Faiza Abdullah

Lab 04

ITAI 2373 Natural Language Processing (NLP)

Professor: Patricia Mcmanus

Reflective Journal: Lab 04 - Text Representation

Key Insights on Text Representation and Its Importance

This lab was an eye-opening exploration into the world of text representation, revealing how raw text is transformed into numerical formats that machines can process. One of the most significant insights was understanding why text must be converted into numbers for machine learning. Human language is rich and contextual, but algorithms require structured, numerical inputs to perform computations like similarity analysis or classification. Learning to implement techniques like Bag of Words (BOW), TF-IDF, n-grams, and word embeddings from scratch clarified how these methods bridge the gap between human language and machine understanding. The lab emphasized that each representation method has unique strengths—BOW for simplicity, TF-IDF for weighting important terms, n-grams for capturing context, and embeddings for semantic relationships—making them suited to different tasks.

Another key takeaway was the trade-off between sparse and dense representations. Sparse methods like BOW and TF-IDF are interpretable but lose word order and context, while dense embeddings capture semantic relationships but are computationally intensive and less interpretable. The ability of word embeddings to perform operations like "king - man + woman ≈ queen" was particularly fascinating, as it showed how vector spaces encode abstract relationships, revolutionizing NLP applications like chatbots and translation systems. This lab deepened my appreciation for the

strategic choices involved in selecting a representation method based on task requirements, data size, and computational constraints.

Challenges Encountered

The lab presented several technical challenges that tested my problem-solving skills. In Step 4 (N-gram Analysis), I encountered an error due to a missing NLTK resource, punkt_tab. The notebook initially used punkt, but tokenization failed until I added nltk.download('punkt_tab') in the setup cell. This issue, similar to one I faced in Lab 02, reinforced the importance of verifying resource dependencies in NLP libraries. Debugging this taught me to anticipate and resolve setup issues proactively, ensuring smooth execution of subsequent steps.

In Part 5 (Classification System), I faced an error in the classification report visualization because the test set occasionally contained only one class (positive or negative) due to the small dataset subset. This led to an incomplete classification report, as the classification_report function expected both classes. The resolution involved modifying the dataset loading cell to use the entire movie reviews dataset instead of a subset, ensuring both classes were present in the test set. This challenge highlighted the importance of data balance in machine learning and the need to validate dataset splits before evaluation.

Additionally, Exercise 6 (Feature Analysis) threw an IndexError because the tfidf_classifier was trained on data with only one class, causing the feature_log_prob_ attribute to have insufficient entries. Fixing this required adding a check in the analyze_important_features_solution function to return empty lists if the classifier lacked data for both classes. This experience underscored the importance of handling edge cases in machine learning pipelines and deepened my understanding of classifier behavior with imbalanced data.

Conceptually, I initially struggled with the intuition behind TF-IDF's inverse document frequency (IDF). Calculating it from scratch in Exercise 3 clarified how it prioritizes rare, informative words over common ones, but I had to revisit the math (e.g., log(n_docs / (docs_containing_word + 1))) to grasp its impact on document similarity. These challenges, while frustrating, were invaluable in building my technical and analytical skills.

Connections to Real-World Applications

The lab's focus on movie reviews made it easy to connect text representation to real-world applications. For instance, **BOW and TF-IDF** are directly applicable to spam detection systems, like those used by email providers. Words like "free" or "win" have high TF-IDF scores in spam emails, enabling classifiers to flag them accurately. This resonated with my experience filtering spam emails, where simple word counts can be effective but benefit from TF-IDF's weighting to reduce noise from common words.

N-grams connect to social media analytics, such as analyzing Twitter (or X) posts for brand sentiment. The lab's n-gram analysis showed how phrases like "great acting" or "poor plot" reveal sentiment better than single words. Companies like Netflix could use this to identify viewer preferences from reviews, improving recommendation systems. This application became vivid when I saw bigrams like "must-watch" in the analysis, which are common in positive reviews.

Word embeddings have clear applications in chatbots, like those used by customer service platforms. The lab's exploration of semantic relationships (e.g., "movie" ≈ "film") showed how embeddings enable chatbots to understand varied user inputs. This connected to my interactions with chatbots that sometimes misinterpret queries, likely due to weaker embeddings or preprocessing, reinforcing the importance of robust representations.

