CS 7337 – Natural Language Processing
Final Exam

Instructions: Clarity of answers is more important than the 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

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

  - Distributional semantics involves the study and analysis of language by using statistical and mathematical analyses of distributional parameters to quantify similarities, such as context and frequency.
  - This idea of the distributional hypothesis is that there is a correlation between meaning and distribution, so words with similar distributions have similar meanings, and the more similar words are, the more often they will occur in similar contexts or distributions. Therefore, it is possible to utilize distribution similarities to predict the meanings of words.
  - Examples:
    - River and prison are two words generally not used in conjunction with each other that also have distinct, unrelated meanings. Their distributions will be different, and the probability of them occurring in the same context is low. However, soup and stew have similar meanings, and the study of the distribution of the word "soup" in a large language sample would statistically and mathematically provide information to determine the meaning of the word "stew."

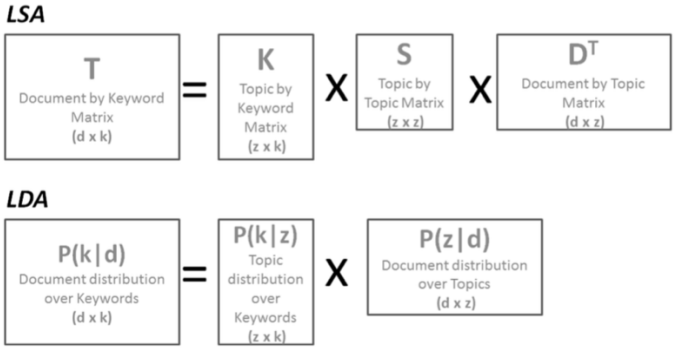
- **[5 pts] Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA) are two widely used techniques for topic modeling. Give a short overview of the two approaches and any similarities/differences between them.**

- LSA is a NLP technique in distributional semantics that analyzes similarities and relationships between language samples and words within them by creating concepts related to the document and terms. It assumes words with similar meanings will be seen in similar documents and is based on the distributional hypothesis. LSA assigns words points in a distributional space and is based on the idea that the closer words are in the space, the more similar they are in meaning. LSA begins with a large term-document matrix and then utilizes a topic-to-topic matrix to determine what maximizes the variance between documents.

- LDA is a statistical model that is used to determine topics by finding groups of words that have high co-occurrences among different documents within a corpus. Each document is seen as a combination of smaller topics, and each word's presence is a data point/attribute within the document. It is a topic model and statistically analyzes the frequency of words and phrases in certain topics. LDA can assign documents to more than one topic.

Differences and Similarities:

- LDA and LSA are both common NLP techniques for topic modeling.
- LDA utilizes probability distributions and LSA utilizes topic-topic matrixes (seen below, taken from Unit 13 slides)
- LDA is less prone to overfitting than LSA.
- LSA is an older model than LDA.
- LSA is simpler and LDA is more complex.
- LSA is based on linear algebra and LDA is based on statistical probability.
- LDA and LSA both create representations of documents and words.



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 on 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?

- Although neither would likely produce perfect results, for the question above, matching products by using product titles, utilizing Cosine similarity would be the better choice. This is due to the fact that the Jaccard similarity utilizes sets in which duplicates cannot exist therefore duplication of words does not affect the similarity score. Whereas with Cosine similarity, duplicated words within the text would impact the overall similarity score and could find similar items depending on how the product title is worded. Additionally, similarity is more important than dissimilarity, as product titles on different sites for the same item could be lengthy or brief. Whereas Cosine similarity measures the similarity between documents irrespective of their size, Jaccard measures the number of similar words in sets divided by the number of words in either or both sets.

a. 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 product

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.

- Cosine similarity score would work only in part in the above data set. A good example of this is in the first row above, in which both products have identical item numbers associated with the products so these should align. However, the titles are different, the items only have one term in common, and the section item only had two terms, causing the numerator in the equation to be 1 and the similarity score to be low. Cosine similarity works better in the second and third rows, showing more similarity between two tv models of the same brand than two completely different products. Incidentally, Jaccard would also return a low similarity score for the first row and function better on the second and third. This is seen below in several Cosine calculations utilizing multiple vectorizers (CountVectorizer, TFIDF and sBERT) as well as a Jaccard score.

	Product Title 1 (Site 1)	Product Title 2 (Site 2)	CountVectorizer Cosine Score	TFIDF Cosine Score	Sentence Transformer (sBERT) Cosine Score	Jaccard Score
0	50 Inch Class H6570G 4K Ultra HD Android Smart...	Hisense H6570G	0.158114	0.163364	0.41582918	0.045455
1	QN75Q90TAFXZA crystal 2.5" Quantum LCD	Samsung crystal UN55TU8000FXZA QLED	0.250000	0.250000	0.583774	0.125000
2	EGLF2 50 Ultra Full Motion Articulating TV Wal...	VIZIO EGLF2	0.188982	0.198145	0.3856305	0.076923

[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.

