdotally, we can make conjectures about cuisines that share close ties, but to what extent do recipes actually differ between these classifications? This report describes a system to compare recipe similarities within classes and across classes. We use the WikiBooks Cookbook data because it is freely available and is meta-tagged by users based on recipe characteristics. Some examples of recipe-wide tags are food categories like "bread recipes" and "fried recipes," as well as, cuisines like "Indian recipes" and "Chinese recipes." Thus, the WikiBooks Cookbook dataset includes both documents and the natural classes that include these documents.

Recipes are interesting from a similarity point-of-view, becasue there may be some functional reasons for similarity and dis-similarity. For instance, bread recipes likely will call for flour and yeast becasue, in general, these are needed for any bread recipe. It makes sense that bread recipes should be similar. On the other hand, while Indian recipes and Mexican recipes both use spices, the cuisines have different flavor profiles precisely due to the specific spices that are commonly used. These differences in ingredients highlights the differences in the cuisine.

The functional contribution of different ingredients to the flavor profie of a recipe highlights the signfiicance of information extraction for the recipe similarity task. While bag-of-words alone can provide a useful baseline similarity score between recipes and class, information extraction on ingredients lists can capture more fine-grained information about the ingredient itself. Ingredients lists are generally short semi-structured phrases or fragments that include a base ingredient (e.g. onion), a unit of measurement and its quantity (e.g. 2 cups), and some properties (e.g. diced). However, simply running these lines through a parser doesn't provide a clean and consistent way of extracting these data, so there is plenty of room to improve extraction through hand-built features. Information extraction on ingredients lists can tease out functional informa-

[1] http://en.wikibooks.org/wiki/Cookbook
[2] http://github.com/gxwang/cs224u_recipe

tion about the ingredients and their quantities.

3 Methodology

3.1 Data Pre-Processing

Our data comes from WikiMedia as a single file of 2,403 XML-formated recipes. The file is only formatted in a way that reflects its status as a set of WikiMedia pages and not as a set of recipes. Thus, there is substantial pre-processing involved in re-formatting recipes from XML to recipe objects. Immediately, we can extract any XML flagged information like the title of the recipe and the useful recipe category tags associated with each page.

However, the actual text of a Cookbook recipe is all included in a single XML flag, and thus does not have any XML formatting. Fortunately, there is some structure that is gained from typical Wiki page internal formatting. For example, ingredients lists are often notated with text headings like "==Ingredients==." We can rely on the fact that WikiMedia contributors keep WikiBooks Cookbook pages reasonably consistent and cleanly formatted. Thus, we can reliably strip away the unnecessary data and keep only the data we need. The output of the preprocessing is a set of recipe objects with ingredient lines that can be parsed to extract the ingredients from these lines.

3.2 Ingredient Extraction

There are three main tasks for ingredient extraction: identifying units and quantities, identifying base ingredients, and identifying ingredient properties. For each ingredient line, our goal is to extract each of those pieces of data. Ultimately, we achieve this through a set of hand-built features to extract this information, which works to a varying degree of success.

First, to identify the units of measurement, we put together a relatively exhaustive list of cooking units (e.g. cups, liters, cloves) and all common variations, expressing them as a long regular expression statement. Running an ingredient line through the Stanford English PCFG parser and examining the collapsed dependencies, we note unit and quantity pairs by identifying a unit using the regular expression described above and the "num" or "number" relations. We further attempt to normalize the units by converting fractional quantities into their decimal form and all liquid and weight units to a base measurement. For example, 1/2 cups is converted to 118.2941 mL, and 1 lb is converted to 453.5924 grams.

Next, to determine the base ingredient in an ingredient line, we perform a second run of the Stanford parser. This time, we remove the already extracted unit and quantitiy, as well as the word "of," because they often interfere with identifying the base ingredient. Then, using the "root" relation from the collapsed dependency parse, we can extract base ingredient. If the "root" relation fails to find a noun, we look for the base ingredient when iterating through the other relations.

Finally, to identify the properties of an ingredient, we use the "nn," "appos," "amod," and "nsubj" relations from the second dependency parse. For example, the line "1 cup chopped onions" returns "onions" as the base ingredient and "chopped" as the property. Our basic approach ignores some things like compound ingredients. So, 3 teaspoons baking powder would lead to powder as the base ingredient and "baking" as its property.

