CS 7337 – Natural Language Processing

Final Exam

Ben Goodwin

April, 2022

Instructions: Clarity of answers is more important than length of answers. Although not required (unless indicated otherwise), feel free to use graphs, charts, visuals, et al in your answers if you feel these artifacts can help support your answers. There are no bonus points for using these artifacts. Submit your answers in PDF or Word document format.

Due date: See course wall announcement.

Q1.

1. [5 pts] What is Distributional Hypothesis in the context of distributional semantics? Give a short explanation with some examples.

The main point of this hypothesis states that there is a correlation between distributional similarity and meaning similarity, which allows utilizing the former to estimate the latter. Furthermore, distributional approaches to meaning acquisition utilize distributional properties of linguistic entities as the building blocks of semantics. A simpler definition of the distributional hypothesis states, “words which are similar in meaning occur in similar contexts.” The general idea behind this hypothesis states that there is a correlation between distributional similarity and meaning similarity.

*“a word is characterized by the company it keeps”* – John Rupert Firth

NLP use cases: Semantic similarity, word clustering, sentiment analysis, document clustering

Examples:

If we want to find a word like *magazine,* we can look for words that occur in similar contexts, such as *newspaper*.

“I was reading a *magazine today”* and “I was reading a *newspaper* today”

“The *magazine* published an article” and “The *newspaper* published an article”

“He buys the *magazine* every day” and “He buys the *newspaper* every day”

If we want to find a word like *duty*, we can look for words that occur in similar contexts, such as *responsibility*.

“His *duty* is to serve the nation” and “His responsibility is to serve the nation”

“I have a *duty* to stop the criminal” and “I have a *responsibility* to stop the criminal”

1. [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.

Latent Semantic Analysis (LSA):

Supports organic topic modeling.

LSA-based topic modeling tries to find groups of words associated with the largest variances between documents in the corpus.

It answers the question, “What small groups of words, when found in documents, predict those documents from being very different overall from the rest of the corpus”?

Dimensionality reduction will get us from a board vocabulary into a smaller number of groups with a smaller number of words in each group that do the best job of differentiating some of the documents in the corpus to other documents in the corpus.

Tries to separate the corpus by means of topic words.

LSA starts with a large term document matrix, but then it creates a topic-to-topic matrix. Leaves most words behind, reduces the words in the vocabulary and chooses ones that to a good job separating documents.

LSA Steps:

1) Create a term-document matrix (using TF-IDF)

2) Use Singular Value Decomposition (SVD) to reduce the number of rows but preserve column similarity because the matrix can be very sparse and noisy.

3) Compare documents by cosine similarity between two vectors formed by two columns.

Latent Dirichlet Allocation (LDA):

Intuitively, LDA construes topics as groups of words that have high co-occurrences among different documents in the corpus.

Different topics can share keywords when those keywords co-occur frequently enough across the different topics.

LDA uses a probability distribution of how likely it is that a word is in a topic and that a topic is assigned to a document.

LDA keeps on moving words into different topics, and reassigning topics to documents in a different way until its maximized the values.

LDA Process:

1. For every word *k* in every document *d*, compute:

P(z|d) = proportion of words in document *d*, assigned topic *z*

P(k|z) = proportion of assignments to topic *z* among all docs having word *k*

2. Now reassign word *k* in document *d* to whatever topic the highest *P*(*z*|*d*) x *P*(*k*|*z*) from all topics that you can substitute for *k*.

Now repeat these steps over and over.

Similarities/Differences:

In LSA the question of “what small groups of words when found in documents, predict those documents being very different overall from the rest of the corpus” becomes unlikely that any of the topics will share keywords, compared to what LDA would produce, LDA can overlap, LSA is wired to find these separations. (If keywords occur across multiple topics)

LDA like LSA it sort of discovers what the topics are that are latent.

Primary difference between LSA and LDA is that LSA uses a topic-by-topic matrix, which looks at how often a topic occurs on the same document as another topic, LSA tries to avoid sharing of document topics. LDA instead looks at probability distributions, and asks, what is the probability of a word being assigned to a certain topic, and the then document distribution over the topic. Probability of a topic being assigned to a particular document.

