Natural Language Processing 2024
Assignment-1 Report

Group No. 41

Task 1: Implementation of Tokenizer

The provided Python code implements a tokenizer based on the BytePair encoding algorithm. The implementation is contained within the `Tokenizer` class, which includes methods for learning vocabulary (`learn_vocabulary`) and tokenizing text (`tokenize`).

Implementation Details:

The implementation consists of the following components:

- **Vocabulary Learning:** The `learn_vocabulary` method processes the input corpus, updating the vocabulary with word frequencies. It then iteratively performs merges to learn the split rules and frequencies.
- **Merging Vocabulary:** The `merge_vocabulary` method takes a pair of characters, updates the merge rules, and merges the old vocabulary based on the new rule.
- **Tokenization:** The `tokenize` method divides the input text into individual letters, tokenizes it based on the learned merge rules, and returns a list of tokens for each sample.

Results:

The implementation has been evaluated based on the specified criteria, and the following results have been obtained:

1. All Possible Tokens in Vocabulary

The tokens obtained from the vocabulary are stored in the file `tokens.txt`. Each line in the file represents a unique token.

2. Merge Rules Learnt

The merge rules learned during the vocabulary learning process are stored in the file `merge_rules.txt`. Each line in the file represents a merge rule as a pair of comma-separated characters.

3. Split Tokens after Tokenizing Test Samples

The tokens obtained after tokenizing a set of test samples are stored in the file `tokenized_samples.txt`. Each line in the file represents a sample with tokens separated by commas.

Task 2

Implementation of Bigram Language Model:

We implemented a Bigram Language Model (BigramLM) within the `BigramLMWithEmotion` class, featuring methods for learning the bigram model from the dataset, applying Laplace and Kneser-Ney smoothing techniques, and generating text with emotion-oriented modifications.

Smoothing Algorithms Comparison:

The results of Laplace and Kneser-Ney smoothing algorithms were compared. As can be seen from the probabilities of the top 5 bigrams for each of these, the Kneser-Ney smoothing method is better suited for the task since Laplace smoothing is a very naive approach and can severely impact the distribution by assigning relatively high probabilities to unseen events. On the other hand, Kneser-Ney smoothing takes into account the continuation counts (thus considering context) while applying the smoothing such that the probabilities of high frequency events are not significantly reduced or that of low frequency events are not significantly boosted.

Top 5 Bigrams for each Method:

- **Top 5 Bigrams**

Bigram	**Probabilities**
<s>, i	0.2693
i, feel	0.1104
feel, like	0.0351
i, am	0.0319
<s>, im	0.0272

Table 1: Laplace Smoothing

Bigram	**Probabilities**
href, http	0.9800
don, t	0.9746
didn, t	0.9722
sort, of	0.9708
supposed, to	0.9450

Table 2: Kneser Ney Smoothing

Bigram	**Probabilities**
href, http	1.0
moshilu, <eos>	1.0
tychelle, to	1.0
hang, out	1.0
nonexistent, social	1.0

Table 3: Without Smoothing

Emotion Modification:

The emotion component (β) was integrated into the standard bigram probability calculation using the formula:

$$\beta = P[w_{i-1}, w_i] + \left(\frac{\text{em}_{(w_{i-1}, w_i)}}{\text{em}_{(w_{i-1})}} \right)$$

Here, $\text{em}_{(w_{i-1}, w_i)}$ refers to the emotional score of the bigram, and $\text{em}_{(w_{i-1})}$ refers to the emotional score of the first word of the bigram.

The β term in the bigram model acknowledges the influence of the preceding unigram on the current bigram. This recognition is crucial as any bigram depends on its precedent unigram, forming a chain of dependencies back to the initial unigram. The emotion component of the bigram is thus influenced strongly by that of the unigram and is thought to be a major component of β.

Extrinsic Evaluation Results

- The accuracy and macro F1 scores obtained from the extrinsic evaluation using the SVC model are reported as follows.

Accuracy	**Macro F1 Score**
72.33%	0.72

Table 4: Evaluation Results

- **Generated Samples**

Emotions	**Sample 1**	**Sample 2**
Love	date by thats also like went dear	enough was horny the than my feeling i to
Fear	chills in has and my but before	which or im them arms intimidated and hand vaguely
Surprise	that like how for cranky just in	wowed boyfriend impressed and stories and dont
Anger	smother and annoyed sitting remember day deny a while asking	hops dangerous my say pissed chris runner enough
Sadness	to suicide struggle and hopelessly losing letter of	something emotionally guilt q again doesnt quiet
Joy	so and the tips different relaxed are actually	mom moment talented cruz business getting it

- **Sample Explanation**
  As can be seen from the samples listed below, all samples tend to have certain words that correspond to the emotion the sample belongs to. Evaluating from a human perspective, we can see that words like "date" or "dear" are most related to the "love" emotion out of all and thus tilt the emotion score of the generated sentence towards "love"
  In the sample expressing surprise, the use of the word 'surprised' itself indicates the presence of surprise emotion.

Emotions	**Sample 1**	**Indicative words**
Love	date by thats also like went dear	'date' 'dear'
Fear	chills in has and my but before	'chills'
Surprise	that like how for cranky just in	'cranky'
Anger	smother and annoyed sitting remember day deny a while asking	'smother' 'annoyed'
Sadness	to suicide struggle and hopelessly losing letter of	'suicide' 'struggle 'hopelessly' 'losing'
Joy	so and the tips different relaxed are actually	'relaxed'

Credit Statement

- **Arjit Singh Arora** - Task 1 + 2
- **Akshat Gupta** - Task 2
- **Kumar Aryan Singh** - Task 1 + 2
- **Swati Sharma** - Task 2