3.3 Cosine Similarity Classifiers

We use cosine similarity in two classifiers: a baseline simple classifier and a more clever classifier that uses ingredient extraction. The simple classifier simple relies on a bag-of-words model, while the clever classifier leverages more specific ingredient information.

The simple classifier begins by identifying a recipe's ingredient list and taking a bag-of-words over all the text in the list. This does not make a distinction between units, base ingredient, properties, or extraneous words and comments. Each recipe is thus reduced to a term frequency vector. The classifier then uses cosine similarity within a recipe category to find the average similarity between each of the recipesi n a cateogry. The same is reported across categories, comparing recipes from one category to recipes from another.

3.4 Latent Semantic Analysis Classifier

Finally, we implement Latent Semantic Analysis for recipe similarity. LSA is a dimensionality-reduction technique for analyzing the relationships between documents and the terms in documents that relies on

```
  <page>
    <title>Cookbook:1-2 Chocolate Cake</title>
    <id>39236</id>
    <revision>
      <id>2167235</id>
      <timestamp>2011-09-12T06:16:35Z</timestamp>
      <contributor>
        <username>QuiteUnusual</username>
        <id>134830</id>
      </contributor>
      <minor/>
      <comment>[[WB:REVERT|Reverted]] edits by [[Special:Contributions
          /124.107.241.151|124.107.241.151]] ([[User talk:124.107.241.151|talk]]) to
          last version by Jomegat</comment>
      <text xml:space="preserve" bytes="1690">{{recipe}} | [[Cookbook:Cake|Cake]]
{{recipesummary|Cake recipes|Unknown serving amount|30-40 mins|2|Image=[[
    Image:Chocolate-Cake-2006-Jan-04.jpg|300px]]}}

'''1-2 Chocolate Cake''' is a simple, fairly easy chocolate cake recipe.

== Ingredients ==

*    cup (80ml) [[cookbook:vegetable oil|vegetable oil]]
*2 ounces (55g) unsweetened [[cookbook:chocolate|chocolate]]
*1 cup [[cookbook:sugar|sugar]] (or    cup)
*1 [[cookbook:egg|egg]]
*1   cups (300g) [[cookbook:Whole Wheat|whole wheat flour]]
*   teaspoon [[Cookbook:salt|salt]]
*   teaspoon [[Cookbook:Baking Soda|baking soda]]
*   teaspoon [[Cookbook:Vanilla|vanilla extract]]
*   cup (180ml) [[cookbook:water|water]]
*   cup (180g) semi-sweet chocolate pieces (such as chocolate chips)
*   cup (120g) chopped [[Cookbook:Nuts and seeds|nut]]s
*(Optional: 4 ounces (115g) [[Cookbook:Cream Cheese|cream cheese]] or [[Cookbook:Sour
    Cream|sour cream]])

== Procedure ==

#Put the oil and chocolate in an 8in (20cm) square pan and set it in 350 F (180 C)
    oven just until the chocolate melts. (Add sour cream or cream cheese if desired.)
#Add remaining ingredients, except chocolate chips and nuts, to the pan.
#Beat with a fork until smooth and creamy (about 2 minutes).
#Scrape the sides of the pan, and spread the batter evenly in the pan.
#Sprinkle with chocolate chips and nuts. Optionally, cut the chocolate chips and nuts
    into the batter to some extent.
#[[cookbook:Baking|Bake]] at 350 F (180 C) for 30 to 40 minutes, until a toothpick
    stuck in the center comes out clean.
#Cool in pan; eat hot or cooled.

[[Category:Cake recipes]]
[[Category:Chocolate recipes]]
[[Category:Recipes_with_metric_units]]
[[Category:Halal recipes]]</text>
    </revision>
  </page>
```

Figure 1: XML Representation Example

the assumption that words similar in meaning occur close together in text. A word count matrix is produced from text and Singular Value Decomposition (SVD) is used to combine some of the columns of the matrix, reducing the dimensions while preserving the similarity among the rows. LSA is particularly useful because it allows us to compare similarity between documents and small groups of terms in a low dimensional space.

Applied to our task,we first generate a term-recipe occurance matrix. LSA then finds the low-rank approximation to said occurance matrix. This low-rank approximation is particularly important because we presume the occurance matrix is fairly noisy, and LSA can reduce the noise in our matrix.

