Julia Layne - Natural Language Processing - April 19th, 2021

Q1. *a. [5 pts] What is Distributional Hypothesis in the context of distributional semantics? Give a short explanation with some examples.*

Distributional Hypothesis evaluates the semantics of a text by assuming that words similar in meaning will appear in similar context. As mentioned in the async, John Firth said “You shall know a word by the company it keeps.” The example we’ve frequently evaluated in this course is between newspapers and magazines. They can be seen in similar sentences:

I was reading the **newspaper** today. I was reading a **magazine** today.

By this hypothesis, we should expect these words: newspaper and magazine to be similar in meaning.

b. [5 pts] Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA) are two widely used techniques for topic modeling. Give a short overview of the two approaches and any similarities/differences between them.

LSA - Uses terms to correlate semantically to form a topic. Uses the assumption that texts close in meaning are formed with words of semantically similar words. Assume topics won’t share words. Uses Singular Value Decomposition to reduce the sparsity of the document matrix. Breaks text into vectors which are used to take the cosine similarity.

LDA - Uses probabilities to denote the likelihood a document belongs to a topic. This allows for a document to be a mixture of topics. Assumes topics can share words. Uses weights for each document to fall under a particular topic and updates these until a ‘best fit’ is found by looking at the document-topic density (alpha) and topic-word density (beta). Alpha looks at topic similarity between documents and beta looks at the similarity of words within a topic.

Both are unsupervised machine learning techniques that use keywords as a means of putting documents into topics.

Q2.

a. [5 pts] You are a Data Scientist for an e-commerce site for electronics which also supports 3rd party sellers. You would like to build a system to find and match the same products that sellers on your website sell so that you can present them in a single product page. You decide to use product titles to compute product similarity. Which similarity metric, Jaccard or Cosine, would you use and why?

Since we are looking at titles to compare, we probably do not care about the duplication of words in a title. The Jaccard similarity will tell us about the common words without caring about potentially longer titles with duplicate information.

b. Consider the following table which lists electronic items for sale on two ecommerce shopping websites. Products in row -1 are the same product, row-2 are different TV models of the same brand and row-3 are different products.

Product Title 1 (Site 1)	Product Title 2 (Site 2)
50 Inch Class H6570G 4K Ultra HD Android Smart TV with Alexa Compatibility 2.5" 2020 Model Black Silver White HDR LED	Hisense H6570G
QN75Q90TAFXZA crystal 2.5" Quantum LCD	Samsung crystal UN55TU8000FXZA QLED
EGLF2 50 Ultra Full Motion Articulating TV Wall Mount Bracket swivel full	VIZIO EGLF2

[10 pts] *Considering your answer to 2 a) will your similarity calculation approach work on this dataset? Explain with examples*.

Jaccard distance doesn't take into account the 'importance' of individual words showing up. Unless we could identify and more heavily weight the SKUs/Model numbers, the jaccard distance wouldn't take into account that matching to these rare words. The example where this break down is important is like H6570G in the first row where the titles are vastly different and only have ONE word in common. However, it is the most important word (the identifying ID from the company) that shows these are the same item. Instead looking at the 'rare' words and weighting them would need to be taken into account.

[10 pts] *Suppose that you are given IDF scores for all tokens (see Table below). Can this help you come up with a better approach for computing title similarity? Explain with examples.*

Now that we have the IDF score of words across all documents, we can then break it down into the weight of that word against the term frequency in the document. The problem with Model numbers being highly important should be handled. As shown below, these all had very high IDFs as expected with a unique item id. It would properly link the first row.

For the second, while they won't have matching ids, they have the rarer word **'crystal'** that is likely because it's a word used by the manufacturer (Samsung) and not others. This makes up for the fact both don't actually list the maker in the title. And the final row will be linked by the very high EGLF2, even though they don't otherwise match.

Product Title 1 (Site 1)	Product Title 2 (Site 2)
50(6.3) Inch(8) Class(8.5) H6570G(10.2) 4K(9.4) Ultra(6.6) HD(5.7) Android(2.6) Smart(6.1) TV(3.9) with(4) Alexa(6.9) Compatibility(15.6) 2.5"(5.7) 2020(6.8) Model(12.6) Black(6.8) Silver(7.8) White(12.6) HDR(12.2) LED (6.9)	Hisense(9.5) H6570G(10.2)
QN75Q90TAFXZA(13.7) crystal(11.3) 2.5"(5.7) Quantum(7.8) LCD(6.8)	Samsung(8) crystal(11.3) UN55TU8000FXZA(16.5) QLED(4)
EGLF2(15.6) 50(6.3) Ultra(6.6) Full(5.6) Motion(6.7) Articulating(2.6) TV(3.7) Wall(8.5) Mount(9.5) Bracket(11) swivel (8.5) full (5.6)	VIZIO(10) EGLF2(15.6)

Julia Layne - Natural Language Processing - April 19th, 2021

Q3.

a. [10 pts] Recommender systems are a subtype of information filtering systems that help users discover new and relevant items by presenting items similar to their previous interactions or preferences. Some famous examples of recommender systems are Amazon's "Books you may like" and Netflix's "Because you watched" carousels.