Q2.

* 1. [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?

Jaccard similarity is good for cases where duplication does not matter, and cosine similarity is good for cases where duplication matters while analyzing text similarity.

I would use Jaccard similarity for the reasons above, furthermore as an example:

“LED TV HD TV blu-ray TV” will have the exact same Jaccard similarity as “LED TV HD blu-ray” which would not be the case for cosine similarity.

In the context of this problem a more desirable result for presenting results on a single product page is via Jaccard.

* 1. 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 2a) will your similarity calculation approach work on this dataset? Explain with examples.

The approach I took was to use the Jaccard similarity because in the context of the written problem, we needed to have multiple potentially duplicate products return after running a search query, and in this case the solution seemed to be Jaccard similarity.

My thoughts were that the Jaccard similarity would perform well on the first and third row, and not as well on the second.

Row 1: Where we have the same product, because Jaccard can handle the case where we have duplication. In this row we have “H6570G” appearing in both cells, so Jaccard will work here.

Row 2: Different TV models of the same brand, between the two cells, we have a single intersecting word, “crystal” and other than that the cells are completely different. Jaccard will have trouble here.

Row 3: Different products all together. There is only one intersecting word here, “EGLF2” and this concerns a completely different product than the one referenced, one is a clearly defined TV mount whereas the other cell seems to refer to something completely different or similar, we aren’t sure due to the lack of information. Again, Jaccard falters here.

To support my reasoning and answers I inputted each of the product descriptions into a Jupyter notebook and computed both the Jaccard distance and the Jaccard similarity.

In the code block below, I computed the Jaccard distance Table

Description automatically generated with medium confidence

In the code block below, I computed the Jaccard Similarity

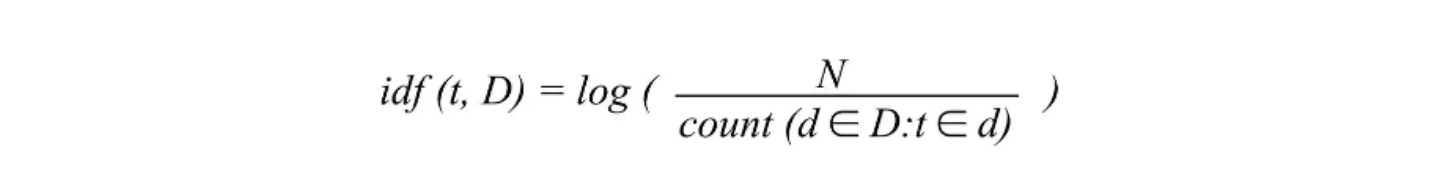
Graphical user interface, application

Description automatically generated

[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.

|  |  |
| --- | --- |
| 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) |

Inverse document frequency looks at how common (or uncommon) a word is amongst the corpus. IDF is calculated as follows where *t* is the term (word) we are looking to measure the commonness of and *N* is the number of documents (d) in the corpus (D). The denominator is simply the number of documents in which the term, *t*, appears in:



There are a few approaches here for computing title similarity bow that we have IDF scores for all the tokens:

We can compute the cosine similarity matrix which contains the pairwise cosine similarity score for every pair of sentences (vectorized using tf-idf)

Since this is a task somewhat akin to a search engine task, we could use these idf scores to rank these results based on relevance, with results which are more relevant to the user having higher idf scores.

Idf will also help with distinguishing relevant and non-relevant terms, hence the *inverse document frequency* incorporates the weight of terms that occur very frequently in the document set.

Q3.

1. [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?

Word embedding features create a dense, low dimensional features when compared to TF-IDF creates a sparse, high dimensional feature, these embeddings also capture the sematic meaning very well.

Word2Vec is a simple neural network model with a single hidden layer. It predicts the adjacent words for each and every word in the sentence or corpus. We need to get the weights that are learned by the hidden layer of the model and the same can be used as word embeddings. In our food example we can assume f1 as our input word, and we have a defined context window of two. This means that we are considering only the 2 adjacent words on either side of the input word as the adjacent words.

