**Homonyms Problem in Sentiment Analysis**

1. **Introduction**

**1.1 Problem Statement**

1.1.1 Sentiment Analysis

* or opinion mining is the process of analyzing large volumes of text to determine whether it expresses a positive sentiment, a negative sentiment or a neutral sentiment.

1.1.2 Homonyms Problem in Sentiment Analysis

* refers to the challenges posed by words that are spelled the same (homographs) or sound the same (homophones) but have different meanings. This can complicate the sentiment analysis process for several reasons:
  + **Ambiguity**: Homonyms create ambiguity in text interpretation. For instance, the word "bark" can refer to the sound a dog makes or the outer covering of a tree. Sentiment analysis tools may struggle to accurately determine the sentiment without understanding the context.
  + **Context Dependence**: The sentiment of a sentence can change based on which meaning of a homonym is being used. For example:

"The bark was rough" (referring to tree bark) vs. "The dogs bark was loud" (referring to a sound).

The sentiment associated with each sentence can vary significantly, impacting analysis results.

* + **Data Sparsity**: In training sentiment analysis models, if homonyms appear in various contexts, it can lead to insufficient examples for some meanings, resulting in poor model performance on ambiguous terms.
  1. **Goals and Objectives**

Solving the Sentiment analysis problem by taking into consideration overcoming the homonyms problem.

1. **Data Description**

**2.1 Dataset Overview**

The used dataset is **Stanford Sentiment Treebank (SST-2)** which is a corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. The corpus is based on the dataset introduced by Pang and Lee (2005) and consists of 11,855 single sentences extracted from movie reviews. It was parsed with the Stanford parser and includes a total of 215,154 unique phrases from those parse trees, each annotated by 3 human judges.

Binary classification experiments on full sentences (negative or somewhat negative vs somewhat positive or positive with neutral sentences discarded) refer to the dataset as SST-2 or SST binary.

**2.2 Features and Labels**

The dataset includes **3** features and **70,042** instances (rows) between training, validation and testing.

Data Fields

* idx: Monotonically increasing index ID.
* sentence: Complete sentence expressing an opinion about a film.
* label: Sentiment of the opinion, either "negative" (0) or positive (1).

**2.3 Training and Testing Split**

| **Data Splits** | **Train** | **Validation** | **Test** |
| --- | --- | --- | --- |
| **Number of examples** | 67349 | 872 | 1821 |

1. **Baseline Experiments**

**3.1 Objective**

 The objective of trying different models is handling the homonyms problem for customer and selecting the best solution/model for overcome this problem.

**3.2 Baseline Models**

I tried different deep learning architectures, and It’s possible to try traditional machine learning models with embedding using Count Vectorizer, Count Vectorizer with N Grams, Word2Vec, TFIDF Vectorizer, Glove and Others, But I thought that Deep Learning will be more efficient for solving this problem.

* **Sequential Models**
  + In my opinion, If I start to handle the homonyms problem, the sequential models are the first solution but not the efficient that will come to my mind.
  + Because when using the sequential models, simply like RNN, I have the embedding vector (hidden state) that carries the contextual information for all the sentence, taking into consideration the different contexts for the same words and their orders. So, I will prefer to try these models, although I know that its accuracy won't be the best
  + **RNN, LSTM, GRU**
* **Bidirectional RNN**
* The second and that is better than the previous solution, because it is a good fit for the sentiment analysis problem, as it allows the model to consider words at the end of the sentence (capture context from both the beginning and the end), which might affect the overall sentiment.
  + Because The attention layer will help in getting different vectors that express different contexts for the same words and the encoder network will see all the input at the same time (No need to architecture such as Bidirectional RNN), So will have a good base to determine the sentiment.

**3.3 Evaluation Metrics**

I used the Accuracy as an evaluation metric and F1-Scor that made result close to accuracy. But I know that F1-Score Macro Average will be a correct choice in this problem because the dataset is imbalanced, and Macro calculate metrics for each label and find their unweighted mean. This does not take label imbalance into account.

* <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html>

**3.4 Results**

As I expected, bi-directional RNN (with GRU) made a very good result with three epochs:

* Training Accuracy: 97
* Validation Accuracy: 82

Unlike the Vanilla RNN that couldn’t classify normal cases:

* Training Accuracy: 55
* Validation Accuracy: 51

1. **Other Experiments**

I also think that architecture such as the transformer **encoder** or **LLMs** that depends on encoder architecture will be the best solution to solve this problem.

So, I tried to use:

* LLM models that its architecture depends on encoder, Like: Bert.
* Or that depends on encoder decoder transformer, like: T5 that ranks the first on Papers with Code Benchmarks.
* Or any other text classification LLM

**4.1 Results**

I thought the result of models like gemini, llama 3, llama 3.1 and mixtral weren't good because:

* They utilize encoder - decoder transformer architecture, so they are more generative than predictive.
* But There is high possibility if they fine-tuned on SST-2 dataset make a good result.
* Bert made a good accuracy better than bi-directional RNN and from the first epoch.

1. **Conclusion**

According to Papers with Code website, the best accuracy reached using this dataset was 97.5.

* Our Findings
  + Bi-Directional GRU:
    - **82.0** as Training Accuracy.
  + Bert:
    - **90.0** as a Testing Accuracy.
    - But I didn’t continue because it was taking very long time.
    - I’m sure if all epochs would complete, it would make a better accuracy than 90.0.

**6. Tools and Resources**

**6.1 Tools**

* Pandas
* Hugging Face
* TensorFlow
* LangChain
* Gemini API
* Groq API (LLAMA 3, LLAMA 3.1, Mixtral)
* Transformers (BERT)

**6.2 Resources**

* <https://paperswithcode.com/task/sentiment-analysis>
* <https://huggingface.co/datasets/stanfordnlp/sst2>
* <https://python.langchain.com/docs/integrations/chat/google_generative_ai/>
* <https://console.groq.com/docs/quickstart>
* <https://www.geeksforgeeks.org/sentiment-classification-using-bert/>
* <https://chatgpt.com/share/66f86b27-7a64-800d-89f1-db8b2efa3ad1>
* <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html>
* <https://medium.com/@manjindersingh_10145/sentiment-analysis-with-bert-using-huggingface-88e99deeec9a>

**7. Questions**

**7.1 Biggest Challenges**

* The time taken by training process is long. But I know that is because of large number of batches. Because the dataset is large. So that pushed me to train for small number of epochs.
* Determining which is the best dataset to use. That is a huge problem, but I preferred that the dataset I will use contains the examples of homonyms. So, I used SST-2 because it’s a large dataset that is highly expected to contain these examples.

**7.2 Key Learnings**

* Good Studying the model architecture makes me predict how this model will behave, and whether each one is better than the other and why. So, I learned from this project to study the theoretical knowledge very good, to can easily expect how the model will behave in which problem so I will save time for not using it.
* I didn’t use Bert before, so it’s a good trial to use it and knowing the possibility of using it as any deep learning network architecture.