Distributional Gastronomics

Jonathan Oberländer and Laura Bostan

Abstract. In this work, we present a distributional semantic model for recipes by building a representation of food ingredients and recipes in terms of co-occurrence vectors, recording their distributional pattern in a recipe corpus. We build a space for ingredients and two spaces for recipes. We show that by composing ingredients in a recursive manner we get a more accurate approximation of recipes than by a more basic method. This idea can be exploited in order to generate new recipes using existing ingredients by looking at similar vectors. Alternatively, it can be used to build an app which might be used to get inspiring ideas for what to cook while having a list of ingredients.

1 Introduction

When explaining the concept of an algorithm to someone unfamiliar with it, an often used example for real-world applications of "algorithms" is *recipes*. Recipes are (somewhat) unambiguous instructions on when to add what ingredient in what way and how to process it to result in a certain meal. Algorithms are written in some kind of language whose interpretation gives the interpreter (whether human or computer), some clue on how to perform it.
Using a online database of recipes, we therefore collect co-occurrence counts for ingredients, and then use them to compose them back into recipes. We find our ingredient space gives interesting results when queried for nearest neighbors of ingredients. The composed recipes, unsurprisingly, are close to original recipes with similar ingredients.
Like Jurafsky (2014), we see food as a language, more specifically, recipes. In the tradition of distributional semantics (Turney and Pantel, 2010), we assume that the "meaning" of a word or ingredient is given by the context in which it occurs. The distributional (gastronomical) hypothesis states that similar ingredients should appear in similar contexts.

2 Related Work

Zhang et al. (2008) build recipe retrieval and creation systems leveraging the relations between food-related words in WordNet. Users can enter a list of ingredients they have, and get presented with suggested recipes that best match the input query. To do that, they also develop a similarity measure between two

lists of ingredients. Furthermore, their system can accurately predict a type of dish and a type of cuisine.
Teng et al. (2012) build and examine ingredient-ingredient networks of complements and substitutes of ingredients. Networks were built using PMI-weighted co-occurences of ingredients in recipes, and suggestions for ingredient modifications in recipes, respectively. Ratings of recipes were predicted well, combining features from both models.
Regneri et al. (2013) build a corpus combining high-quality video recordings of cooking tasks with aligned textual descriptions, who in turn are aligned to predefined activity descriptions. While it also includes (a few) cooking ingredients, its focus is not on lexical and distributional knowledge of them, but on the cooking process itself, and the actions and tools involved, plus linking it with visual information. Similar work in semantic parsing of instructional cooking videos was done by Malmaud et al. (2014).
In a similar direction goes Tasse and Smith (2008), who develop a formal instruction language called *MILK* that recipes can be written in, and develop a small corpus of 350 recipes translated (by annotators) into *MILK*. In contrast, we build on a large and inconsistent web corpus and therefore don't model the cooking process.
Ahn et al. (2011) test Rozin's flavor principle (Rozin, 1973) at the molecular level across the world, and find that Western European and North American recipes tend to pair ingredients that share the same flavor, whereas the East Asian recipes combine ingredients that do not have overlapping compounds. This difference might characterize a cuisine. Recently, the "food pairing hypothesis" was tested for Indian cuisine too (Jain et al., 2015). It was found that the flavor sharing was significantly less than expected by chance and that the spices are ingredients that contribute the most to the negative food pairing.

3 Methods

We view recipes as sentences of a language, where a word is an ingredient. We further simplify our model by saying that the order of words or ingredients doesn't matter, which is motivated by the fact that ingredients in our corpus aren't always sorted by order of application.
Open Recipes [1] is a database of recipes, automatically collected by crawler scripts, in a unified JSON format. As our input corpus, we used the latest Open Recipes dump (accessed on 2014–03-20). For practical reasons, we ignore a large part of the information present in the corpus: Neither instructional information on the cooking process is used, nor the amounts of ingredients.

