

Final Exam CS 7337 NLP

Paritosh Rai



CS 7337 – Natural Language Processing



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 Distribution Hypothesis come with slogan:

*“You shall know the word by the company it keeps”*

* *J. R. Fifth*

The idea of the *Distributional Hypothesis* is the distribution of words in a text holds a relationship with their corresponding meanings. The more semantically similar two words are, the more they will tend to show up in similar contexts and with similar distributions. Stating it other way round two phrases with different semantical meaning, their distribution is likely to be different.

Example: Computer and dog are two words unrelated in their meaning, and in fact they are not often used in the same sentence or will not have similar kind of word distribution. On the other hand, the words dog and cat do not have similar meaning, but they do carry some kind of similarity (both are domestic pets) so we will see the sometimes together and will be seen around similar kind of words so they may share some aspect of meaning.

This hypothesis is often stated in terms like “words which are similar in meaning occur in similar contexts” (Rubenstein & Goodenough, 1965);

* “words with similar meanings will occur with similar neighbors if enough text material is available” (Sch¨utze & Pedersen, 1995);
* “a representation that captures much of how words are used in natural context will capture much of what we mean by meaning” (Landauer & Dumais, 1997);
* “words that occur in the same contexts tend to have similar meanings” (Pantel, 2005), just to quote a few representative examples.

*Ref: http://www.diva-portal.org/smash/get/diva2:1041938/FULLTEXT01.pdf*

Few Examples: Tea and Coffee

|  |  |
| --- | --- |
| I drink tea every day. | I drink coffee every day. |
| Who is making the tea? | Who is making the coffee? |
| I like both hot and cold tea. | I like both hot and cold coffee. |

Form the above example we can construed that tea and coffee are very similar words.

The general idea behind the distributional hypothesis is there is a correlation between distributional similarity and meaning similarity, which allows us to utilize the former (distribution similarity) to estimate the latter (meaning similarity).

**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.**

*Ref: http://ceur-ws.org/Vol-1815/paper4.pdf*

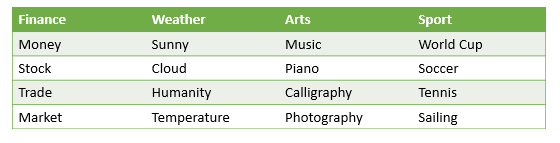
**Latent Semantic Analysis (LSA):**

Latent Semantic Analysis is one of the natural language processing techniques for analysis of semantics, help to extract meaning of a corpus of text with the help of statistical analysis of words in the document/corpus. LSA builds the matrix that contains documents across the rows and columns that records the number of times a particular word is included in the documents. It leverages TF-IDF as value for the matrix. LAS leverages the TF/IDF (Term Frequency/ Inverse Document Frequency) counting the occurrence of each word and rare words to provide them weights based on their rarity. The TF-IDF text analysis technique helps converting the documents into vectors where each value in the vector corresponds to the TF-IDF score of a word in the document. Each word has its own axis, the cosine similarity then determines how similar the documents are.

Followed by dimensionality reduction to higher dimension to reduce noise and sparcing to make information manageable using techniques like SVD (Singular Value Decomposition). This results in the two matrices – the document-topic matrix and the topic term matrix. LSA uses these vectors to give output of document similarities in the form of a cosine similarity matrix. Values range from -1 to 1, where -1 represents two documents that are complete opposites, and 1 represents two of the same documents.

**Latent Dirichlet Allocation (LDA):**

LDA is a method of data mining to manage large documents archives (corpus). It can summarize a corpus, classify/ categorize the documents, identify the common theme, and enable readers or search engines to find relevant article. It is also referred as topic modeling. The algorithm leverages statistical/ probability distribution methods that analyzes the words of the original texts to discover the themes that run through them, how they connect and change over time. LDA is a generative probabilistic model, that assumes a Dirichlet prior over the latent topics.



LDA review the topic and removed certain words that are not adding value to analysis. Certain words like a, the, with, can may not contribute much to topic so need to be removed from the modeling (remove stop words). Build a multi nominal distribution linking words in the document with topics.