4 Results and Discussion

4.1 Ingredient Extraction

We use ingredient extraction as a way to improve upon a bag-of-words similarity model. The task of hand-built features for ingredient extractio is also an interesting problem in itself. Thus, we'd like to achieve goot performance with our extraction system. To test ingredient extraction, we evaluate by testing on 50 randomly drawn records across all ingredient list lines.

Data Type	Precision	Recall	F1	Accuracy
Ingredient	0.73	0.90	0.80	0.67
Quantity	0.88	1	0.94	0.88
Unit	0.97	0.85	0.91	0.88
Properties	0.57	0.71	0.63	0.62

Table 1: Evaluation on 50 Random Ingredient Lines.

As expected, our best results come from extracting quantities and units. Specifically, quantity recall is 1 because our system never guesses that the quantity of an ingredient should be missing. So, any errors should come as incorrectly guessed or unexpected quantities. However, quantity precision is also quite high. In general, the task of determining the quantity is relatively easy because of the "num" and "number" relations.

Extraction on units is also quite good. The system has high precision because units are identified using a regular expression. Thus, most errors come from units that were missed by the regular expression and not mis-identified units. The latter example only occured once in the 50 testing examples. Our strategy to capture units seems reasonable, as recall on units is still decent. However, there is clearly room for improvement over the hand-built regular expression we've used for this task.

Base ingredient and ingredient property extraction is significantly more poor. Ingredient recall is high because to extract a base ingredeint, our system only requires that a line's root must be a noun or one of the ingredient property relations must include a noun. Thus, our strategy to capture base ingredients is fairly broad. This also explains why base ingredient precision is quite a bit lower. As a result, ingredient F1 is decent, though accuracy is quite low.

Ingredient property extraction performs the worst of all extraction tasks. Like with base ingredients, recall is decent due to a broad strategy to capture relations. Any word that modifies or is joined with a base ingredient is identified as an ingredient property. However, the strategy is also somewhat naive and can easily miss properties if the ingredient lines are complex or if the base ingredient is mis-identified. Finally, precision on properties suffers the most. The errors here almost always result from a mis-identified base ingredient or a compound ingredient. If the ingredient is mis-identified, the system will often incorrectly identify properties related to the mis-identified ingredient. These errors generally result from complex ingredient list lines. We expect a compound ingredient to result in errors. A term like "hard boiled egg" will parse into "hard" and "boiled" as properties of egg. Clearly, the correct designation would be "hard boiled egg" as the base ingredient and not just "egg." The low precision results in reasonably poor F1 and accuracy.

Extraction of ingredient information can be improved in several key ways. First, the hand-built features, while reasonably powerful for the WikiBooks Cookbook data, may nto be as robust to outside recipe data. Future work could use automated extraction methods to perform this task. Specifically, with labelled data, Conditional Random Fields is a tried and tested information extraction method that we'd like to try for this task. Additionally, because there are certain patterns in the ingredients list text, any bootstrapped information extraction system

could provide good results. These automated methods could be used in conjunction with hand-built features for quantity and unit, which both performed well. Another improvement would be looking at Semantic Role Labelling (SRL). In this project, we used the dependency parse to find ingredients, properties, units, and quantities. However, the dependency parse is not overly reliable for the fragmented form of ingredients list text. Instead, SRL could be useful in determining relation patterns and words that could have been useful for some ofthe trickier eamples in extracting ingredients and properties. Finally, gathering data on compound ingredients could greatly improve base ingredient extraction. For example, we could generate a list of bi-grams and trigrams that are strongly associated around base ingredients. For example, if we wanted to capture "baking soda" but not "chopped onions" as a compound ingredient, we can note that "soda" may really only appear as "baking soda," while "onions" will appear as "chopped onions," "sliced onions," "peeled onions," and others.

4.2 Recipe Similarity

We expect that similarity should be high within each of the recipe categories, especially on very specific recipes (e.g. "bread recipes"). Less defined recipe categories like "camping recipes" will likely have a lower average similarity. We also expect that cross-category similarities should be lower than intra-category similarities. We'd suspect that there are some categories of recipes that are closer in nature than others. For example, "bread recipes" may be closer to "flatbread recipes" than "beverage recipes."

4.3 Electronically-available resources

ACL2012 provides this description in LaTeX2e (acl2012.tex) and PDF format (acl2012.pdf), along with the LATEX2e style file used to format it (acl2012.sty) and an ACL bibliography style (acl.bst). These files are all available at `http://www.acl2012.org`. A Microsoft Word template file (acl2012.dot) is also available at the same URL. We strongly recommend the use of these style files, which have been appropriately tailored for the ACL2012 proceedings. If you have an option, we recommend that you use the LaTeX2e version. **If you will be using the Microsoft Word template, we suggest that you anonymize your source file so that the pdf produced does not retain your identity.** This can be done by removing any personal information from your source document properties.