Word2Vec then picks the nearby words (words in the context window) one-by-one and finds the probability of every word in the vocabulary of being the selected adjacent word. In this case our vocabulary can be fined as food items f1, f2,…,fn.

Note: the context window can be changed as required.

Word2Vec has pre-trained embeddings to help get the vectors we’re after. In this context we need to convert the foods into vectors and find the similarities between these vectors to recommend foods to co-purchase. The other option is to train our own word embeddings assuming we have a large enough dataset.

To get Word2Vec to produce recommendations, Word2Vec takes the word (food) and give a D-dimension vector, Word2Vec will then calculate Word2Vec for the food. This is how to compute average Word2vec.

Now on to the recommender system…

A content-based recommendation system recommends foods to a user by considering the similarity of foods (what was often purchased with what), it can also do novel things like consider a user’s previous purchase history to recommend a food. To implement this, we can use cosine similarity

1. [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?

In a continuous bag of words model – The distributed representations of context (or surrounding words) are combined to predict the word in the middle.

Skip-gram model – The distributed representation of the input word is used to predict the context.

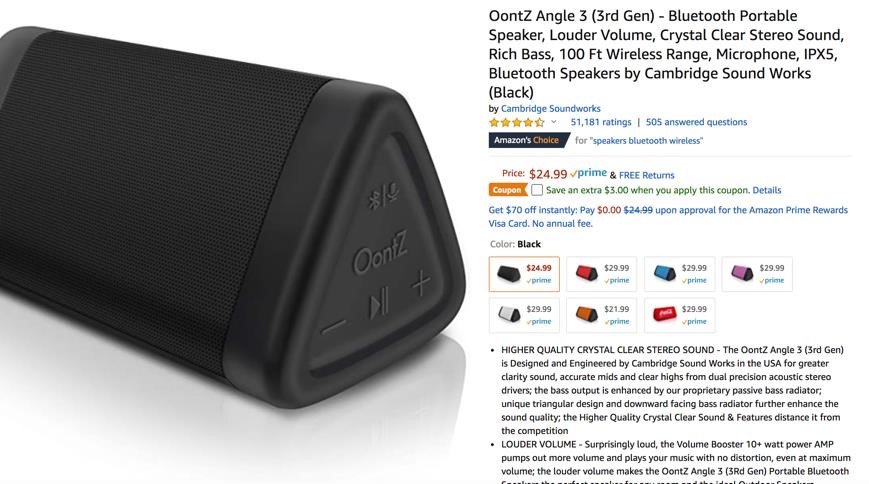
The two are somewhat mirrored versions of each other. Some differences include with continuous bag of words learns better syntactic relationships between words while Skip-gram can better capture semantic relationships. Another difference speaks to sensitivity to rare and frequent words, since Skip-gram relies on single words, it has less sensitivity to overfit frequent words, whereas continuous bag of words is prone to overfitting frequent words because they appear several times along with the same context. Skip-gram tends to be more efficient in terms of documents required to achieve good performance, considerably less than continuous bag of words.

The answer to the last part of the question is that it depends. If speed is the necessity, continuous bag of words is typically faster to train and can do a better job representing more frequent words.

If we are placing importance on representing rare words, skim-gram is preferred. However, on the converse if we don’t have time to train and rare words aren’t ultimately important, then continuous bag of words wins.

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:



1. [5 pts] What features would you use for your machine learning-based classifier?

I believe a good start to classify products to their most appropriate category, but at a micro and macro level. The example above could be listed as *Bluetooth speakers* (micro category) and audio devices (macro) category. To determine the macro and micro categories, the developers can look at some summary statistics and see all of the macro categories and ideally some of the micro-ones too. The thinking here behind macro and micro is that this would prevent users from erroneously find home theatre speakers when they were looking for a portable Bluetooth speaker. My approach would have the features (macro) as sort of an umbrella and the more specific ones (micro) falling under this umbrella and take all of these features and apply them towards all of the products. Product name and description would be the features put under the most scrutiny for analysis since I feel they could derive the most value, despite text not being that easy to work with descriptions of electronics are straightforward, however images associated with these may not be as such. Using the example above, the distinguishing shape is a triangle, with consumer products I most commonly associate this with Toblerone (the chocolate bar). We also must emphasize using both learning and hand-crafted rules (written by domain analysts) extensively.

