Natural Language Processing

Ngram and Naive Bayes Lab

(Artificial Intelligence – AI 502)

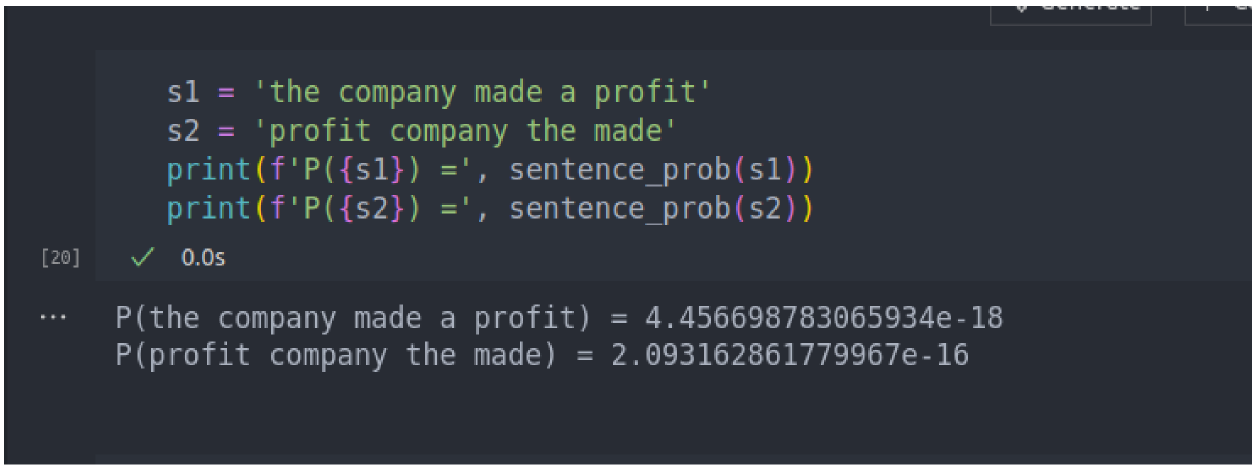
(Fall-2025)

Submitted By: Syed Nouman (2025MSAI104)
To: Dr. Muhammad Faisal Hayat

University of Engineering and Technology, Lahore

Exercise 1

Compute sentence probabilities for: "the company made a profit" "profit company the made"?



```
s1 = 'the company made a profit'
s2 = 'profit company the made'
print(f'P({s1}) =', sentence_prob(s1))
print(f'P({s2}) =', sentence_prob(s2))
```

```
P(the company made a profit) = 4.456698783065934e-18
P(profit company the made) = 2.093162861779967e-16
```

P(the company made a profit) = 4.456698783065934e-18

P(profit company the made) = 2.093162861779967e-16

Compare which is higher (more grammatical).

We compare the two numbers:
4.456698783065934 x 10^-18
2.093162861779967 x 10^-16

Means

P("profit company the made") > P("the company made a profit")

That means our current bigram model assigns a higher probability to "profit company the made", even though it's not grammatically correct in English.

This is a common issue with simple bigram models:

- Model only look at local word pairs (like "profit company" and "company the"), not the overall grammatical structure.
- If "profit company" or "company the" appear more often in training data, they can make that weird sentence seem more likely statistically.

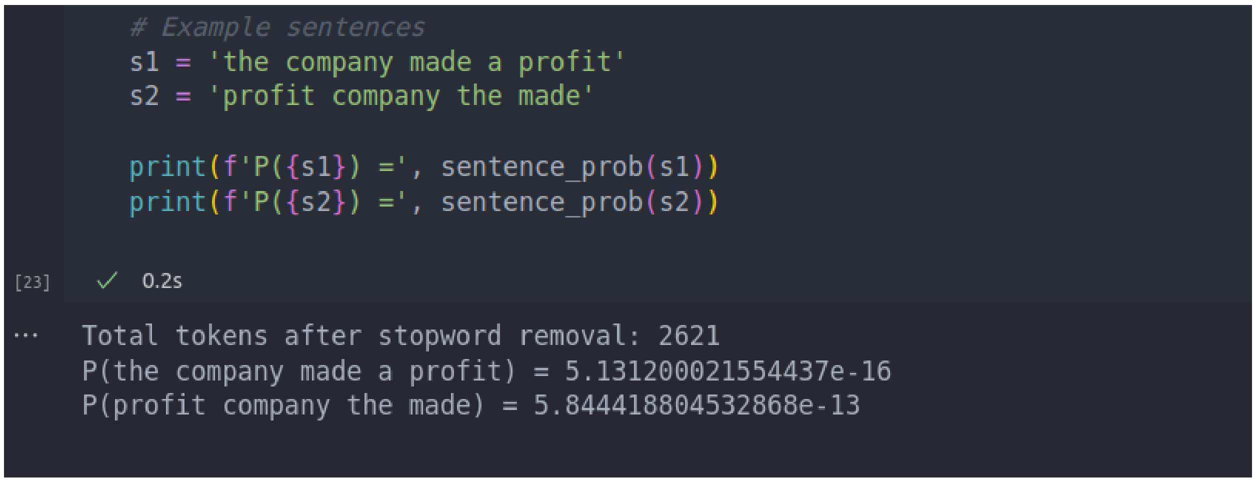
Compute perplexity on 10 unseen sentences from Reuters.