[1] https://github.com/fictivekin/openrecipes

```
12 whole Dinner Rolls Or Small Sandwich Buns (I Used Whole Wheat)
1 pound Thinly Shaved Roast Beef Or Ham (or Both!)
10 ounces carrots, shredded on a box grater or sliced whisper thin on a mandolin
optional: whole pomegranate seeds or fresh/dried rose petals, a bit of bee pollen
```

Fig. 1. Examples of uncleaned "ingredients" in the corpus

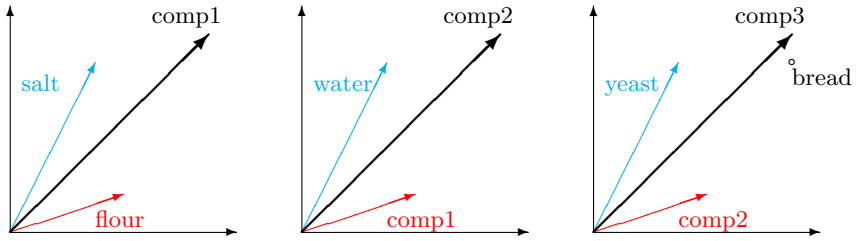

Fig. 2. Recursively composing $\overrightarrow{bread} \approx \overrightarrow{flour} + \overrightarrow{salt} + \overrightarrow{water} + \overrightarrow{yeast}$

Co-occurrences between ingredients were then collected. As a first step, the ingredient names were cleaned. Because of noisy data entered by users (some examples in Figure 1[2]) and amounts and units being part of the ingredients in the corpus, there was a total of 412,858 different strings for ingredient names. Using a number of heuristics, that included cutting away anything that looked like a unit, a number, parts in parantheses, after commata, a naive stemming procedure, and more, we reduced that number:

No existing stemmer was used, due to the specific nature of our data: The cleaning wasn't in the scope of a stemmer, since we actually want to get rid of a lot of words, not just normalize them. Instead, the cleaning/stemming was done using a number of regular expressions. First, substrings in parantheses were removed, then everything after a potential first comma, all digits (including symbols for fractions commonly used). A few other replacements and prunings were made, like splitting the text at "for" and only using the first part (which turns e.g. "tomatoes from spain" to "tomatoes"). Most importantly, using a hand-compiled list, units and their abbreviations were removed, since we don't want to handle "1 tsp sugar" and "2 g sugar" differently.

In the end, the number of different ingredients was 6514 (see Table 1).

[2] There are many different strings that could be reduced to the same ingredient. Sources for noise are comments by users (see the first two lines of Figure 1), several distinct ingredients listed as one (see the last line), or descriptions of how to handle the ingredient; in general everything that leads to inconsistent entries for the same ingredient.

Number of recipes	172893
Number of ingredients (tokens in corpus)	1689892
Number of ingredients (types in corpus)	412858
Number of ingredients (types after unifying)	6514

Table 1. Information about the corpus

The resulting list was then weighted by occurrences. We considered only ingredients used more than ten times to filter out further noise, which brought us down to a reasonable number of 6,326 relevant ingredients. We considered purging even more, after all, this list still contains almost 150 distinct elements containing the string "onion", but we assumed we would lose information by removing adjectives, at least for some ingredients. A "large onion", for instance, could be used in very different contexts than a "small onion".

$$\mathrm{PPMI}(a, b) = \max\left(0, \log \frac{P(a, b)}{P(a)P(b)}\right) \tag{1}$$

A large co-occurrence matrix of ingredients was then automatically created. For handling the vector space operations, we utilized the DISSECT toolkit (Dinu et al., 2013). The matrix was imported as a vector space, weighted with the Positive Pointwise Mutual Information (PPMI, see (1)) measure and reduced to 20 dimensions (after experimenting on 200, and 50 dimensions too), using Singular Value Decomposition. We chose such a low number in the hope that similar features would merge together and our results would eventually not just be reproducing the original recipes when ingredients were composed, but also finding similar recipes that *don't* necessarily contain the ingredients of the original composition.

We then built recipe spaces using two different methods: In our **BasicRecipes** space we obtained recipe vectors by just setting the co-occurrence of each recipe with all of it's ingredients to 1.

The **ComposedRecipes** space was created by summing the vectors of all of its ingredients using a recursive (see Figure 2) Weighted Additive function (Mitchell and Lapata, 2010) with α and β set to 1: For a given list of n ingredients, if n is 1, return the first element of the list as the composed vector. Otherwise, take the last two vectors from the list, compose them using the Weighted Additive function, and put them back on the list. This way, even recipes with a large amount of ingredients could be reduced to the same simple vector as ones with fewer ingredients.

4 Results

Input ingredient	Nearest neighbors
flour	warm water, egg yolk, nutmeg
milk	melted butter, butter or margarine, eggs
mozzarella	basil, pasta, freshly grated parmesan
blueberries	frozen mixed berries, peaches, strawberries

Table 2. Nearest neighbors (cosine distance) of some ingredients

We manually and subjectively evaluated our ingredient space by giving it an input ingredient and judging whether we consider the most similar vectors to be related gastronomically, where as a similarity measure we used the Cosine Distance, defined as $cos(u, v) = \frac{<u,v>}{\sqrt{||u||\,||v||}}$. For the most part, this works very well. When presented with an ingredient, the nearest neighbors seemed to "fit well" to it (Table 2).

