CS 263 Final Project
Cooking Up Recipes: A Recipe Recommender Based on Text Similarity

Ashley Chiu
UID: 003940452
ashleychiu@ucla.edu

Ritvik Kharkar
UID: 904314219
ritvikmathematics@gmail.com

Adam Rohde
UID: 404945871
adamrohde@ucla.edu

Abstract

In this project we use a range natural language processing techniques (bag of words, Doc2vec, recurrent neural networks-based LSTM models, and pre-trained language model BERT) to learn vector space embeddings for cooking recipes. We use these embeddings to determine similarity between recipes and recommend such similar recipes, based on user input to a web demo app. We assess the quality of recommendations and embeddings by human review, PCA, and cluster analysis, finding that our models produced mixed results. Generally, simpler models seem to perform best.

1 Introduction and Motivation

The shelter-in-place orders implemented across the world in response to the COVID-19 pandemic have fueled a renewed interest and increased need to cook at home. This behavioral change motivates our goal of creating a tool to make cooking more accessible and/or help expand one's culinary horizons in the form of a recommender system for recipes.

Most simply, cooking recipes are structured documents consisting of three main sections: a title, a list of ingredients, and a sequence of instructions, where instructions contain unstructured text.

Based on reviewed related works, we employ several different NLP techniques on this data, ranging from a simple bag of words model to a state-of-the-art BERT model to learn features and word embeddings from cooking recipes. From these learned embeddings, we build a recommender system based on the cosine similarity between recipes.

We judge the quality of recommendations and vector space embeddings with human review, PCA, and cluster analysis. Given computational constraints and the nature of the data, this review shows mixed results across the models. Generally, we see that simpler models tend to produce the most consistent recommendations.

The remainder of this report contains a brief description of each of the chosen models (Bag of Words, Doc2Vec, LSTM, BERT; see Section 3.2) applied to the the Recipe1M+ dataset from MIT CSAIL (see Section 3.1), as well as how we define and evaluate recipe similarity (see Section 4).

Prior to our description of methods, we briefly review several recent related works that inspire and support our recommender system in Section 2. Following our description of methods, we compare the results of the various models we attempted in Section 4 and share details of our demo in Section 5. Lastly, we offer concluding remarks and discuss potential extensions of our work in Section 6.

About the Collaborators Our team is comprised of 3 first-year MS students from the Statistics Department. Our approach herein reflects our interests in *model selection* and understanding *relationships* in data (i.e. between recipes). We would like to note that this is a first course in CS for the team and that we have no prior experience with AI.

2 Related Work

NLP techniques have been used to analyze cooking recipes by the academic research community, newspapers[1][2], and even popular commercial applications such as Yummly[3]. However, related works show that the techniques utilized to analyze such text span an incredibly broad range due to the unique instructional nature of cooking recipes.

Most simply, recipes have been analyzed using bag-of-words models, which can fairly accurately detect similarities based on ingredient lists. While this may be sufficient for elementary comparisons between recipes, it can fail to capture situations in which the quantities or actions to such ingredients

[1] LA Times
[2] NY Times
[3] Yummly.com

dramatically alter the end result. For example, several recipes may contain ingredients like flour, milk, sugar, and eggs, but differences in cooking technique can produce incredibly varied food items.

To capture this underlying complexity, Jermsurawong and Habash (2015) developed Simplified Ingredient Merging Map in Recipe (SIMMR), an ingredient-instruction dependency tree representation of recipe structure. In this dependency tree, the leaf nodes are each ingredient used in the recipe, which reveals insight into the underlying structure of the recipe, as well as relationships between sets of ingredients, instructions, and the order in which actions can be be taken to achieve the end result.

Others like Teng et al. (2012) construct an ingredient complement network, which analyzes similarities in recipes by capturing the relationships between ingredients. Teng et al. use these networks to examine and visualize which ingredients tend to co-occur frequently. Such networks reveal clusters of ingredients, which can be utilized to recommend ingredient substitutes or fed as features to a downstream recipe ratings prediction task.