The mini search engine demo in **Part 5** illustrated how TF-IDF powers search engines like Google, ranking documents by relevance to queries like "great acting performance." This made me think about how search engines prioritize results based on distinctive terms, a concept I now understand through TF-IDF's mechanics.

Questions That Arose

Several questions emerged during the lab, reflecting my curiosity about scaling these techniques. First, how do practitioners optimize the choice of representation method in large-scale systems? The lab compared BOW, TF-IDF, and embeddings, but real-world datasets are vast and diverse—how do teams decide which method to prioritize? Second, how do contextual embeddings like BERT improve upon the static embeddings we explored? I'm curious about their ability to handle polysemy (e.g., "bank" as riverbank vs. financial institution). Third, the ethical considerations section raised questions about bias mitigation: what practical techniques exist to debias embeddings, and how effective are they in real-world applications like hiring or content moderation? These questions inspire me to dive deeper into advanced NLP topics.

Comparisons Between Approaches

Comparing **BOW and TF-IDF** was insightful. BOW's simplicity—counting word frequencies—makes it fast and interpretable but ignores word importance. For example, common words like "movie" dominate BOW vectors, reducing their discriminative power. TF-IDF, by weighting rare terms like "cinematography," produced more meaningful similarity scores, as seen in the heatmap visualizations, making it better for tasks like document retrieval. However, both are sparse and lose word order, limiting their use in context-sensitive tasks.

N-grams versus **unigrams** highlighted the value of context. Unigrams like "great" are ambiguous, but bigrams like "great acting" or trigrams like "amazing visual effects" capture sentiment and

specificity, as shown in the n-gram analysis. However, higher-order n-grams increase vocabulary size, making them computationally expensive—a trade-off I noticed when the analysis slowed with trigrams.

Word embeddings versus **sparse representations** was the most striking comparison. Embeddings' dense vectors capture semantic relationships (e.g., "king" and "queen" being close in vector space), enabling tasks like analogy solving, which sparse methods cannot do. However, embeddings require large training data and are less interpretable, as seen in the similarity matrix, where scores were harder to trace back to specific words compared to TF-IDF. This comparison taught me to choose representations based on task needs—sparse for interpretability, dense for semantics.

Future Applications

I am excited about developing a sentiment analysis tool for online reviews, using TF-IDF or n-grams to classify sentiments in product or movie reviews. The lab's classification pipeline, with its ~0.80 accuracy for n-grams, provides a blueprint for this, and I'd experiment with preserving negation words like "not" to improve accuracy. Another project could be a content recommendation system for news articles, using TF-IDF to rank articles by relevance to user queries, similar to the mini search engine demo.

I also envision building a chatbot for a specific domain, like tech support, using word embeddings to understand user queries with varied phrasing (e.g., "fix" ≈ "repair"). The lab's embedding exploration showed how to leverage semantic similarities, which I could combine with spaCy's preprocessing from Lab 02 for robust intent detection. These projects feel achievable thanks to the lab's hands-on approach, giving me confidence to tackle real-world NLP challenges.

Conclusion

Lab 04 was a transformative journey into text representation, revealing the mechanics behind turning words into numbers. Overcoming challenges like the punkt_tab error, classification report issues, and the IndexError in Exercise 6 strengthened my debugging skills and deepened my understanding of NLP pipelines. The real-world connections to search engines, chatbots, and social media analytics made the concepts tangible, while comparisons between BOW, TF-IDF, n-grams, and embeddings clarified their trade-offs. The ethical considerations, especially around bias, underscored the responsibility of building fair NLP systems. My lingering questions about optimization and contextual embeddings motivate me to explore further, while the lab's practical exercises equip me to start building my own NLP applications. This experience has solidified my foundation in NLP, and I'm eager to apply these insights to create impactful, ethical AI solutions.

References (Used in Notebook)

https://www.nltk.org/api/nltk.tokenize.html

https://www.nltk.org/api/nltk.stem.html

https://www.nltk.org/api/nltk.corpus.html

https://scikit-learn.org/stable/modules/feature_extraction.html

https://radimrehurek.com/gensim/models/word2vec.html

https://www.nltk.org/api/nltk.corpus.html#nltk.corpus.stopwords

https://scikit-learn.org/stable/modules/model_evaluation.html