- The additional data point of the IDF score allows us to determine which words are important in the product title. This creates an additional data point that can be used to reduce the vector size by allowing higher IDF scores to be used while removing any lower IDF Scores. This can potentially increase the accuracy of the similarity score. Using weights of inverse document frequency, common terms have lower weights and uncommon ones higher, reducing the impact of common terms.
- For example, within the first row, the product's unique identifier, H6570G, has an IDF Score of 10.2, which allows us to see that that word is more important or relevant than "Android" with an IDF score of 2.6. The two product titles share only one word, causing a poor similarity score from Jaccard or Cosine, but with IDF, if the shared item is used rarely in the two samples, the numerator will increase, and the similarity score will be higher.

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)

Q3.

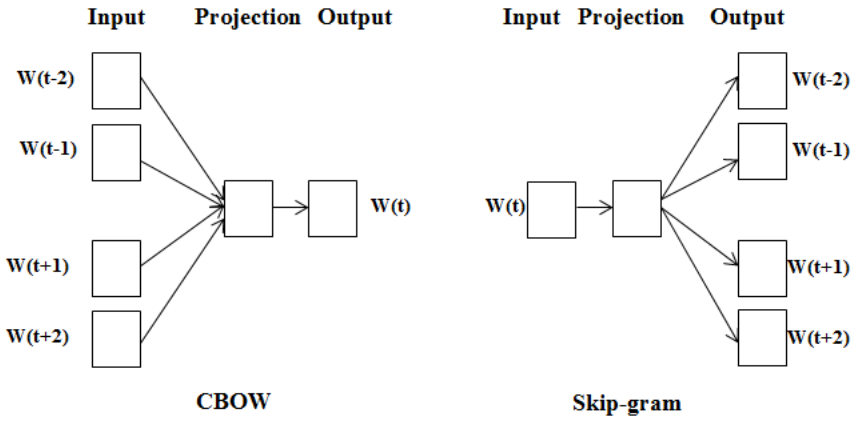
a. [10 pts] Recommender systems are a subtype of information filtering systems that help users discover new and relevant items by presenting items similar to their previous interactions or preferences. Some famous examples of recommender systems are Amazon's "Books you may like" and Netflix's "Because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

- The use of Python's Word2Vec (W2V) is extremely helpful in recommender systems as it can be used to predict nearby words within sentences with respect to the context of the sentence. W2V is a simple neural network model that has a singular hidden layer that predicts nearby context words. This concept can be used in a recommender system as words that are used closely together (i.e. sweet and sugar), or also in this example, co-purchased together, would have similar vectors that point in the same direction. Words with similar vectors that point in the same direction would be recommended to the end user, whereas words with dissimilar vectors would not be recommended to the end user.

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?

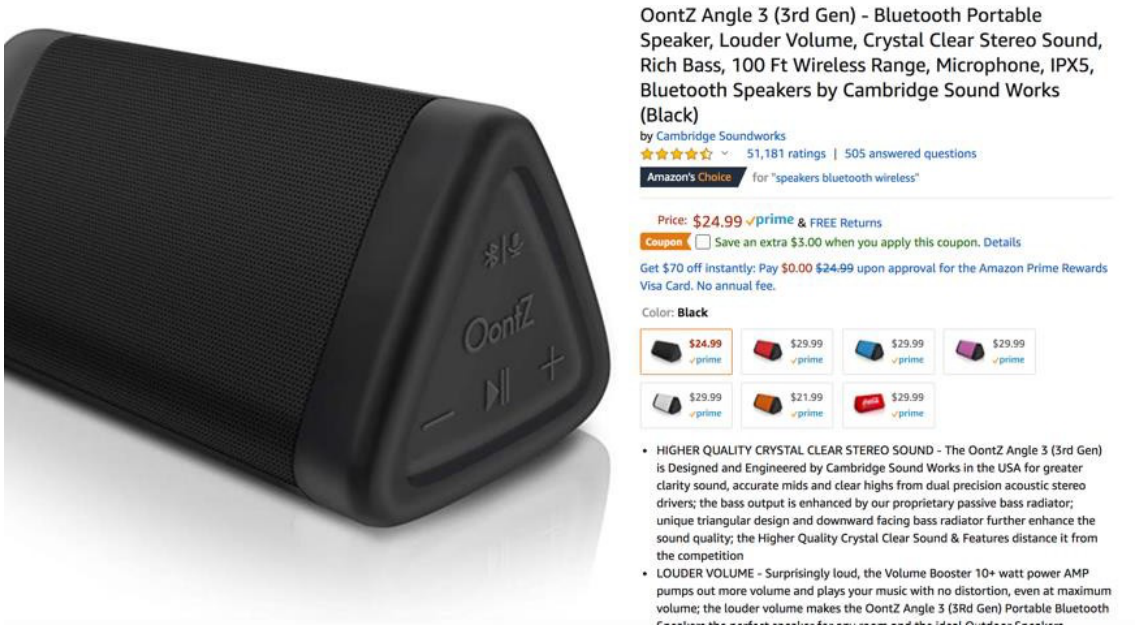
- CBOW is a neural network model that learns how to predict words close-by by using all the words inputted into the model. It utilizes the sum of background vectors to predict the target word. Skip-gram is the exact opposite of CBOW. It predicts the context of a given word. This is done by feeding the target word into the model and the output is the predicted context of the sentence. The CBOW model is more likely to overfit and is better at learning syntactic relationships. Skip-gram is less likely to overfit and is better at capturing semantic relationships.
- Model differences seen below (https://arxiv.org/pdf/1309.4168v1.pdf)