Another common approach is to focus heavily on word-embeddings, as seen in the works of Marin et al. (2019) and Salvador et al. (2017). In these works, embeddings are learned using bi-directional LSTM, which enable a more in-depth understanding of each recipe based on both ingredients and preparation. These embeddings are incredibly useful for comparing recipes, as we can directly evaluate their quality using cosine similarity and recipe arithmetic. This work goes one step further by learning joint embedding of recipes and images.

Many of these methods have also been combined, as by Chang et al. (2018), where we see embeddings and clustering being used in an interactive tool to help analyze and identify usage patterns of particular ingredients and cooking methods.

Based on these works, we develop the approach of examining different models ranging from the simplest Bag of Words to more complex language models (e.g., BERT). Likewise, we use clustering to help evaluate and visualize the performance of our models on the ultimate recommendation task.

3 Methodology

3.1 The Data

Our feature vectors and word embeddings are trained on a subset of the Recipe 1M+ dataset, a large-scale, structured corpus from MIT CSAIL that includes over one million cooking recipes sourced from popular cooking websites (Marin et al., 2019). Further, due to the large volume of recipes and limited computing power, we isolate our training to only "baking" recipes (title, ingredient list, or instructions must contain the keyword "bake") – approximately 130,000 observations.

Following traditional recipe structure, each recipe in the data consists of a title, ingredients section, and instructions section. The ingredient section contains an ingredient list, along with required quantities. The instructions contain an ordered sequence of actions to perform on the ingredients.

Data Challenges We originally planned to focus our models on the cooking instructions, as these often capture both the ingredients and related actions. However, our initial attempts to train models on this data hit several computational bottlenecks caused by the length of the instruction strings and the long tail of the resulting vocabulary. Marin et al. (2019) acknowledge the issue of vanishing gradient as a challenge with this data. It is also important to note that often instructions are very ambiguous and require high-level semantic understanding (e.g., "mix wet and dry ingredients"). To address these challenges, we shifted instead to train our models on the recipe titles concatenated together with the recipe ingredients, which are orders of magnitudes shorter than recipe instructions (see Figure 1). Additionally, we limit our data to recipes that contain at most 75 words. Based on Figure 1 this has minimal impact to number of training observations, but greatly improves computational efficiency.

From our personal experience, we believe there is still some inherent meaning in the order of ingredients, as ingredients are often listed in the same order that they are used in the instructions, giving ingredient lists the flavor of simplified instructions. Similarly, recipe titles capture additional specificity (e.g. name of the dish, flavors, or defining actions).

Lastly, our data is largely unlabeled. We identify that a small portion of baking recipes are accompanied by nutritional categorizations (5,005 recipes), which can treat as "pseudo-labels" that can offer a basis for evaluating the strength and legitimacy of our learned clusters (see Section 4.2). *As such, the below analysis and model evaluation is limited to this subset.* This also allowed us to generate the required similarity matrices with the memory resources we had available for our web app.