- the company announced a dividend
- the stock rose after the report
- the market closed higher today
- analysts expect further gains
- the bank reported quarterly profits
- investors were encouraged by the results
- shares of the company increased
- economic growth remains strong
- pacificorp said it plans new investments
- the deal was approved by regulators

Perplexity: 3819.603289525883

Exercise 2

Try removing stopwords and see how accuracy changes.



```
# Example sentences
s1 = 'the company made a profit'
s2 = 'profit company the made'

print(f'P({s1}) =', sentence_prob(s1))
print(f'P({s2}) =', sentence_prob(s2))
```

```
[23]  ✓  0.2s
```

```
...  Total tokens after stopword removal: 2621
     P(the company made a profit) = 5.131200021554437e-16
     P(profit company the made) = 5.844418804532868e-13
```

Before removing stopwords:

Model used all tokens (including “the”, “a”, “is”, etc.)
These common words help capture grammatical structure and better fluency modeling.

After removing stopwords:
Many frequent connective words (“the”, “a”, “is”, “of”) are removed.
This makes sentence patterns less predictable and grammar less captured results in probabilities drop or become less reliable.

Removing stopwords generally reduces accuracy (and increases perplexity) because the model loses important structural cues.
Results confirm this as we can see the sentence probabilities changed drastically (became much smaller).

Exercise 3

Explain how N-gram models capture syntax (local dependencies) while NB models capture semantics (word sentiment).

N-gram models look at how words appear together.
For example, pairs or triples of consecutive words.

This helps them learn word order and grammar-like patterns or syntax.

Example:

“the company made a profit” → has bigrams like (“the company”), (“company made”), (“made a”), (“a profit”).
The model learns that “made a” is common, but “profit company” is not.

So, N-gram models are good at capturing local word order i.e. how words fit together grammatically.

While

Naive Bayes ignores word order i.e. it only cares which words appear, not where.
It uses word frequencies to guess the overall meaning or sentiment.

Example:

Sentence: “This phone is amazing and beautiful.”
Important words: amazing, beautiful
Naive Bayes says: Positive (because these words are usually found in good reviews)

Sentence: “This phone is terrible and boring.”
Important words: terrible, boring
Naive Bayes says: Negative (because these words are common in bad reviews)

So, NB models focus on word meaning (semantics) rather than order.

Optional Conceptual Questions

1. Why does smoothing improve perplexity on unseen text?
Smoothing adds a small probability to words or word pairs that the model never saw during training.
Without smoothing, any unseen word pair would get probability zero, which makes perplexity infinite.
By giving a small non-zero probability to unseen words, smoothing makes the model more general and helps it handle new or rare sentences better.

2. How does perplexity relate to branching probability of a language model?
Perplexity tells us how confused a language model is when predicting the next word.
It's like asking: "On average, how many choices does the model consider possible for each word?"
A lower perplexity means the model is more confident (less confused).

Low perplexity → few likely next words → better model
High perplexity → many possible next words → weaker model

3. Why might Naïve Bayes misclassify "not good"?
Naive Bayes treats each word independently.
It sees the word "good" (positive) and ignores the effect of "not" before it.
So it might classify "not good" as positive, even though it means negative.

4. How can we use a held-out corpus to estimate interpolation weights between unigrams, bigrams, trigrams?
A held-out corpus is a small set of data not used in training. We use it to test how well different combinations of unigram, bigram, and trigram probabilities perform.
By adjusting the weights (λ_1, λ_2, λ_3) for each model on this held-out data, we find the mix that gives the lowest perplexity.
Use extra data to tune how much trust to give to unigram, bigram, and trigram models for better predictions.

Comparison of Statistical vs Lexical Approaches to Sentiment Detection

1. Lexical Approach

- Uses predefined word lists (lexicons), lists of positive and negative words.
- Counts how many positive and negative words appear in a sentence.
- Decides the sentiment based on which count is higher.

Example:
Sentence: “The movie was wonderful but slow.”
Positive words: wonderful
Negative words: slow
Overall: Mixed or slightly positive

Advantages:

- Simple and easy to understand.
- Works without any training data.

Disadvantages:

- Doesn’t handle context or negation well (e.g., “not good” → wrongly positive).
- Misses subtle meanings or sarcasm.

2. Statistical Approach

- Uses machine learning models (like Naive Bayes, SVM, or neural networks).
- Learns patterns from labeled training data i.e. many examples of positive and negative text.
- Finds which words or combinations of words are strong signals of sentiment.

Example:
The model learns from data that “amazing”, “great” → positive
and “awful”, “terrible” → negative.

Advantages:

- Learns from data automatically.
- Can capture context and word importance better than lexicon methods.