4.4 Format of Electronic Manuscript

For the production of the electronic manuscript you must use Adobe's Portable Document Format (PDF). This format can be generated from postscript files: on Linux/Unix systems, you can use `ps2pdf` for this purpose; under Microsoft Windows, you can use Adobe's Distiller, or if you have `cygwin` installed, you can use `dvipdf` or `ps2pdf`. Note that some word processing programs generate PDF which may not include all the necessary fonts (esp. tree diagrams, symbols). When you print or create the PDF file, there is usually an option in your printer setup to include none, all or just non-standard fonts. Please make sure that you select the option of including ALL the fonts. *Before sending it, test your PDF by printing it from a computer different from the one where it was created.* Moreover, some word processor may generate very large postscript/PDF files, where each page is rendered as an image. Such images may reproduce poorly. In this case, try alternative ways to obtain the postscript and/or PDF. One way on some systems is to install a driver for a postscript printer, send your document to the printer specifying "Output to a file", then convert the file to PDF.

Additionally, it is of utmost importance to specify the **US-Letter format** (8.5in $\times$ 11in) when formatting the paper. When working with `dvips`, for instance, one should specify `-t letter`.

Print-outs of the PDF file on US-Letter paper should be identical to the hardcopy version. If you cannot meet the above requirements about the production of your electronic submission, please contact the publication chair above as soon as possible.

4.5 Layout

Format manuscripts two columns to a page, in the manner these instructions are formatted. The exact dimensions for a page on US-letter paper are:

- Left and right margins: 1in
- Top margin:1in

- Bottom margin: 1in
- Column width: 3.15in
- Column height: 9in
- Gap between columns: 0.2in

Papers should not be submitted on any other paper size. If you cannot meet the above requirements about the production of your electronic submission, please contact the publication chair above as soon as possible.

4.6 Fonts

For reasons of uniformity, Adobe's **Times Roman** font should be used. In LaTeX2e this is accomplished by putting

```
\usepackage{times}
\usepackage{latexsym}
```

in the preamble. If Times Roman is unavailable, use **Computer Modern Roman** (LaTeX2e's default). Note that the latter is about 10% less dense than Adobe's Times Roman font.

Type of Text	Font Size	Style
paper title	15 pt	bold
author names	12 pt	bold
author affiliation	12 pt	
the word "Abstract"	12 pt	bold
section titles	12 pt	bold
document text	11 pt	
captions	11 pt	
abstract text	10 pt	
bibliography	10 pt	
footnotes	9 pt	

Table 2: Font guide.

4.7 The First Page

Center the title, author's name(s) and affiliation(s) across both columns. Do not use footnotes for affiliations. Do not include the paper ID number assigned during the submission process. Use the two-column format only when you begin the abstract.

Title: Place the title centered at the top of the first page, in a 15 point bold font. (For a complete guide to font sizes and styles, see Table 2.) Long title should be typed on two lines without a blank line intervening. Approximately, put the title at 1in from the top of the page, followed by a blank line, then the author's names(s), and the affiliation on the following line. Do not use only initials for given names (middle initials are allowed). Do not format surnames in all capitals (e.g., "Zhou," not "ZHOU"). The affiliation should contain the author's complete address, and if possible an electronic mail address. Leave about 0.75in between the affiliation and the body of the first page. The title, author names and addresses should be completely identical to those entered to the electronic paper submission website in order to maintain the consistency of author information among all publications of the conference.

Abstract: Type the abstract at the beginning of the first column. The width of the abstract text should be smaller than the width of the columns for the text in the body of the paper by about 0.25in on each side. Center the word **Abstract** in a 12 point bold font above the body of the abstract. The abstract should be a concise summary of the general thesis and conclusions of the paper. It should be no longer than 200 words. The abstract text should be in 10 point font.

Text: Begin typing the main body of the text immediately after the abstract, observing the two-column format as shown in the present document. Do not include page numbers.

