**CS 7337 – Natural Language Processing**

Final Exam 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.

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

*The distributional hypothesis states that there are certain frequencies that can be expected for the occurrence of words in each corpus (i.e. words which are similar in meaning occur in similar contexts). An example of an expectation of a distribution could be that of stop words such as* ***the, and, a*** *– in English, these occur frequently and hence a corpus of English text may contain high frequencies of these words. Other examples could include* ***names or proper nouns****, which would be expected to occur at much lower relative frequencies than say* ***common adjectives or verbs****.*

*One can use the distributional hypothesis to compare two texts and assess whether they have similar meanings. If the relative occurrences of words in two bodies of text have high overlap that could be an indication of them having a common subject or 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.

*Latent Semantic Analysis is a method of matrix simplification (or* ***single value decomposition)*** *that makes use of* ***term frequency-inverse document frequency (TF-IDF)*** *to weight a feature matrix in such a way as to lower the impact of high frequency terms with respect to those of lower frequency. The idea is that words which occur frequently carry less information and should not be weighted as highly as words that are more idiosyncratic.*

*Latent Dirichlet Allocation is another probability-based technique, but it does not rely on weightings in the same way that LSA does. Instead, probabilities are determined for each word at the topic and document level for each topic. LDA leverages the probability that a particular vocabulary (or group of words) pertains to a topic and then uses those probabilities to calculate what topic the overall document may be referring to.*

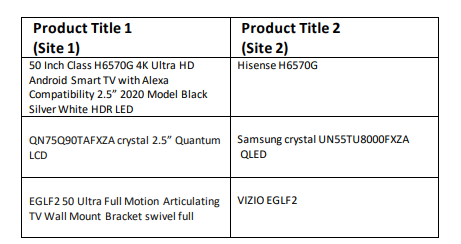
## Q2.

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?

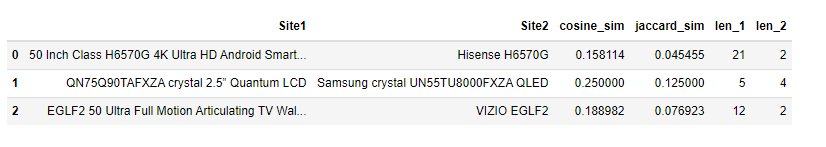
*Since the relative size of the text in the is instance is expected to be small – the relative distance between any two given samples of text can be easily represented with* ***Cosine*** *similarity. Cosine similarity scores* ***are less affected*** *by the relative length of comparison between two texts – whereas Jaccard similarity* ***is affected*** *by the length of one text versus another.*

*In our case, if we expect that the title of a product may contain the* ***item # or sku*** *in both descriptions – we would want a very high similarity score, independent of the length of each description – Cosine similarity is likely to work better.*

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.



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



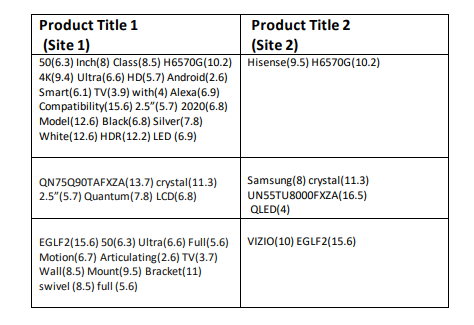
*In the first case – Jaccard Similarity performs rather* ***poorly*** *- assigning a value of only 0.045 to the pair of titles that represent the* ***same item****. This is because the lengths of each text are quite different. Despite containing the same unique ID, Jaccard has scored them the lowest similarity out of the bunch.*

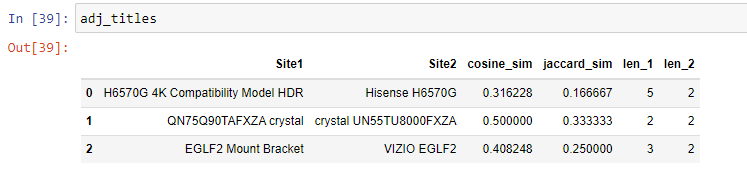
*Cosine similarity does an excellent job on the first pair, but overall does a poor job on pairs 2 and 3. While score for pair 2 (same manufacturer, diff item) should be fairly high – it shouldn’t be higher than the score for pair 1 (same item). It also does a poor job of matching the item in pair 3, but as a retailer – having the* ***articulating arm*** *for mounting the television listed in site 2 could drive sales of that item as well.*

*Overall, I would stick with the original decision to use Cosine similarity over Jaccard. Were it more of a guarantee that the lengths of each text were similar, perhaps Jaccard would perform better.*

[10 pts] Suppose that you are given IDF scores for all tokens (see Table below).