Next, we manually looked at the nearest neighbors of composed recipes with recipes in the two recipe spaces. Our investigations seem to confirm our intuition: The **ComposedRecipes** space outperforms the primitive **BasicRecipes** space. When given, for example, "water", "flour", "yeast", and "salt", composing them leads to nearest neighbors such as "home made mini bread", "Italian bread", and "french baguettes" in **ComposedRecipes**, but "pina colada baklava", "huckleberry cheesecake", and "honey-mint yogurt smoothie" in **BasicRecipes**.

Quantitative evaluation of our work proved challenging. To automatically evaluate the quality of our vectors, we took all of our ingredients that are found in WordNet (Miller, 1995), calculated the Cosine Distance between their vectors, and compared it with the JCN distance (Jiang and Conrath, 1997, see (3)[3]) in WordNet, using the Brown corpus (Francis and Kucera, 1979) as a seed corpus. As a comparism, we also compared it with the path distance in WordNet (see (2)).

$$Sim_{\text{path}}(ing1, ing2) = \frac{1}{\text{minpath}(ing1, ing2)} \tag{2}$$

$$Sim_{\text{JCN}}(ing1, ing2) = \frac{1}{IC(ing1) + IC(ing2) - 2 * IC(LCS(ing1, ing2))} \tag{3}$$

[3] IC is the information content of a concept based on some corpus, LCS is the least common subsumer, i.e. the deepest node that subsumes both ingredients.

Cutoff at n	Path distance	JCN distance
10	0.63	0.53
20	0.30	0.11
50	0.06	0.13
100	0.04	0.11

Table 3. Spearman's ρ correlation between Cosine similarity and Path distance or JCN, respectively, at various cutoffs.

The Spearman correlation values for various cutoffs (using the most common n ingredients) can be seen in Table 3. None of them were significant with $p < 0.05$, but that is not too surprising: In WordNet, words are arranged according to their actual hierarchy, while our space attempts to represent how they are used in cooking, which could be (and is assumed to be) very different. For example, if *butter* and *olive oil* were used interchangeably in recipes, they would be very close in our space, but not in WordNet, because *olive oil* is infact something very different from *butter*.

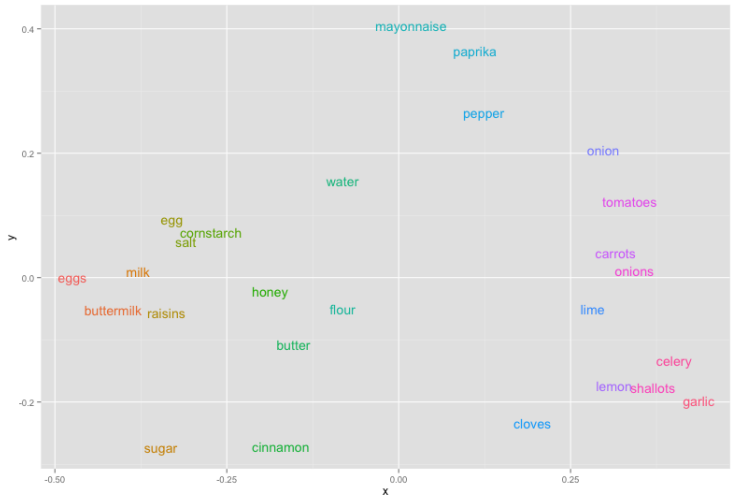

Fig. 3. 2D projection of ingredients

A 2D projection of some top ingredients from our space (plotted using the R library LSAfun (Fritz Guenther, 2014) with the Multidimensional Scaling (MDS) method), can be seen in Figure 3. It can be seen that some similar