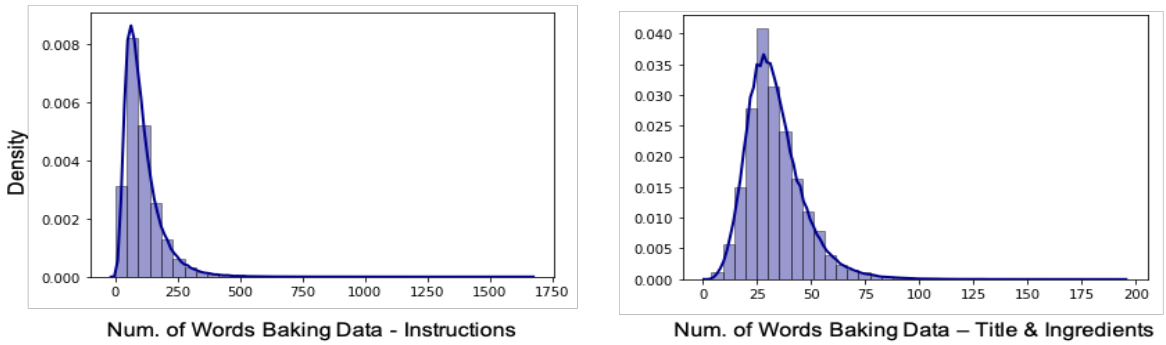

Figure 1: Baking Subset Statistics

3.2 Models

Bag of Words As a baseline, we implemented the Bag of Words model which represents each sample (a recipe title together with its ingredient list) as a vector of 0s and 1s indicating whether a word appears in the sample or not. Our related works (Jermsurawong and Habash, 2015) and other reviewed literature on content-based recommender systems cite this as a primitive, but popular and effective model for recipe analysis.

Doc2vec To overcome the fact that Bag of Words does not capture word ordering and distance between words in a sample, we choose Doc2vec (Le and Mikolov, 2014), an extension of Word2vec, as our next model. Whereas Word2vec captures word order and distance, Doc2vec goes further by adding a document vector, allowing us to indicate that each sample is an independent, self-contained recipe. Specifically, Doc2vec predicts the next word in many contexts from each recipe using both word and recipe vectors in multiclass classification.

LSTM In the spirit of Marin et al. (2019), we aimed to build a bidirectional long short-term memory (LSTM) recurrent neural network to learn recipe instruction embeddings. Without labeled data, we formulate our model as an autoencoder. LSTMs are often good at avoiding the vanishing gradient problem over longer sequences, as well as capturing semantic information that is lost in models like bag of words. However, as mentioned, the long tail of our vocabulary and computational concerns led us to abandon instructions in favor of recipe titles and ingredient lists. While this slightly weakens the case for using LSTM, still, based on the discussion in Section 3.1, we felt that the LSTM might be able to learn meaningful embeddings and continued to train it to test this.

BERT In the interest of evaluating the strength of recommendations produced by a state-of-the-art model, we augmented the pre-trained Bidirectional Encoder Representations from Transformers (BERT) model first introduced in Devlin et al. (2019). BERT offers an out-of-the-box language model containing pre-trained deep bidirectional representations from unlabeled text. The main add provided is that both directions are considered simultaneously at each layer, a feature that has been shown to improve performance on a wide range of NLP tasks. In order to observe how the out-of-the-box BERT model performs on our recommendation task, we fine-tune the model using our recipe data. Although our data may not have an explicit directional component, the considerations in Section 3.1 justify our experimentation with BERT.

4 Model Evaluation and Results

Evaluation Challenges As our goal is to develop a recommender system, an initial challenge was deciding how to evaluate the strength of recommendations across models. Note that we make recommendations by calculating the cosine similarity between recipe learned embeddings. Unlike other trained recommender systems, our data does not contain a ground-truth, pre-existing ratings, or any labels to help supervise the learning process.[4] Based on these limitations, we settle on a two-step approach to model evaluation.

First, we use the demo outlined in Section 5 to subjectively decide how one model performs relative to the others across varied inputs. Given a recipe input, humans can easily judge whether recommendations appear similar and reasonable. While this method of evaluation is subjective, we

[4] As discussed, evaluated data contain "pseudo-labels" in the form of nutritional categorization. This information is not used in training. See 3.1 and 4.2

believe it is meaningful, as it best mimics user perception of the quality of recommendations.

Second, we use unsupervised K-Means clustering, a widely used method of document classification (Singh et al. (2011), Allahyari et al. (2017)), on the first two principal components of recipe embeddings learned by each model to more objectively evaluate whether each model forms "natural" clusters. Principal Component Analysis (PCA) helps to reduce dimensionality of the word vector spaces, as well as aids as a tool for visualization. We also use the available nutritional information to evaluate whether any clusters form based on nutritional (i.e., sugar, fat, salt, saturates) content.