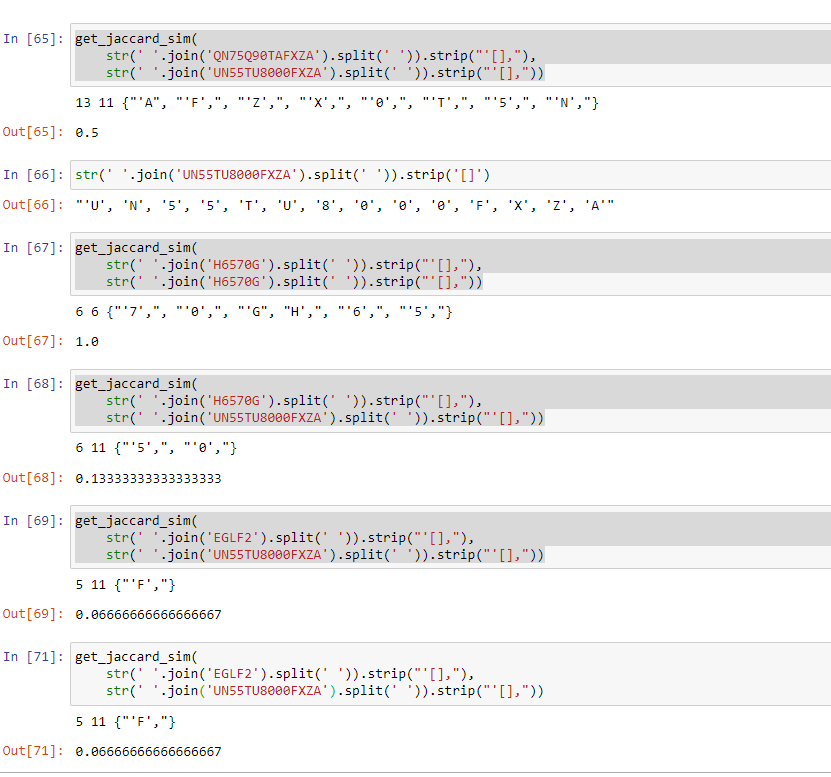
Can this help you come up with a better approach for computing title similarity? Explain with examples.





*A simple way to adjust the values for Jaccard similarity to be more accurate would be to apply an IDF filter. If we filter out values that are below a threshold of 9 the longer texts are shortened significantly and the Jaccard scores more closely match those of Cosine similarity. This is particularly true of the first pairing – the Jaccard score raises by a factor of 4. The second and third pairings remain hard to discriminate, the high IDF score assigned to* ***crystal*** *in pairing 2 makes matching appear better than it really is. It’s more a coincidence that the values for similarity are high – any two items with crystal in the names would be related similarly. Ideally, a letter level comparison of each word could lead to better results. The model strings in each site description are similar enough that they would score highly on a similarity score (they both end in the string “****FXZA”****) – weighting each word by it’s similarity to other words (on a letter by letter basis) could lead to better results as well. Searching for substrings of unique combinations of characters as substrings – in this case – grouping by 4 and iteratively searching* ***only the strings that do not appear in a common dictionary (searching known words for these patterns could be computationally expensive and probably unnecessary)****. These “words” are likely ids or serial numbers (or manufacturer ids) - and can be weighted by their level of overlap. Assigning the two ids a non-zero weighting as the effect of including them in the dot product. For the example in this case it may not make a difference, but if we wanted to match* ***many******items*** *to other item descriptions, having a heuristic search algorithm for identifying skus/ids is a great start.*

*An even simpler approach is to calculate the Jaccard similarity of each word to one another – this would be helpful for matching items together where sku-like labels were in the text:*



*Examining the jaccard scores would show a relationship between the items of the same manufacturer (0.5) and the same item (1.0) – whereas each of the skus would compare poorly against one another (when not of the same manufacturer).*

# 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?

*Setting up a weighted bag of words in such a way so that the key words that are in the description of item 4 get a high weighting (or a lower distance in cosine similarity space) to those in item 1. That way, when a user searches for any of the key terms in either item, the other item is likely to appear. Basically, this method amounts to weighted TF-IDF technique and is simple to implement as it only involves modifying the affinity matrix via the dot/hadamard product.*

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?