You are building a recommender system for your food delivery service startup and have data on co-purchases for food items f1, f2, . . ., fn (for example, food item f1 is commonly bought together with food item f4). How can you use techniques such as Word2Vec to recommend similar items to users who may have bought or show interest in any one of the items?

Word2Vec allows us to vectorize words and get a better understanding of where descriptor words for food can be swapped to get similar things:

Thai Chili Peanut sauce - Thai + Chinese -> Chinese Hot Chili Peanut Oil

This would work best with item descriptions, as names may not be as clear on how closely they relate. *Polish Hunter's Stew* And *Italian Sausage and Cabbage Stew* are very similar items, but only one contains their common ingredients: sausage and cabbage. Without taking into account descriptions, they would appear to be as similar to beef stew as they are to each other. Even though beef stew contains no beef or cabbage.

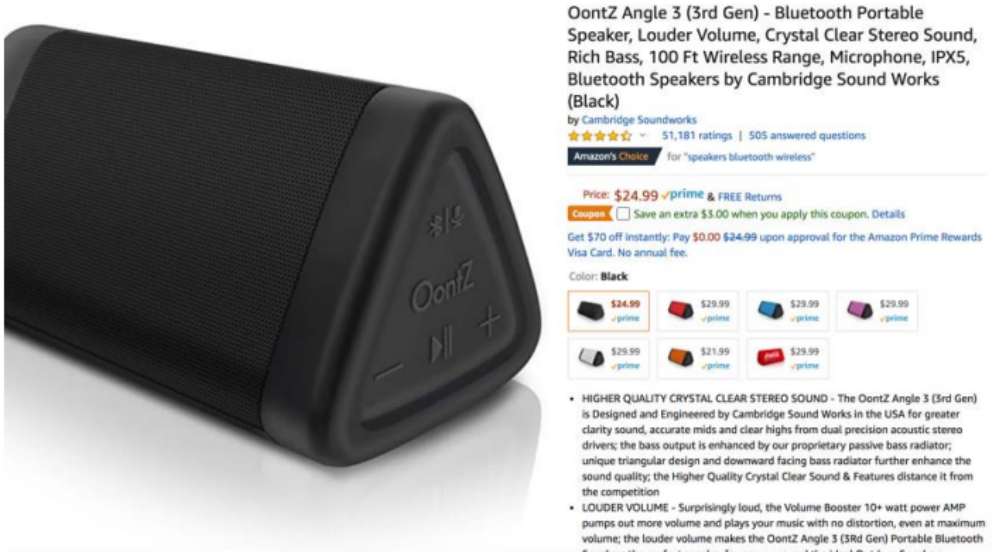
b. [10 pts]Word2Vec implements two different neural models: skip-gram and continuous bag of words (CBOW). Briefly explain the differences between the two models. Under which circumstances would you prefer the skip-gram model over CBOW

The Skip-gram takes in a single focus word input vector and expands it out into many possible context words as the output. CBOW does the opposite and takes in many inputs from a sliding window around the context words, and outputs a focus word.

In our example, it's probable that the startup wouldn't have a large training set to begin with and could prove their model/product on the skip-gram. As their user base, and therefore their data set, grows in size, the model will take too long to train on skip-gram. They would have to retrain less frequently and the results wouldn't be as good. Instead they should switch to using the CBOW method, which will train much faster and make use of their large dataset.

Q4.

You are building a product classification system for an online electronics store. The system should classify an incoming stream of millions of products to one of the 3000+ leaf level product types in the taxonomy such as laptops, smart TVs, wireless headphones, car speakers, among others. The system should be very precise because it's important to assign products to the right category to facilitate the customer shopping experience. Each instance in your dataset has product title, description and image fields. See example below:



a. [5 pts] What features would you use for your machine learning-based classifier?

Product title - These titles can get somewhat cluttered with information and possibly other features taken could be removed from the title to de-duplicate information that is later in the document. This long title is realistically just: OontZ Angle 3

Product Seller (the 'by Cambridge Soundworks') - certain brands/sellers may be categorized differently, such as manufacturers that are considered luxury brands.

Stars and Reviews - This would be both the hard number '3 stars' and the text itself

Product features (crystal clear sound, louder) - this and other features should be pulled out for comparison to other products. Sentiment around these individual features could be used to determine how 'good' these marketed features are. So a search for 'best loud bluetooth speaker' could actually be based on what users thought were the best, not just the seller describing the item as loud.