Indent when starting a new paragraph. For reasons of uniformity, use Adobe's **Times Roman** fonts, with 11 points for text and subsection headings, 12 points for section headings and 15 points for the title. If Times Roman is unavailable, use **Computer Modern Roman** (LaTeX2e's default; see section 4.4 above). Note that the latter is about 10% less dense than Adobe's Times Roman font.

4.8 Sections

Headings: Type and label section and subsection headings in the style shown on the present document. Use numbered sections (Arabic numerals) in order to facilitate cross references. Number subsections with the section number and the subsection number separated by a dot, in Arabic numerals. Do not number subsubsections.

Citations: Citations within the text appear in parentheses as (Gusfield, 1997) or, if the author's name appears in the text itself, as Gusfield (1997). Append lowercase letters to the year in cases of ambiguities. Treat double authors as in (Aho and Ullman, 1972), but write as in (Chandra et al., 1981) when more than two authors are involved. Collapse multiple citations as in (Gusfield, 1997; Aho and Ullman, 1972). Also refrain from using full citations as sentence constituents. We suggest that instead of

"(Gusfield, 1997) showed that ..."

you use

"Gusfield (1997) showed that ..."

If you are using the provided LaTeX and BibTeX style files, you can use the command `\newcite` to get "author (year)" citations.

As reviewing will be double-blind, the submitted version of the papers should not include the authors' names and affiliations. Furthermore, self-references that reveal the author's identity, e.g.,

"We previously showed (Gusfield, 1997) ..."

should be avoided. Instead, use citations such as

"Gusfield (1997) previously showed ... "

Please do not use anonymous citations and do not include acknowledgements when submitting your papers. Papers that do not conform to these requirements may be rejected without review.

References: Gather the full set of references together under the heading **References**; place the section before any Appendices, unless they contain references. Arrange the references alphabetically by first author, rather than by order of occurrence in the text. Provide as complete a citation as possible, using a consistent format, such as the one for *Computational Linguistics* or the one in the *Publication Manual of the American Psychological Association* (American Psychological Association, 1983). Use of full names for authors rather than initials is preferred. A list of abbreviations for common computer science journals can be found in the ACM *Computing Reviews* (Association for Computing Machinery, 1983).

The LaTeX and BibTeX style files provided roughly fit the American Psychological Association format, allowing regular citations, short citations and multiple citations as described above.

Appendices: Appendices, if any, directly follow the text and the references (but see above). Letter them in sequence and provide an informative title: **Appendix A. Title of Appendix**.

Acknowledgment sections should go as a last section immediately before the references. Do not number the acknowledgement section.

4.9 Footnotes

Footnotes: Put footnotes at the bottom of the page. They may be numbered or referred to by asterisks or other symbols.[3] Footnotes should be separated from the text by a line.[4] Footnotes should be in 9 point font.

4.10 Graphics

Illustrations: Place figures, tables, and photographs in the paper near where they are first discussed, rather than at the end, if possible. Wide illustrations may run across both columns and should be placed at the top of a page. Color illustrations are discouraged, unless you have verified that they will be understandable when printed in black ink.

Captions: Provide a caption for every illustration; number each one sequentially in the form: "Figure 1. Caption of the Figure." "Table 1. Caption of the Table." Type the captions of the figures and tables below the body, using 10 point text.

5 Translation of non-English Terms

It is also advised to supplement non-English characters and terms with appropriate transliterations and/or translations since not all readers understand all such characters and terms.

Inline transliteration or translation can be represented in the order of: original-form transliteration "translation".

6 Length of Submission

Long papers may consist of up to eight (8) pages of content and an additional two (2) pages of ref-

[3] This is how a footnote should appear.

[4] Note the line separating the footnotes from the text.

erences, and short papers may consist of up to four (4) pages of content and two (2) additional pages of references. Papers that do not conform to the specified length and formatting requirements are subject to re-submission.

7 Other Issues

Those papers that had software and/or dataset submitted for the review process, should also submit it with the camera-ready paper. Besides, the software and/or dataset should not be anonymous.

Please note that the publications of ACL2012 will be publicly available at ACL Anthology (http://aclweb.org/anthology-new/) on July 2nd, 2012, one week before the start of the conference. Since some of the authors may have plans to file patents related to their papers in the conference, we are sending this reminder that July 2nd, 2012 may be considered to be the official publication date, instead of the opening day of the conference.

Acknowledgments

Do not number the acknowledgment section. Do not include this section when submitting your paper for review.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Association for Computing Machinery. 1983. *Computing Reviews*, 24(11):503–512.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.