*These two methods amount to different ways of labelling (or embedding) training data.*

*Continuous bag of word neural models aim to predict a word by examining the* ***context (surrounding words)*** *and determining the most likely* ***single******word*** *that fits. Mathematically, the input to the model could be****wi−2,wi−1,wi+1,wi+2*** *the preceding and following words of the current word we are at. The output of the neural network will be****wi*** *the current word* ***(in the case of a window size = 5).***

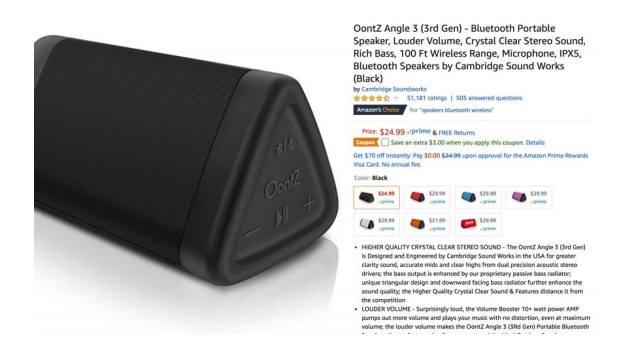
*In a skip gram model – the aim is to derive the* ***context*** *given the single* ***word (wi).*** *Defining the window size parameter in this method sets the maximum windows size for the algorithm and words within the predicted context (****wi-1****…wi-window) are weighted by their distance from the target word. In this way, the most likely context of a given window size can be estimated. Skip gram models also employ a mechanism for introducing* ***non-continuity*** *– the skip parameter allows n-grams to be formed that are not a simple shifted window (as is the case with CBOW).*

*Skip gram algorithms typically work well with smaller training sets and are more likely to predict specialized terms of speech that a CBOW approach is likely to miss due to low probabilities of word occurrence. For training speed and common words, the CBOW approach would be more appropriate.*

*Skip gram could be more useful for search engine type solutions (deriving context from a string of text), whereas predicting the next word a user may type could be better suited for a CBOW.*

# 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?

*Given an infinite budget one could use image recognition for categorical labels by taking a sample of 3-6K samples and paying some interns to label each image with a variety of categorical tags. From there, you could train an image classifier to predict tags. This could be rather inefficient and costly. It’s more likely easier to use a pure NLP approach using the short text string in the product title along with the text of the description (all stripped of stop words, n-gram tagged, and potentially lemmatized) and then training a clustering algorithm such as k-means to predict the cluster of each new text string.*

*Once a cluster is determined, the values describing that cluster (headphones, wireless, apple, mac, ipad etc) can be linked back to a key entity database which is already likely maintained in the backend system (lists of companies, products, manufacturers etc). This is a wealth of data that can be leveraged to classify text strings of unknown products.* ***Apple*** *is likely listed as a company in this database, hence the context assigned to Apple for this classifier should allow for higher weight to be assigned to the word apple as a Proper Noun (****OontZ*** *in this is example is easy to assign context to as it doesn’t have meaning other than as a PN). Associations for the word* ***head*** *could be treated similarly - almost all contexts would involve the physical context of head (a human head for* ***headphones****, or a* ***headset****) – it would less likely appear in the context as in the head of a group (****head of the class****) – though these associations have more balanced probabilities in the general corpus of English language, given our context we must make adjustments to our priors in order to make more accurate classifications. Allowing a human to modify the rules of context in a straightforward way (by tweaking probabilities) can impart context to an ML algorithm that would otherwise be absent.*

*In the case of a product that contained just a few the words in the example above “****OontZ, bluetooth, stereo, speakers, 100 ft, Cambridge…****” one could infer the manufacturer, the likely type(s) of equipment/device, the size among many other categorical classification.*

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?