Product color - Since color preference can be very important in some item searches

Product price - We'd want to be able to potentially change categories: cheapest, moderately priced, or luxury

Questions and answers - There may be missing information in the description that could be pulled from the questions and answers. Such as if this didn't mention 'loud' or if an individual may ask if this speaker can be heard in a large room.

Product Image - this may be part of a well to help group items that are nearly identical but may have different names/resellers. This could be used to group these items.

b. [5 pts] Assume that you only have access to product titles in your dataset (i.e., you have less data to play with) instead of product titles, description and images. How will this affect feature engineering and the NLP pipeline for your classifier?

It will limit the information to pull out. However, much information has been crammed into the titles and we will still be able to grab many features above. Named Entity Recognition could pull out the brand or seller in the titles. We then could pull out features such as 'crystal clear sound' and bluetooth. These might be found by comparing items and taking frequent terms among products as likely features. It also states the color of the item. These could be pulled out to make them more discrete, or simply used in direct word comparisons of titles.

The most important data we would miss in this is grabbing sentiment around these features. Instead of being able to pull out what users thought of how 'crystal clear' the sound is, we'd just have to take that as a feature from the title. This would make it harder to group items for queries like: best clear bluetooth speaker. We might list this item, but then the reviews say that this is actually useless.

c. [10 pts] Obtaining training data is paramount for a large-scale classification system. You have a limited budget and can't hire an army of analysts to manually label every single instance. Discuss some strategies for obtaining training data for the classifier.
We would need to use an unsupervised learning technique based on the words. A technique such as LDA would allow us to get these items assigned to a topic (the catalog categories in this case). We would need to find a lexicon around electronics if available and expand upon it with any updated tech buzzwords. These could be identified by getting the most frequent words not appearing in existing lexicon. LDA would also be a benefit because it has a probability across many categories, and we could use this probability to rank them within those assigned categories. Those with higher topic values would show up earlier in a search or category filter.

d. [5 pts] How would you handle products that are misclassified?

Since we don't have analysts, we probably also don't have testers. The user's activity would help us with misclassifications. When an item appeared in a search or category filter, but did not receive any clicks we would want to save that as data that could be used to lower the weight of probability for the given topic it was shown in.

Julia Layne - Natural Language Processing - April 19th, 2021

Q5.
a. [10 pts] Sentiment analysis: consider the following review of a restaurant:
“I took my father out for dinner to Le Bistro on New Year’s Eve. The décor and service were fantastic. We enjoyed the food, especially their French countryside specials and their Chardonnay collections. However, my father thought the menu prices were a bit on the high side. Valet parking was also expensive. Overall, we definitely recommend Le Bistro for special occasions!”
Overall rating: 8 stars out of 10 “

Identify the opinion object(s), feature(s), opinion(s), opinion holder(s) and opinion time in this review.

Opinion Objects:
    Le Bistro
Feature:
    decor, service, food, ‘French Countryside specials’, chardonnay, prices, menu, valet
Opinion Holders - Opinions:
    I (reviewer): decor fantastic
    I (reviewer): service fantastic
    Father and I: enjoyed the food
    Father and I: enjoyed the French Countryside specials
    Father and I: enjoyed the chardonnay collections
    Father : menu prices are high
    I (reviewer): Valet Parking is expensive
    Father and I: (positive) recommend Le Bistro
Opinion Time:
    New Year’s Eve

b. [10 pts] Design a sentiment analysis system for restaurant reviews (see example in 5a). Your answer should make use of the techniques discussed in class. The output of the system should assign a sentiment label of Positive or Negative to reviews

We already have overall ratings in the form of stars. I think it would be more interesting to pull out the features for ratings as positive or negative. This could help in finding a place with the ‘best burger’. Since opinions are given on individual features of the review we should break up the review into sentences.

These clauses would need to be NP Chunked to get the features of the review through POS tagging. These would likely be the nouns and direct objects within these clauses. Within these clauses we could use a lexicon like SentiWordNet to determine the overall opinion on the given feature.

We’d have to go for entity recognition (unlikely people will use names in their reviews as seen above). We would need to find things like ‘father’, ‘husband’, ‘coworker’, and ‘I’. These would be

tagged as the opinion holders. Opinions not held by the reviewer (me/I) may need to be lowered in weight as they are not leaving the review, but may influence the reviewers overall score.

These positive/negative features pulled out of the review could then be averaged to mark the entire review as positive/negative. As the reviews are already rated with stars, we know that 5 and below are negative. This would also give us something to test against, we would expect the overall review to reflect the average sentiment among the features.