4.1 Recommendation Quality

For several recipes, we subjectively evaluated the recommendations produced by all four models. Example output for a recipe titled **Hazelnut Buckwheat Shortbread Dipped in Dark Chocolate** is shown below. This recipe is indicative because of the many component ingredients in its title. The recipe contains the following ingredients: **coconut oil, date sugar, sea salt, ground cardamom, ground cloves, vanilla extract, buckwheat flour, flour, water, and dark chocolate**. The top recommendations are as follows:

BoW: Basic Shortbread, Miniature Chocolate Hazelnut Cakes, Chocolate Chip Shortbread
D2V: Coconut Shortbread, Hot Fudge Ooze Cake, Devils Food Cupcakes with Vanilla Buttercream Glaze
LSTM: Best Ever Eggless Banana Oatmeal Muffins, The Best Chocolate Chunk Cookies Ever!, Blueberry Mango Coconut Crisp
BERT: Rye Bread for the Bread Machine, Boston Brown Bread, Strawberry Snack Cake

Subjectively, we see that Bag of Words gives the most direct recommendations, echoing many of the same words from the recipe title. Doc2Vec seems to pick up on the coconut oil and vanilla extract included in the ingredients but not immediately in the title. Lastly, LSTM and BERT seem to offer recommendations which are not "obvious" given the title and ingredients of the input recipe. Their recommendations offer a wider range / diversity of options, while still remaining in the realm of "breads" that are also mostly within the same "sweet" flavor profile. This example is representative of the results observed for other recipe inputs.

4.2 K-Means Clustering

We begin our clustering analysis by performing principal component analysis on learned recipe embeddings. This allows us to 1) determine how much variability exists in each set of embeddings, potentially reduce dimensionality, and 2) plot each recipe in 2D via the first two principal components.

Figure 2 shows that as model sophistication increases, more of the variation in the embeddings is explained by just the first principal component.

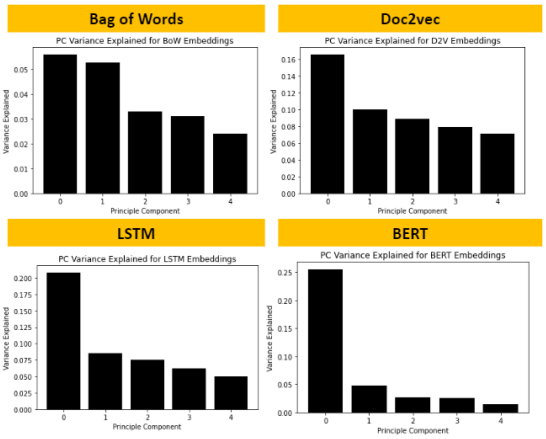

Figure 2: Scree Plot of Model Embeddings

Next we cluster using K-Means ($k = 2$), observing in Figure 3, that only Bag of Words displays two "natural" clusters. The more complex models do not appear to form meaningful clusters: the distinction straight down the middle is simply the algorithm minimizing distance between the required $k = 2$ cluster centers and the actual data points.

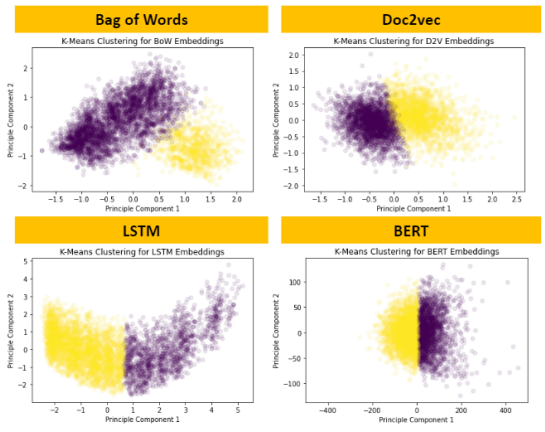

Figure 3: K-Means Clustering on Model Embeddings

Despite these results, we can still gain insight into model performance by subjectively reviewing

the most common unique words occurrences across recipes in each cluster. In the case of LSTM/BERT, this allows us to evaluate whether each direction of the first principal component tends to capture some semantically meaningful, human interpretable connection between recipes. We see that this results in something like sweet vs savory in BoW. However, the clusters for the other models are more difficult to interpret. Refer to Appendix D for word clusters.