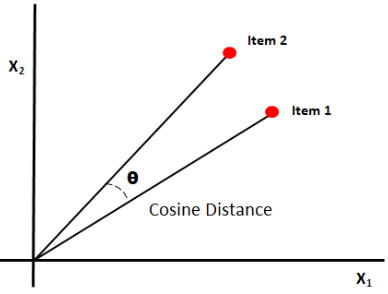
The output will be in the form of a topic matrix, with each row representing the probability distribution defined over the topics for each document. The user specifies the total number of topics that the words are sorted into, and each value in the matrix ranges between 0 and the user-defined number of topics.

|  |  |
| --- | --- |
| Similarity | |
| LSA | LDA |
| Both use Bag-of-words as input matrix | |
| Both are unsupervised learning | |
| Differences | |
| LSA | LDA |
| Latent Semantic Analysis | Latent Dirichlet Allocation |
| Useful for **small scale** | **Large Scale** data mining |
| LSA focus on **reducing matrix** dimension | LDA solves **topic modeling** problems |
| **Number of times** word included in doc | Characterized by a **distribution** of words |
| Uses cosine similarity matrix | Outputs a matrix |
| Values range: -1 to 1 | Values range: Between 0 and the user-defined number of topics |
| LSA was introduced in 2005 by Jerome Bellegarde. | LDA was introduced in 2003 by David Blei, Andrew Ng, and Michael I. Jordan |

**Q2.**

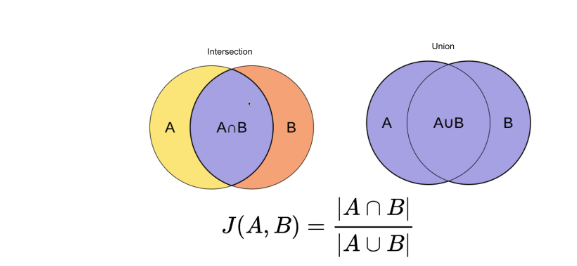
**a. [5 pts] You are a Data Scientist for an e-commerce site for electronics which also supports 3rd party sellers. You would like to build a system to find and match the same products that sellers on your website sell so that you can present them in a single product page. You decide to use** **product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

**Cosine Similarity:**

The cosine similarity calculates the cosine of the angle between two the vectors. It is used to determine how similar documents are to each other. The TF-IDF text analysis technique helps converting the documents into vectors. Each word has its own axis, the cosine similarity then determines how similar the documents are. The cosine similarity can take on values between -1 and +1. If the vectors point in the exact same direction, the cosine similarity is +1. If the vectors point in opposite directions, the cosine similarity is -1.

Use case: Cosine similarity is good for cases where duplication does matter.

**Jaccard Similarity:**

Cosine similarity is for comparing two real-valued vectors, and Jaccard similarity is for comparing two binary vectors (sets).

Jaccard similarity divides the size of the intersection by the size of the union of the sample sets. Higher the value of intersection higher is the similarity in text or document.

Use case: Jaccard similarity is good for cases where duplication does not matter

Ref: https://towardsdatascience.com/calculate-similarity-the-most-relevant-metrics-in-a-nutshell-9a43564f533e

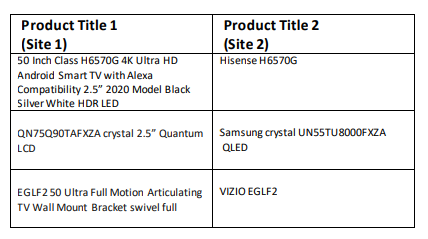
**Recommendation:**

Both Cosine similarity and Jaccard similarity are common metrics for calculating text similarity. Calculating the Jaccard similarity is computationally more expensive as it matches all the terms of one document to another document where are cosine similarity takes whole sentence vector (so computationally less expensive).

As we know titles of products are short phrases, not full sentence.

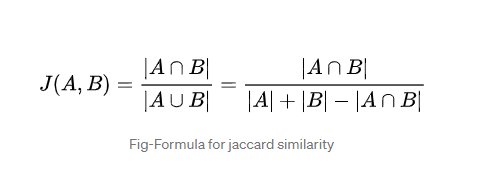
We will be using only product titles to compute product similarity, it should not create too much computational load and duplication is not a big concern while comparing product titles, so will recommend using Jaccard Similarity to do word by word comparison.