*The approach describes in 4a) largely accounts for this limitation. One can create a custom regex parser using a normalized database of information like those readily available in most inventory systems. Creating custom regex tags based on the values in a table of* ***manufacturers****,* ***products****,* ***colors****,* ***units*** *etc would allow context to be assigned to a variety of input strings. Most of these tables look like this:*

* *Id*
* *Short Name*
* *Long Name*

*Building a custom regex parser that maps back to these types of tables can be done with some simple SQL and python coding.*

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.

*If one really wanted to expand the corpus of labeled data in a cheap way – web scraping is always an option. To the extent that competitors or manufacturers have product listings of their own, one could scrape the pages for* ***item descriptions****,* ***sku/product ids******and any other categorical tags that may exist on the page.*** *To the extent that other sites allow searches of their own (or have backend APIs) – one can pass in common search terms to see what items are returned by* ***other classifiers****. This could be very helpful in creating “labeled” training sets in a quick/automated way.*

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

*To identify products that are misclassified one could track click activity on an item. Items that are not being clicked at all are likely to be in categories that are incorrect or are not showing up at all in search results (even worse!). This could be due to a variety of issues (misspellings, bad skus etc) but limiting the search space to the items with very low frequencies could allow us to tweak the regex parser to account for some unexpected edge cases (Apple/Appel or Google/Googel transpose errors in naming etc).*

*Ideally, misclassified items could be addressed simply by extending the values in one or more of our lookup tables (perhaps there is new manufacturer that we don’t have listed or a new game for a gaming system). Inserting these values into our tables allows us to build out the contextual hierarchy of our grammar and can be maintained with relatively low overhead.*

# 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.

**“I took my father out for dinner to Le Bistro on New Year’s Eve.”**

*This sentence is free from any sentiment and is merely a factual statement regarding the diner’s experience. It does set a clear value for the (main) Opinion Object:* ***Le Bistro****. It also sets the time of the review as* ***New Year’s Eve (though when the review occurred is unclear)****.*

*The review really begins in the next lines:*

**“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!”**

***-*** *Opinion Object: We can safely assume the object to be the subject of the review: the* ***restaurant*** *– but there is also a secondary object which is being reviewed – the* ***valet parking.*** *Although these two items are related, mixing sentiment amongst them could be problematic.*

*- Opinion Features: The features of the restaurant that are being assessed are the* ***décor, service, food, specials, Chardonnay collection, the menu and valet parking.***

*- Opinions: the* ***décor and service*** *both received positive opinion/sentiment – as did the* ***food, French specials and Chardonnay*** *– but the* ***valet parking*** *and the* ***rest of the menu*** *received slightly less enthusiastic opinions from one member of the party (dad thought they were too expensive).*

*- Opinion Holders: although this review seems to be written by a sole individual, the opinions of multiple individuals are contained in the text. The son has generally provided positive sentiment around his experience whereas the father expressed some negative sentiment with respect to menu pricing and parking.*

*- Opinion Time: From the context above, this opinion was expressed from the point of view of a diner that has dined at the place sometime in the past (****enjoyed*** *–* ***was expensive etc)****.*

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.

* Start off by cleaning all the review texts (stripping stop words, lemmatizing, POS/N-gram tagging etc)
* Applying a named entity recognition process (NER) – potentially a relation extraction if time allows.
* We can easily apply a lexical based approach to the model (in this case a dictionary-based approach would be very straightforward) – but we can also apply a simple supervised learning technique to diversify our analysis.
* The dictionary based approach could score words simply on their **vader lexicon** sentiment scores and then aggregate the score across each review – while not the most sophisticated technique, it could set a good baseline (or be useful in conjunction with another model).
* A more advanced system could be designed to create an auto labeled dataset. Since we have a large body of review texts (which are most likely to contain a star or numerical scale along with the raw text). We could easily create training examples where reviews with 2 stars or less are labeled as **negative** and reviews with higher scores could be labeled as **positive**. Using these review texts to train a **decision** **tree classifier** would give a different look at the sentiment in the document than that provided by the lexical/dictionary approach.
* Combining these two scores together to assess sentiment could be far more useful than either system acting independently. In fact – examining where the two methodologies diverge could lead to model tweaks that make them both better as time progresses.