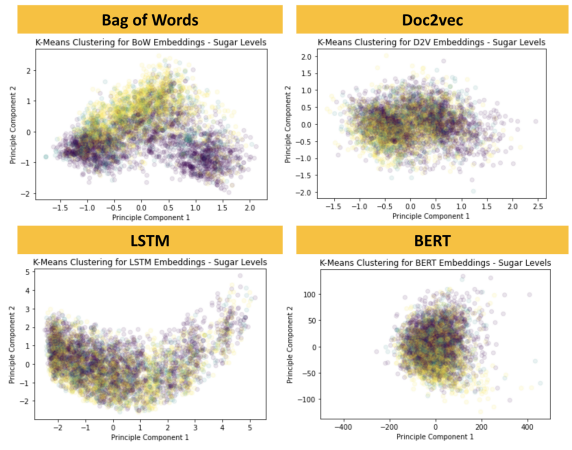

Figure 4: Sugar Level Clustering on Embeddings

Nutritional Clustering Our final method of evaluation exploits the nutritional information available for the 5,005 recipes detailed in Section 3.1. The nutritional information is already categorized into three groups for each of sugar, fat, salt, and saturates content (i.e. "high", "medium", "low"). Figure 4 shows the first two principal components with data points colored according to sugar categories. We see that all the models have some weak clustering. With the largest distinction between sugar categories appearing for BoW. Figures for other nutritional clusters appear in Appendix D.

In sum, we assess recommendations and embeddings with human review, PCA, and cluster analysis. Given computational constraints and data challenges, this review shows mixed results across the models. Generally, we see that simpler models produce the most consistent and direct results.

5 Demo

Our demo is an interactive website that allows users to see the recommendations provided by each of our models firsthand. The landing page offers a selection of random recipes from our subset of the Recipe1M+ dataset, which users are prompted to select according to their tastes. Our site processes this input and issues back a list of other recipes from our dataset that are calculated to be the most similar to the selected recipe(s). For evaluation purposes, recommendations are provided from each of the four implemented models. Our final demo is housed at http://nlp-recipe-project.herokuapp.com.

6 Conclusion and Future Work

Our simple/survey-like demo is an elementary step towards more sophisticated NLP-based recommender systems for cooking recipes. And, more importantly, provides insight on which aspects of a cooking recipe are most important for gauging similarity between recipes.

As we have seen in Section 2 (Related Works), there are many other NLP techniques, which could be integrated into our models. Once such extension that follows directly is utilizing clustering beyond a means of model evaluation. Instead, we can actively learn more sophisticated clusters of recipes to help identify types of cuisines, taste profile, and/or difficulty level by incorporating additional data into the training process or obtain labeled data.

Additionally, our system is currently self-contained to a specific subset of the Recipe1M+ dataset. Users calibrate the recommendation by selecting preferred food items that come from the dataset itself. In the future, we can augment our system to allow independent user input, including links to their own recipes, available ingredients, or specific flavors. By tracking clicks on recommendations, we can also learn which suggested recipes users like and which they dislike, which could then be fed into a reinforcement learning system.

Also, as outlined in Section 3.1, we had to make several compromises because of computational bottlenecks. By using stronger computing resources, allowing them to run for longer periods of time, and researching optimized data storage and manipulation techniques, we can provide even more context to our recommender system.

To conclude, the ability to analyze cooking recipes is not only relevant for a wide range of people, but may also pave the way for studying other technical manual/instruction-type data. Translating text such as cooking recipes into semantic representations can be a means to facilitating more sophisticated natural language processing tasks and inference such as question answering.

References

Mehdi Allahyari, Seyed Amin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. A brief survey of text mining: Classification, clustering and extraction techniques. *CoRR*, abs/1707.02919.