**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.**

Ref: https://medium.com/analytics-vidhya/introduction-to-similarity-metrics-a882361c9be4



**Jaccard Similarity Calculation for Three Rows:**

|  |  |  |
| --- | --- | --- |
| **Row1** | **Row2** | **Row3** |
| A = 21  B = 2  A ꓵ B = 1  AꓴB = |A|+|B|-|A-B|=21+2-1 =22  J (A, B) = 1/ (21+2-1) = 1/22 = 0.04545 | A= 5  B = 4  A ꓵ B = 1  AꓴB = |A|+|B|-|A-B|= 4+5-1 =8  J (A, B) = 1/ (5+4-1) = 1/8 = 0.125 | A= 12  B = 2  A ꓵ B = 1  AꓴB = |A|+|B|-|A-B|= 12+2-1 =13  J (A, B) = 1/ (12+2-1) = 1/13 = 0.0769 |
| **0.04545** | **0.125** | **0.0769** |

**Comparison of Three rows using Jaccard Similarity**

|  |  |  |
| --- | --- | --- |
| **Product Title 1 (Site 1)** | **Product Title 2 (Site 2)** | **Jaccard Similarity** |
| 50 Inch Class H6570G 4K Ultra HD Android Smart TV with Alexa Compatibility 2.5” 2020 Model Black Silver White HDR LED | Hisense H6570G | **0.04545** |
| QN75Q90TAFXZA crystal 2.5” Quantum LCD | Samsung crystal UN55TU8000FXZA  QLED | **0.125** |
| E GLF2 50 Ultra Full Motion Articulating TV Wall Mount Bracket swivel full | VIZIO EGLF2 | **0.0769** |

My recommendation in 2a is to use Jaccard similarity matrix. **However, the recommended similarity approach (Jaccard Similarity) in 2a is not working well with the given data set.**

Comparing Row 1 and Row 2: Row1 that have similar product (same TV model H6570G) and Row 2 has different product with same brand (Samsung). Calculation above shows lower Jaccard Similarity Score for Row 1. Whereas expectation was that we should have high Jaccard Similarity Score for Row 1 as both sites are carrying same model (H6570G) compared to row 2 having different product with same brand name.

Comparing row 1 and Row 3: Row 1 has similar product (same TV model H6570G) and Row 3 has totally different product (Wall mount TV stand vs TV) but score of different product in Row 3 is higher than Row 1. Whereas expectation was Row1 should higher score.

Comparing Row 2 and Row 3: Row 2 has similar brand (Samsung) and Row 3 has totally different product (Wall mount TV stand vs TV). In this case expectation aligns with calculated score that Row2 should and is higher than Row 3.

Comparing Row 1 , Row and Row3: Based on description Row1 should have highest score followed by Row 2 and then Row 3. Were as calculated Jaccard Similarity Score Row2 is highest, followed by Row3 and then Row1.

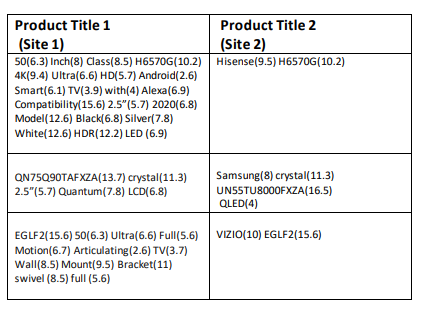
**Conclusion:**

Although we got alignment with Jaccard Similarity Score in one case, but overall Jaccard Similarity Score did not do a good job with the given dataset. However, I stay align with my recommendation to use Jaccard Similarity as Jaccard Similarity are best to use in scenarios where duplication does not matter.

It is critical with any data driven matrix how solid is input data. In the cases discussed above there is challenge with how the title is defined. It is important to have right title definitions to capture accurate characteristic of the product. This task becomes more challenging when data is coming from multiple sources as every source have uses different criterial to define the title of the product, some focused-on function of product, same just pot part number or simply use model name and some are application base or based on what customer will search. Like in this case we are comparing products at two different site and site owners have put in different diligence and criteria to define the title of the products.

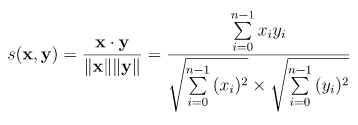
While pulling the data in from other sites we should do audit and alignment to ensure data search and categorization can take place appropriately.