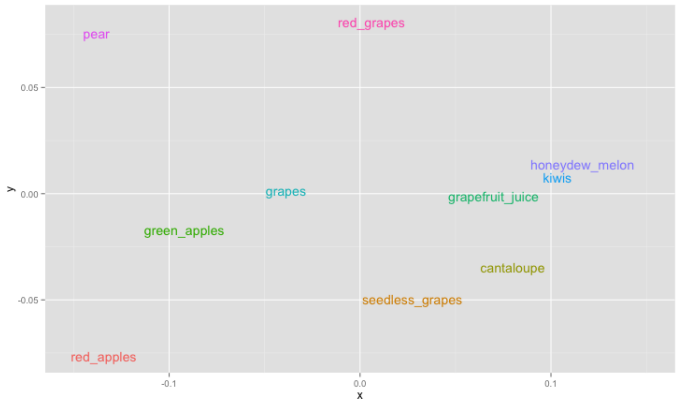

Fig. 4. 2D projection of the 10 nearest neighbors of "grapes"

ingredients cluster together: The spices "paprika" and "pepper" together on the top, vegetables on the middle right, and mostly sweet or neutral items on the left side of the plot ("sugar", "honey", "raisins").
In Figure 4, the nearest neighbors of "grapes" were plotted using the same technique. It can be seen that all grape-related vectors are fairly close to each other, followed by other fruits and a fruit juice.

5 Conclusion & Future Work

The presented methods are just a first step in what we expect to be a long future of distributional gastronomics. We took only the most obvious steps and expect others (and ourselves) to come up with more advanced techniques in the near future.
We composed solely using untrained Weighted Additive functions, and Cosine distance as a similarity measure. Future work could explore other composition methods and similarity measures. It would also be interesting to see whether giving up on some of our simplifications can produce more interesting results:
If the ingredient list is expected to be sorted in some way, order of ingredients could play a role in the weighting of composing vectors. A different improvement could lie in harnessing the supplied amounts (and units) of ingredients to see whether an ingredient plays a major role in a recipe or is just a small addition.
An obvious improvement to our data would have been a cleaner corpus with standardized, fixed ingredient names, clearly seperated by amounts and units.
We completely ignore the cooking instructions, by which we obviously lose a lot of information, exemplified by the fact that there are many ways to combine a set

of ingredients. Integrating it — maybe in a manner similar to Teng et al. (2012) — into the vector model should yield an improvement. A different source corpus as a start could partially solve many of the problems mentioned, but would likely come with a loss in size.

Better methods of automatic evaluation would be a welcome addition to our work, alternatively, evaluating it against a large-scale human gold standard: One could use a platform like Amazon Mechanical Turk[4] to have annotators assess similarity judgements made by our system.

Very interesting to see would be applications built on distributional gastronomics. As proposed in our abstract, such an application could consist of making recipe proposals to a user who enters a list of ingredients they have. A very simple proof-of-concept of such an application was written as a small script working on our ingredient and **ComposedRecipes** space.

As shown in Ahn et al. (2011) (and recently validated for Indian cuisine in Jain et al. (2015)), recipes from different parts of the world behave very different: While the flavor in western cuisine is mostly similar, East-Asian cuisine pairs contrasting flavors in typical recipes. Given that insight, vector spaces for different cuisines will look very different and localized spaces might gain informativeness over general ones.

[4] http://mturk.com

Bibliography

Ahn, Y.-Y., Ahnert, S. E., Bagrow, J. P., and Barabási, A.-L. (2011). Flavor network and the principles of food pairing. *Scientific reports*, 1.

Dinu, G., Pham, N. T., and Baroni, M. (2013). Dissect- distributional semantics composition toolkit.

Francis, W. N. and Kucera, H. (1979). Brown corpus manual. *Brown University*.

Fritz Guenther (2014). *LSAfun: Applied Latent Semantic Analysis (LSA) functions*. R package version 0.1.

Jain, A., K, R. N., and Bagler, G. (2015). Spices form the basis of food pairing in Indian cuisine. *ArXiv e-prints*.

Jiang, J. J. and Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*.

Jurafsky, D. (2014). *The Language of Food: A Linguist Reads the Menu*. WW Norton & Company.

Malmaud, J., Wagner, E. J., Chang, N., and Murphy, K. (2014). Cooking with semantics. *ACL 2014*, page 33.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439.

Regneri, M., Rohrbach, M., Wetzel, D., Thater, S., Schiele, B., and Pinkal, M. (2013). Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*, 1:25–36.

Rozin, E. (1973). *The flavor-principle cookbook*. Hawthorn Books.

Tasse, D. and Smith, N. A. (2008). Sour cream: Toward semantic processing of recipes. Technical report, Technical Report CMU-LTI-08-005, Carnegie Mellon University, Pittsburgh, PA.

Teng, C.-Y., Lin, Y.-R., and Adamic, L. A. (2012). Recipe recommendation using ingredient networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 298–307. ACM.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *CoRR*, abs/1003.1141.

Zhang, Q., Hu, R., Mac Namee, B., and Delany, S. J. (2008). Back to the future: Knowledge light case base cookery. In *Conference papers*, page 15.