Minsuk Chang, Léonore V. Guillain, Hyeungshik Jung, Vivian M. Hare, Juho Kim, and Maneesh Agrawala. 2018. Recipescape: An interactive tool for analyzing cooking instructions at scale. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 1–12.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jermsak Jermsurawong and Nizar Habash. 2015. Predicting the structure of cooking recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 781–786.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2019. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.*

Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. 2017. Learning cross-modal embeddings for cooking recipes and food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

V. K. Singh, N. Tiwari, and S. Garg. 2011. Document clustering using k-means, heuristic k-means and fuzzy c-means. In *2011 International Conference on Computational Intelligence and Communication Networks*, pages 297–301.

Chun-Yuen Teng, Yu-Ru Lin, and Lada A. Adamic. 2012. Recipe recommendation using ingredient networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 298–307.

A Division of Labor

We certify that all collaborators contributed to this project equally. For transparency, certain tasks were led as follows. We evaluated the recipe recommendation quality together.

Ashley Chiu Related Works, Bag of Words, Cluster Analysis, written materials initial drafts

Ritvik Kharkar BERT, Demo/App construction

Adam Rohde Data cleaning and processing, Doc2Vec, LSTM

B Code and Data

All code and data is available on GitHub at `https://github.com/ritvikmath/nlp-recipe-project`.

Full Recipe 1M+ dataset is available from MIT CSAIL at `http://im2recipe.csail.mit.edu` upon registration.

Data included on Google Drive consists of our preprocessed baking subset and final app/analyzed data.

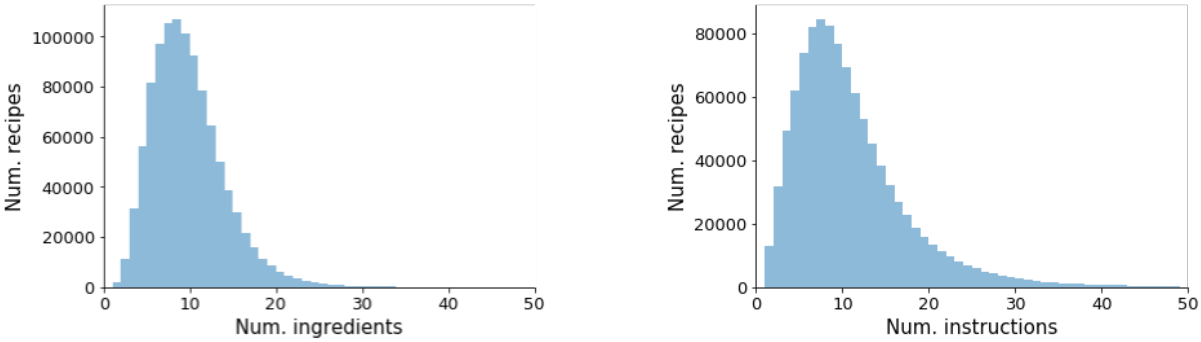

Figure 5: Number of Ingredients and Instructions in Full Recipe 1M+ Dataset

C Recipe 1M+ Example Data

Title: Hazelnut Buckwheat Shortbread Dipped in Dark Chocolate

Ingredients:

- 1/2 cup coconut oil
- 1/4 cup date sugar
- 1/8 teaspoon sea salt, plus a pinch
- 1/8 teaspoon ground cardamom
- 1/8 teaspoon ground cloves
- 1 teaspoon vanilla extract
- 1 cup buckwheat flour, plus extra for rolling out cookies
- 1/2 cup hazelnut flour
- 1/4 cup water
- 3 ounces dark chocolate (70% cacao)
- 2 tablespoons coconut oil'

Instructions:

1. Preheat the oven to 350 degrees F and generously dust your work surface with, buckwheat flour.
2. Lightly grease a cookie sheet and set aside.
3. In a large mixing bowl, using a wooden spoon or an electric mixer, cream the coconut oil and date sugar.
4. Add the 1/8 teaspoon salt, cardamom, cloves, and vanilla.
5. Mix in the buckwheat and hazelnut flours, and water.
6. When the dough is thoroughly combined, turn it out onto your work surface.
7. Liberally dust the rolling pin with buckwheat flour, and roll the dough out to about 1/8 inch thick.
8. Using your favorite cookie cutter, cut out shapes (I like to use a glass dipped in flour as my cookie cutter).
9. With a floured spatula, transfer cookies to the cookie sheet and bake for 20 minutes.
10. Remove from the oven; leave the cookies on the pan to cool.
11. Meanwhile, in a double boiler or using a metal bowl placed on top of a pot of simmering water, melt the chocolate, coconut oil, and a pinch of salt.
12. Whisk thoroughly and remove from the heat.
13. You may dip your cookies in the chocolate, or using a spoon, cover half of each cookie with a spoonful of chocolate.
14. Alternatively, you can drizzle chocolate on the cookies for a lacy look. (You may have leftover chocolate.)
15. Let sit for 1 hour at room temperature before serving.

D Results

D.1 Cluster Word Occurrence

After performing principal component analysis and k-means clustering, we analyze clusters by reviewing the most common unique word[5] occurrences (defining words) within each cluster under each model. We do this to evaluate whether clusters can be "labeled" by a specific character or flavor profile. As noted in Section 4.2, we see that that the number of defining words decreases as model complexity increases. In Bag of Words, we see more clear distinction between cluster 1 and cluster 2, where cluster 1 tends to include ingredients that are more "savory" (e.g. oregano, rye, tomato) or flavor-neutral (e.g. flax, yeast, rise), while cluster 2 favors "sweet" ingredients (e.g. cream, blueberries, almond, vanilla). We see a smilar behavior in Doc2Vec. Distinction becomes less evident in LSTM and BERT.

Bag of Words

Cluster 1: 'machin', 'fast', 'abm', 'easi', 'unbleach', 'soft', 'vital', 'bagel', 'tabl', 'instant', 'activ', 'extra', 'flax', 'oregano', 'french', 'pita', 'gluten', 'italian', 'sesam', 'hot', 'rye', 'sunflow', 'basil', 'rise', 'warm', 'basic', 'fluffi', 'loaf', 'molass', 'yeast', 'yogurt', 'virgin', 'tomato', 'cornmeal', 'flake', 'skim', 'bun', 'pizza', 'rosemari', 'nonfat', 'dough', 'quick', 'beer', 'homemad'

Cluster 2: 'lemon', 'syrup', 'peanut', 'mix', 'cream', 'ginger', 'pie', 'blueberri', 'ice', 'biscuit', 'bar', 'cranberri', 'corn', 'granola', 'low', 'sweet', 'bacon', 'chicken', 'cayenn', 'roast', 'almond', 'mustard', 'vanilla', 'pork', 'cocoa', 'sauc', 'cooki', 'soy', 'red', 'fat', 'chip', 'pecan', 'pumpkin', 'extract', 'nut', 'appl', 'pure', 'light', 'cornstarch', 'sour', 'mapl', 'egg', 'nutmeg', 'dark'

Doc2Vec

Cluster 1: 'machin', 'canola', 'easi', 'unbleach', 'thyme', 'plain', 'instant', 'activ', 'granola', 'flax', 'cayenn', 'sesam', 'parsley', 'sunflow', 'rye', 'honey', 'rise', 'warm', 'soy', 'yeast', 'fashion', 'yogurt', 'old', 'cornmeal', 'flake', 'pizza', 'mapl', 'dough', 'sea'

Cluster 2: 'clove', 'shorten', 'oatmeal', 'lemon', 'confectioners', 'mix', 'crust', 'peach', 'pie', 'blueberri', 'ice', 'biscuit', 'bar', 'low', 'pastri', 'heavi', 'beef', 'roast', 'pork', 'cocoa', 'cooki', 'molass', 'pumpkin', 'shortbread', 'appl', 'cook', 'cornstarch', 'sour', 'nutmeg'