**[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**.



**Cosine Similarity:**

The cosine similarity can be calculated using IDF score. The cosine of the angle between two vectors. Following is the formula to calculate cosine similarity:



Following is the calculation for Cosine Similarity for each row

**Cosine Similarity Calculation for Row 1:**



**Cosine Similarity Calculation for Row 2:**



**Cosine Similarity Calculation for Row 3:**



Cosine similarity has a similar issue as Jaccard Similarity Score. The Cosine Similarity approach as well does not align with product similarity. Row one shows the lowest score (supposed to have the highest similarity), and row 3 offers the highest score (having the lowest similarity as they are the different products).

Row1: Is set of two similar products and has lowest Cosine Similarity score.

Row2: Is same brand but different TV model. Has medium Cosine Similarity score.

Row3: Have different product (TV vs. TV stand) but has highest Cosine Similarity score.

**Conclusion:**

As quality of input data remains an essential factor. Garbage in garbage out. Misalignment in input data can lead to incorrect correlation and similarity. Before doing any analysis, it is critical to do data alignment, especially when data is coming from different sources. For example, the model number or type of equipment (TV vs. TV stand) or vendor may be a higher weightage. The following three approaches can be used to merge the data from different sources:

1. Perform a data profiling activity: Determining your data's accuracy, completeness, and validity.
2. Fix data quality issues after the profiling process: Correct spelling differences, nicknames, abbreviations, inconsistent format, missing numbers.
3. Perform a final data profiling check: Revalidate the process.
4. Start building something like data indexing (similar to Amazon) so some consistency can be derived. Indexing is storing the products in a database with attributes as keys

As observed, using either Jaccard Similarity Score or Cosine Similarity Score does not align with actual data. Leveraging a suitable method helps. However, the outcome is as good as the input data. My recommendation to use Jaccard Similarity still stay strong. However, it recommended to carry out data profiling before using data for any kind of analysis.

**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?**

Word2Vec or W2V is a group of word embedding algorithms that provides state-of-the-art results on NLP. They are based on the Distributional Hypothesis, which states that words in the same contexts tend to suggest similar meanings. Words that are similar to each other have similar vector representations. The mathematical function (cosine similarity) can be calculated using vectors to determine the semantic similarity between the words.

We will be following three big steps process to build Word2Vec food recommendation system:

Clean up purchased food item data

Text Processing using W2V

Possible food items based on similar purchases

Possible purchases based on individual choices

Normalize food items to build vector space

There are multiple ways to quantify the distance between them. They take as input a corpus of text and spans a vectorial space, where each word is mapped into a vector. Words that more often appear in similar contexts are mapped into vectors separated by shorter Euclidean distances. Including Word2Vec there are multiple tools like “*Doc2Vec”* and “*FastText”* can support this type of effort. This made possible to capture their similarities.

However, here we will leverage Word2Vec to encode the food items, from the datasets and convert them to vectors. While word2vec is powerful and super easy to understand, it also has its flaws. If a word isn’t present in our vocabulary, the model won’t be able to represent its vector. There is no cross-language support.

Once we have this vector space, we can identify recommendations in two different ways:

1. For a given food item, we can find other similar food items users are likely to purchase, by calculating the vector similarity between the specific food item and other items and identifying the top n items with highest similarity. A customer who buys fountain coffee has a significant likelihood of getting donuts.

2. We can map individual user’s food purchasing habits on to this vector space by calculating the average of vectors of all the items purchased by the user. Once we map the user to this space, we can follow the same approach as above to identify the similar food items the user may like and provide them as recommendations.

Following are eight (8) step processes to create Word2Vec to recommend items to users who may have bought or show interest in one of the items:

Step 1 Data Treatment: Secure the data on co-purchases for food items (as we are startup, we may have to purchase the data from market later we can leverage our own data to do the same). Do necessary clean up and preparation to the data ready for text processing using W2V. Extract the buying history and sequence of purchase.

Step 2 Split Data Traing and Validation: Split the data into 90 -10. It is excellent practice to set a small part of the data for validation purposes. 90% of data will be used to create Word2Vec embedding.

Step 3 Create Token-Embedding Mapping: Create a sequence of purchases in both test and training data.

Step 4 Quantify Text Data to Vector: Leveraging the pre-trained Word2Vec embeddings model performs lexical analysis and generates words as food data vectors.

Step 5 Organize the Quantified Data: With vectorized words generated from running the model on our food data, we can create text classification to perform numerical comparisons of items. Closer the word shorter the distance.

Step 6 Summing and Normalizing the Vector: As food items are composed of multiple words, sum the vector for each word and normalize to get a vector for the item (for example, cheeseburger => cheese + Burger). This may help grouping similar items in vector space and running our model on them.