- The Skip-gram model would be chosen over the CBOW model when there is little training data and the task is oriented semantically.

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 a product title, description, and image fields. See the example below:



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

- To create the machine-learning-based classifier, I would utilize the product title and description. This would allow me to create similarity matrixes to determine which items should be classified.

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, descriptions, and images. How will this affect feature engineering and the NLP pipeline for your classifier?

- If we were only to have access to product titles, this would impact the NLP pipeline as there will be significantly less training data points that can be used. Feature engineering would be impacted as well due to the fewer amount of data points. The NLP pipeline would need to be changed to utilize different algorithms that can produce better results with less training data, or one could potentially use previously trained data from third parties (i.e. HuggingFace).

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.

- Due to the cost of obtaining training data for large-scale classification systems and having a limited budget, one could utilize pre-trained neural network models from HuggingFace or other 3rd parties. This would significantly decrease the cost and time to create a training dataset and could potentially exponentially increase the results of the model. In addition to the pre-trained model, one could add additional training data to the existing model by allowing the model to only train on new layers. This allows one to train a model at a low cost in an efficient way.
- Another option would be to manually label "enough" data points to create a good enough training data set. While this is time-consuming, it would allow one to create a classification model with low cost but high effort.
- Lastly, we could utilize the inputs of consumers as data points in our model. This significantly increases the number of data points and could help in the creation of labeled items.

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

- To handle misclassified products effectively, I would investigate where the model is misclassifying products. If the model is heavily misclassifying a certain product, there could be an issue with the feature engineering side. If the model is not heavily misclassifying a certain product, there could be more feature engineering that may need to be done to effectively decrease the misclassifications.

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 Object(s):
  - Le Bistro Restaurant
- Feature(s):
  - Décor
  - service
  - food
  - menu prices
  - valet parking,
  - service
  - special occasion
- Opinion(s):
  - Menu Prices are high
  - décor and service are fantastic
  - food is enjoyable (French countryside special and chardonnay)
  - menu prices are high
  - valet parking is expensive
  - recommended the location for special occasions
- Opinion Holder(s):
  - Author of review
  - father of author
- Opinion Time:
  - New Years 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 <u>Positive</u> or <u>Negative</u> to reviews.

- In creating a sentiment analysis system for restaurant views, we will need to begin with a dataset. These reviews could come from several different places such as Google, Twitter or Yelp. Once the data is loaded in and preprocessed utilizing the Python libraries NLTK or SpaCy. Within the preprocessing step, we can feature engineer a sentiment score by utilizing the review score (1-3 ratings with be negative while 8-10 ratings will be positive). This feature engineering will help in our training for our model. We will then need to vectorize our reviews using a vectorizer such as tf-idf. After the preprocessing and vectorization is complete, we can split our data into training and test datasets to pass into our model. Since word

sentiment can have multiple levels such as love can include adoration, affection, caring, or attraction, we must determine the best hierarchy of emotions to utilize in our nlp pipeline. Shaver's Hierarchy of emotions is commonly used in sentiment analysis and will need to be included in our machine-learning pipeline. Finally, our model needs to be chosen. The python package SpaCy has a module, SpacyTextBlob, that will output a sentiment score that can be categorized into positive or negative reviews. Any review that returns a sentiment score higher than 0 will be considered a positive review and any score of 0 or below will be considered negative. Further adjustments to this cutoff will need to be made once the

- In short, the process is as follows:
    - Obtain data (Twitter, Google, Yelp)
    - Preprocess (Spacy's Tokenizer)
    - Feature engineer (bucket negative and positive based on review score)
    - Vectorize reviews (sklearn's TfidfVectorizer)
    - Split into test/train data sets (sklearn's train_test_split)
    - Add hierarchy of emotions to nlp pipeline (Shaver's Hierarchy of Emotions)
    - Model (SpaCy's SpacyTextBlob polarity score)
    - Bucket model results based on score