LSTM

Cluster 1: 'roast', 'plain', 'cranberri', 'instant', 'pizza', 'easi', 'nut', 'granola', 'oatmeal', 'molass', 'blueberri', 'mix', 'pumpkin', 'dough', 'quick', 'bar'

Cluster 2: 'substitut', 'machin', 'canola', 'unsweeten', 'cornmeal', 'cocoa', 'carrot', 'soy', 'red', 'low', 'sour', 'parmesan', 'heavi', 'bacon', 'vegan', 'free'

BERT

Cluster 1: 'machin', 'plain', 'pork', 'cayenn', 'molass', 'mix', 'blueberri', 'sea', 'bar'

Cluster 2: 'cornmeal', 'flake', 'canola', 'rise', 'easi', 'soy', 'low', 'bacon', 'ice'

D.2 Cluster Nutritional Labels

Below are additional plots of the first two principal components, colored by nutritional labels for fat, salt, and saturates content. As discussed in Section 4.2, this information was not used to train our models and serves as a secondary method by which to analyze the quality of our clusters.

As discussed, nutritional content is "labeled" as high, medium, and low, which corresponds to the following: Sugar categories are over 15g (high), 5-15g (medium), and under 5g (low). Fat categories, 15g, 5-15g, and under 5g. Salt categories are over 1.5, 0.3-105g, and under 0.3g. Saturates categories are over 5g, 1.5-5g, and under 1.5g.

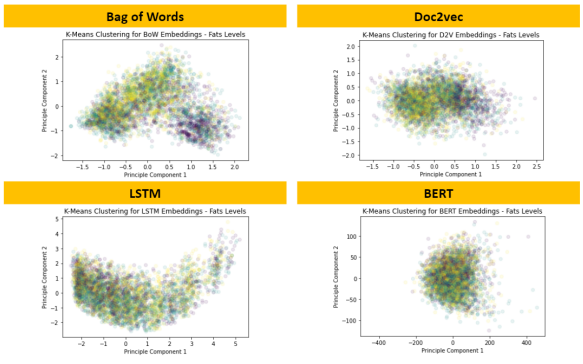

Figure 6: Fat Level Clustering on Embeddings

[5] Note that these are stemmed words.

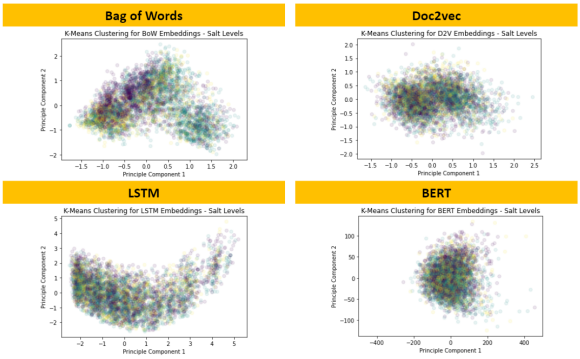

Figure 7: Salt Level Clustering on Embeddings

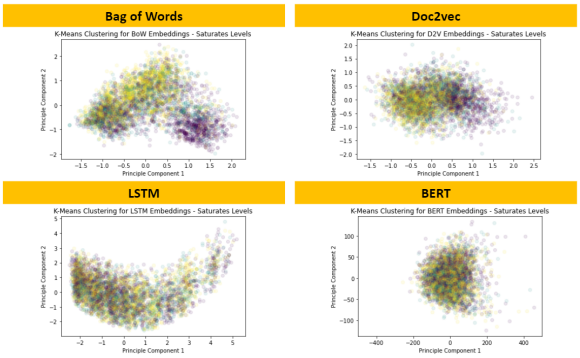

Figure 8: Saturates Level Clustering on Embeddings

For each nutritional category, bag of words tends to show the most distinguishable clusters, followed by Doc2Vec. However, overall, figures 6 and 7 show very little clustering related to fat or salt. Lastly, figure 8 shows some mild clustering based on saturates for BoW and Doc2vec.