Step 7 Make Predication: Leveraging co-purchase food items to predict the next possible purchase. The model will leverage user ID and product ID to predict potential next purchase. The model will take the average of all the vectors of the products that user has bought (so far) and find similar products. Train model with 90% of data and create a prediction model.

Continue building the user ID data with frequent purchases. For example, what specific brand of coffee user drinks or coffee with sugar or without sugar.

Step 8 Check Performance: Validate the co-purchase model with 10% validation data. If data is not within the permissible limit, try a different prediction model, adjust the window size, adjust hyperparameters, and retune the model.

Step 9 Execute: If validation set accuracy is in acceptable rollout the User ID and Product ID model and continue building your own database.



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.**

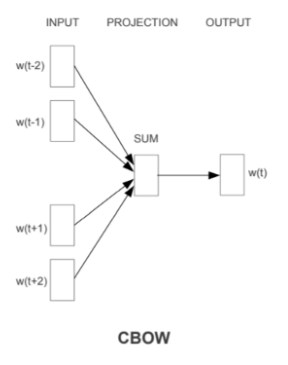
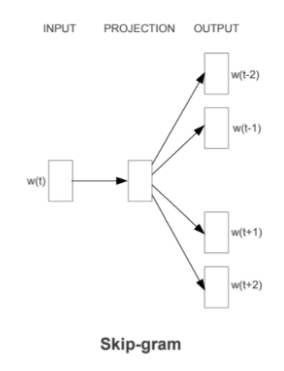
**Word2Vec or W2V** is a technique or algorithm using a two-layer neural network model to learn associations of words in Natural Language Processing (NLP). It is a technique to convert text into numbers. Word2vec represents each distinct word with a particular list of numbers called a vector. The vectors are chosen carefully such that a simple mathematical function (the cosine similarity between the vectors) indicates the level of semantic similarity between the words represented by those vectors. The usefulness of W2V is to group the vectors of similar words. It detects the similarity of words mathematically without human intervention. Word2Vector carries out vector similarity analysis in two ways:

1. **CBOW (Continuous Bag of Words):** Using context to predict the target word. To achieve this, the model calculates the average of context vectors. This average is forward propagated to learn the prediction by maximizing the conditional probability of the actual target word.

CBOW is several times faster to train than skip-gram, slightly better accuracy for frequent words.

1. **Skip-gram:** Using a word to predict the target context. Skip-gram is the exact opposite of the CBOW model. Skip-gram starts with the input word vector and predicts the n-neighboring words. Focus word is the single input vector and the target context words are the output layer. Leveraging backpropagate of errors, the neural network model fine-tunes the weights to minimize the total prediction errors across all context words.

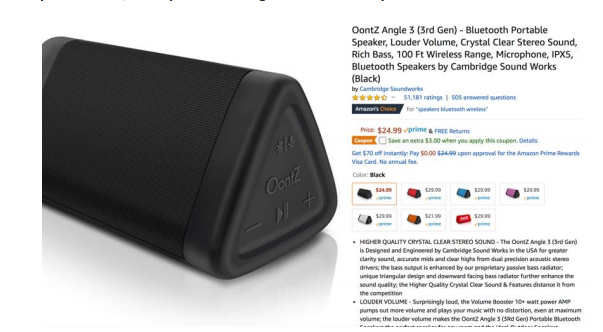
Skip-gram works well with small amount of training data, is able to represent well rare words and phrases well.

The COB and Skip-gram are mirror version of each other. COB predicts single word Skip-gram predict multiple words.

|  |  |  |
| --- | --- | --- |
|  | **CBOW** | **Skip-gram** |
| **Defination** | Predict the words leveraging neighboring words in pre-defined window size. | Predict several context words from single word |
| **Over fitting** | CBOW is prone to over fitting as it appears multiple times with same context | Less sensitive to over fitting as it relies on one word |
| **Data Size** | Need bigger training set | Works well with small amount of data |
| **Rare Words** | May not be very effective with rare words | Represent well with rare words |
| **Frequent Words** | Slightly better Accuracy | Less accurate than CBOW |
| **Training Speed** | Faster to train than skip-gram | Takes longer to train than CBOW. As it uses pair wise training sample. |
| **Output** | One word | Multiple words (context) |
| **Input** | Context | word |
| **Propagation** | Average Forward Propagation | Backward Error Propagation |
| **When to use** | Use in scenario where we need faster training from data (than the skip-gram), looking for slightly better accuracy and data set have frequent words. | Better for scenarios with a small amount of the training data. Data set contains, rare words or phrases (works better with rare words). Skip-gram is heavily used in recommendation solutions. |