1. [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?

Here the biggest downside would be less data to validate classification with, using all three we could form a very probable picture of what a product is, with only the product title we are a bit more limited on accuracy. Right off the bat, I would suggest getting more humans involved in the process to evaluate titles and correctly classify, but in their absence the feature engineering and NLP pipeline will change.

Obviously if we lose data and are left with only product titles, more emphasis will need to be placed on these.

We can look at length analysis to maybe discover that some categories are systemically longer than others, or we can deduce that a product has some of its description in the title.

We can also look at sentiment analysis again to extract all the possible meaning from the product title.

We can also look at named-entity recognition in the title to also determine tag named entities.

Word frequency could also be important here.

Don’t stem or use stop word removal

Bigrams and degree-2 polynomial mappings features are beneficial for product title classification.

1. [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.

In the case where we cannot obtain large amounts of training data and can’t have analysts handle all labeling we must strike a balance of combining crowdsourcing and in-house analysts to evaluate and analyze the system, to achieve an accurate, continuously improving, and cost effective solution for classification.

This can be accomplished by machine classifying incoming products and then using crowdsourcing to continuously evaluate the results and flag cases judged incorrect by the crowd.

You could also consider a rule-based approach, however this is not practical and is a slow and daunting process.

1. [5 pts] How would you handle products that are misclassified?

A first and easy answer would be outsourcing/crowdsourcing, however this can be very expensive and not easy to scale.

A secondary approach would be to:

Classify incoming products, then use crowdsourcing to continuously evaluate the results and flag cases judged incorrect by the crowd.

Analysts examine flagged cases, and fix them by writing new rules, relabeling certain items, and altering developers.

The newly created rules and the relabeled items are incorporated into the system, and the developers fine tune the underlying classification algorithm.

For the items that the system refuses to classify, the analysts examine them, then create hand-crafted rules as well as training data to target those cases. The newly created rules and training data are incorporated into the system, and it is run again over the product items.

Q5.

1. [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 object: Le Bistro (Restaurant)

Features: Dinner, Décor and service, food, French countryside specials, Chardonnay collections, menu prices, valet parking, special occasions

Opinions: The father, “my father thought the menu prices were a bit on the high side”, and the writer, “we enjoyed the food, especially their French countryside specials and their Chardonnay collections”, “Valet parking was also expensive”, “overall rating: 8 stars out of 10”

Opinion holder: The writer and the father of the writer.

Opinion time: The meal was had during New Year’s Eve.

1. [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.

Graphical user interface, Teams

Description automatically generated

Steps to analyze sentiment for restaurant reviews:

1) Text processing

Cleaning of text

Tokenization

Stemming

2) Word Embedding

Word2vec

3) Model

4) Performance Metrics

5) Summary

As part of step 1, we’ll clean the text and do things like: lowering case, removing special characters, removal of stopwords, removal of numbers, removal of whitespace

With step 2, we’ll tokenize. Here we will remove and replace suffixes from a token to obtain the root of the word. Probably implement using NLTK.

With step 3, we will do the heavy hitting. This is the step where we will use Word2vec to embed our words into vectors. We will be left with our corpus. This technique takes a large corpus of text as its input and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus assigned to a corresponding vector in the space.

Next up we will have to fit a model, for this task we have a binary output (positive or negative review). We will use logistic regression, as it outputs the probability of the input belonging to the class.

We can now talk about model performance metrics, we will use precision and recall as our primary performance metrics.

Precision is the fraction of the relevant instances from all the retrieved instances. It helps us to understand the usefulness of the results.

Recall is the fraction of relevant instances from all of the relevant instances. Recall helps us understand the coverage of the results.

We can combine these two to produce the F1 score which is the harmonic mean of these two metrics.

I think here it is very important to continually monitor this model as the vocabulary may change at a moment’s notice.