**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?**

We will use all three product details to classify the result, i.e., Product Title, Product Description, and Image Field. With more fields, we will be able to build better machine learning classification. As we know, incoming streams may not maintain similar and consistent formats. Some incoming streams will have excellent descriptions or with very brief Titles. Others may have very elaborate Titles others may help with great images. So, leveraging all three information will support each other and enable us to build better alignment among the products. Say if incoming stream does not have a good titles and data description and images may help the system to put in suitable classification. Effectiveness of combining image and text (tiles and description) to predict and gain insight into the classification with enhance multiple folds.

Associate each text (title and description of product) with an image, and then convert the image and the text into two feature vectors using a Convolution Neural Network (CNN) and a Recurrent Neural Network (RNN) respectively; the weighted combination of the two feature vectors can enhance the classification. To enhance the RNN’s performance implement a Convolution-RNN (C-RNN) structure that applies convolution layers on top of RNN’s attention signal. The final prediction result using the combined model significantly outperformed the SVM and linear regression baseline.

Leveraging Neural system, makes it more resilient to noise, easy to scale and can be done with minimal domain knowledge. Although may require lot of query time and data.

Ref: http://cs231n.stanford.edu/reports/2017/pdfs/816.pdf

**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?**

Ref: https://www.csie.ntu.edu.tw/~cjlin/papers/title.pdf

Titles are different from texts in several aspects, such as the length of each instance (most titles are concise), the distribution of lengths (most product titles have similar lengths), and the grammatical structure (most titles are incomplete sentences). A product title classifier may need to be designed differently, however, it has go through parsing, indexing and matching.

Existing systems for text classification, like NLTK, uses the bag-of-words approach so that each feature corresponds to one or several words (word to context). Text data is preprocessed by unify words in various morphological forms (stemming) and retain useful lexicons (stop-word removal), followed by a conversion from texts to feature vectors (e.g., TF-IDF model). Then classification methods like linear SVM or Naive Bayes are applied on a labeled corpus to obtain a classifier for categorizing new texts.

Conventional text classifications still apply to product title classification in classifying products leveraging product titles. For example, data normalization is essential for faster SVM training. However, we may have to do few things differently due to the difference in nature of product titles (short length and incomplete sentences):

1) Stemming and stop word removal are not very helpful. This may lead to eliminating some critical information.

2) As the titles are short, bigram, or degree -2 polynomial mapping is more effective. As in the example in question Bluetooth Speaker to treated as one feature vs. two features.

3) As title lengths are similar, data normalization speeds up SVM training. Product classification is generally multi-class, so SVM performs better.

4) As product titles are different, the performance of binary and TG-IDF representations are similar.

As in this case, the NLP pipeline will not include processes to leverage images and extract information from product descriptions to enhance product classification. We may have to account for additional procedures to improve classification accuracy by getting help from SMEs or industry experts, leveraging other databases, or tracing customer search, clicking, and purchase behavior to enhance classification. Using a combination of learning, hand-crafted rules, crowdsourcing, and in-house analysts extensively will improve classifier solution performance and cost-effectiveness.

**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.**

Correct classification of products is a very critical activity in an electronic store. The right amount of attention and focus, and diligence is vital for the success of the business. It will be very challenging to build a large-scale classification system that can create multiple classes that are mutually exclusive and multi label with a limited budget and resources. As discussed above to build an efficient and cost-effective classifier it is recommend using combination of multiple solution. Following are few recommendations to obtain training data for classification:

**Apply 80/20 Rule:** In any store, certain times move fast than others, say 80% of transactions are on 20% of items. We can deploy the limited number of resources and get a robust and enhanced data set for the 20% of correctly classified items. Put additional diligence, funding, and audit to ensure this 20% of the transactions are perfect. For the remaining 20% transaction, leverage a few other methods below to make the best effort.

**Outsource / Offshore:** This will be a hardcore manual effort. Outsource this manual effort to an Offshore company. Offshoring the work will undoubtedly help with budget limitations. There are many products, and we will be doing this with manual brute force effort. It will be a very daunting task. Human intervention increases the possibility that human errors, skill variance, and cultural thought processes may result in incorrect classification. For example, certain products may be considered a luxury in developing countries vs. seen as economic products in developed nations. However, putting an additional set of eyes will undoubtedly be better. Offshore outsourcing may help with new evolving product lines as well.

**Apply Rule-based Classifier:** Define specific rules to add products to training data set list made after reviewing a small sample of data. May be using model number or occurrence of certain words in Title. We have seen in the above example (Question 2a) that this technique is not give very reliable outcome. However, right training data may help classification discussed in question 2a.

**Crowdsourcing:** Share the challenge with the social network. Engagement of broader audience to trigger Self Organized crowds. Post the challenge on the internet, recruit the crowd to work on the challenge, and allow the crowd to organize in teams. Teams compete to provide the best answer. The winning team is compensated. Although, crowdsourcing cannot easily scale and is challenging to manage quality control issues. This challenge can be overcome by collaboration crowdsourcing with the internal data analytics team. We implement the best idea and will get many great ideas in the Self Organized crowds’ competition. Leveraging the best idea and hybrid with other ideas, we may take the effort to a new level.

**Web Crawling:** Look at a similar website (like Amazon or best buy) and see if we can find where we can do data crawling and create a database from other websites in a similar business to create a database.

**Request data from Manufacturers and Vendors:** Work with Equipment manufactures and vendors to get the list and discretion of their databases. They will release their items list very quickly as it may help them push their sales. The manufacturer's list will be an electronic form of data, and levering some of the NLP techniques, we can build a robust classification algorithm. Comparing the incoming titles with vendor Titles may help us with better classification and larger and reliable training data.

**Extracting Data from Past Research:** Look at sites like Kaggle if they have done some research or analysis. Combine multiple such inputs to create the electronic items corpus. Using the NLP technique, create a classification.

**Build Basic Structure and Correct as we go**: Create initial structure. Based on customer feedback and movement, clicking of curser and purchase behavior keeps adding to data base and correcting the classification in existing set to make it more accurate.

Accuracy is not free, and data is not free either. So, if we try something cheaper, there will be a trade-off. To stay frugal, we must get innovative and creative. We should leverage the combination of multiple methods discussed above to enhance our training data. The more extensive, more stable, and solid our training data is, we can do a better job crate robust training data that will enable us to minimize the errors or miss classifications. This will be continuous process to keep fixing the errors and keep adding new product lines to training data set to keep it up to date.

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

It is very challenging to get a perfect multi class, mutually exclusive classification. However, we must do continuous diligence to find and fix it. Every wrong classification should be seen as a lost sale opportunity. Misclassification is a two-step process:

* Finding misclassified item
* Fixing the misclassified item

We will be following steps to manage the misclassified items:

**Look at search and clicks on the website:** Generally, online purchases start with searching using search engines or respective websites (like Amazon). Use NLP techniques to find similarity (cosign similarity) between search and final purchase. If they match, then it is a correct classification for that item, and if they have a low similarity score, it may require some investigation. If multiple attempts for the same item lead to a different purchase, this may need a deeper dive.

**Unique Search:** Look at unique searches and see if they align with the proper outcomes or led to closer on purchases. This may not be the case of misclassification; it might be customers looking for something different and unique. That is an excellent opportunity to in cash. Look final purchase and link to the right classification. For example, pineapple search in electronic store. Maybe a customer is looking for speakers of the shape of a pineapple. If we have that product, let us ensure sufficient details are added to the title or description to manage these unique questions.

**Learn from Misclassification:** If we find Samsung TV stand is miss classified as Samsung TV. Review the stands of other brands of TV. Identify if there are similar issues. Look into classification mythology used to classify the specific item. If it was similarity-based, add additional words to the description and title. If it is rule-based, review the rules, and incorporate necessary adjustment(s) to the ML/classification algorithm. Also, ensure incoming streams are modified with additional descriptions (whether at our end or income stream end) in the title or product description to ensure we learn from one mistake and fix others.

**Regular Audit:** Carry out regular audits of small random samples to ensure the correct items are picked up. Take a complete list of the items streaming to the store. Randomly pick the items to validate the classification. We can build a schedule-based script that keeps picking random samples from the sample and validates the accuracy and confusion matrix to identify the errors. Based on the type of error, make the necessary adjustments not limited to rule fixing and script adjustment.

**Human Error:** Build a classification matrix that is strong enough to identify the spelling error, fix the spelling error or typos, and direct the user to the correct class. We can leverage extensive text corpus to find similar words and adjust the words in the search algorithm.

**Misspell and Typo in Product Title and Description:** Using the right size of windows is critical when evaluating similarity, so minor typo and spelling word misses can be overcome when looking at CBOW (Continuous Bag of Words). Compare the title and description word with a large word corpus to identify typos and spelling mistakes and fix them.

**New Items Added:** New items will continuously be added to the portfolio. Build a script to pick newly added items regularly. Validate their search outcome. Validate the description, title, and rules of ML to ensure they will get correctly classified.

**Corner Cases:** Look out for corner Cases. Ruled-based classification may not do a great job putting them in the right bucket. Have a manual workaround in place to manage these types of cases.

**Summarize:** No one system or method will minimize the incorrect classification. The hybrid approach will deliver solid synergies that will help identify and fix the gaps quickly. There has to be a contiguous process to keep detecting them and correcting them. Bothe learning (ML) and hand-crafted rules are critical and need to work hand in hand. Crowdsourcing is a very critical tool but needs to be under the microscope. Couple crowdsourcing with in-house analytics and developers to enhance the outcome. Combo of Human and machine is very effective and provides multi-fold synergy gains.

**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: *Le Bistro***

**Opinion Features: décor, service, food, drinks, price, valet parking, overall experience**

**Opinions: fantastic, enjoyed, a bit on the high side, expensive, definitely recommend**

**Opinion Holders:**  **reviewer, his/her father**

**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.**

There are three commonly used major sentiment analysis models. The model selection will depend on the amount of data to process and desired accuracy by business:

**Rule or Lexicon Based Approach:** This approach uses manually crafted rules for data classification to determine the sentiment. In this approach, words from the dictionary were used with positive or negative values to denote their polarity and sentiment strength to calculate a score. More innovative rules can be developed to identify context-based sentiments. It counts positive and negative, and positive sentiment words. If the number of positive sentiments is more than negative, it will return the positive sentiment. If both are equal, it will release neutral sentiments.

**Automated or Machine Learning Approach:** In this approach, a machine learning algorithm uses supervision learning. An algorithm is trained with training samples until it can predict with accuracy the sentiment of the text. Then large pieces of text (test set) are fed into the classifier, predicting the sentiment as negative, neutral, or positive. Naïve Bayes, Logistic Regression, and Support Vector Machines (SVM) are widely used for large-scale sentiment analysis because they are capable of scalability.

**Hybrid Approach:** Hybrid sentiment analysis models are the most modern, efficient, and widely used approach for sentiment analysis. This can get the benefits of both automatic and rule-based systems. Hybrid models can offer the power of machine learning coupled with the flexibility of customization.

In general lexicon-based method may work well with a good lexicon dictionary. However, lexicon-based method may not adapt to the evolving language as seen in social media platforms like Twitter, Facebook, and Instagram. Opting for a hybrid approach with a combination of lexicon or rule-based approach and machine learning approach may reveal the best results in these kinds of scenarios.

Following is the outline to design a sentiment analysis system using the lexicon-based approach:

**Step 1:** Download the data

**Step 2:** Remove the stop words. Do not remove punctuation marks. This help identify some sentiments (for example “!”)

**Step 3:** Tokenize the text.

**Step 4:** identify the POS (Part of Speech) tag for each of the tokens leveraging words from words from the dictionary.

**Step 5:** Find the sentiment score or polarity score using sentiment analyzer. (Positive, negative, or neutral)

Example: SentimentIntensityAnalyzer gives a dictionary of score in four categories: Negative, Neutral, Positive, and Compound score. Use compound score to classify the sentiments:

Compound score > 0 is positive sentiment

Compound score < 0 is negative sentiment

Compound score = 0 is neutral sentiment

**Step 6:** Run step 5 for all the tokens in give review.

**Step 7:** Aggregate the positive and negative polarity scores for all the tokens to obtain overall sentiment score of each review (total of positive polarity scores – total of negative polarity scores)

**Step 8.** If the overall sentiment score is positive, assign the sentiment as positive; if it is negative, assign the sentiment as negative for review.

**Step 9:** Repeat Step 5 to 8 for each review and identify sentiment for each review.

