# Text Mining and Analytics

Session 5: Class Activity
**N-Gram Language Models**

Instructor: Behrooz Mansouri
Spring 2024, University of Southern Maine

# Files Structure

- On BrightSpace you have a Zip file named “TM_CA1_Lyrics”
- There are 6 directories, one per music genre
  - Rock, Rap, Pop, Metal, Country, Blues
  - In each directory, there are 21 songs for that genre

- Notes: the lyrics might need preprocessing

# Building Unigram Language Model

We will build an unigram language model for each genre; to do this, we follow these steps:

1. Use the file UnigramModel.py
    a. Complete the method “read_file_in_directory”; this method takes in the directory path, iterate on all songs, and returns a dictionary
    b. This dictionary has tokens as the keys and their frequencies as values

10–15 Minutes

# Building Unigram Language Model

We will build an unigram language model for each genre; to do this, we follow these steps:

1. Use the file UnigramModel.py
   a. Complete the method “read_file_in_directory”; this method takes in the directory path, iterate on all songs, and returns a dictionary
   b. This dictionary has tokens as the keys and their frequencies as values
2. Complete the freq_to_prob method; This method converts the frequencies to probabilities

$$P(w_i) = \frac{c(w_i)}{\sum_{\tilde{w}} c(\tilde{w})}$$

3–5 Minutes

# Building Unigram Language Model

We will build an unigram language model for each genre; to do this, we follow these steps:

1. Use the file UnigramModel.py
   a. Complete the method “read_file_in_directory”; this method takes in the directory path, iterate on all songs, and returns a dictionary
   b. This dictionary has tokens as the keys and their frequencies as values
2. Complete the freq_to_prob method; This method converts the frequencies to probabilities
3. Complete the calculate_probablity method; this method takes in the input string and the dictionary from step 2 and returns the probability of that string being generated by the language model

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

3–5 Minutes

# Building Unigram Language Model

We will build an unigram language model for each genre; to do this, we follow these steps:

1. Use the file UnigramModel.py
   a. Complete the method “read_file_in_directory”; this method takes in the directory path, iterate on all songs, and returns a dictionary
   b. This dictionary has tokens as the keys and their frequencies as values
2. Complete the freq_to_prob method; This method converts the frequencies to probabilities
3. Complete the calculate_probablity method; this method takes in the input string and the dictionary from step 2 and returns the probability of that string being generated by the language model
4. Prepare your code for testing; you will be given input text, and you have find the genre for which its language model give the highest probability of input text being generated by its language model

10 Minutes

# Test Cases

- For each test case, submit your answer to Slack Channel: “Class Activities” (Screenshot showing probability per genre)

You used to call me on my cell phone
Late night when you need my love
Call me on my cell phone

# Test Cases

- For each test case, submit your answer to Slack Channel: “Class Activities” (Screenshot showing probability per genre)

If you like to gamble
I tell you, I'm your man
You win some, lose some
It's all the same to me

# Test Cases

- For each test case, submit your answer to Slack Channel: “Class Activities” (Screenshot showing probability per genre)

A singer in a smokey room
A smell of wine and cheap perfume
For a smile they can share the night
It goes on and on and on and on

# Test Cases

- For each test case, submit your answer to Slack Channel: “Class Activities” (Screenshot showing probability per genre)

Yes, the eagle flies on Friday and Saturday I go out to play
Eagle flies on Friday and Saturday I go out to play
Sunday I go to church, then I kneel down and pray

# Test Cases

- For each test case, submit your answer to Slack Channel: “Class Activities” (Screenshot showing probability per genre)

If you want my future, forget my past
If you wanna get with me, better make it fast
Now don't go wasting my precious time

# Test Cases

- For each test case, submit your answer to Slack Channel: “Class Activities” (Screenshot showing probability per genre)

Sometimes it's hard to be a woman
Giving all your love to just one man
You'll have bad times
And he'll have good times
Doin' things that you don't understand

# Test Cases

- For each test case, submit your answer to Slack Channel: “Class Activities” (Screenshot showing probability per genre)

If you like to gamble
I tell you, I'm your man
You win some, lose some
It's all the same to me
The pleasure is to play
Makes no difference what you say
I don't share your greed
The only card I need
Is the Ace of Spades
The Ace of Spades
Playing for the high one
Dancing with the devil
Going with the flow
It's all a game to me

# Building Bigram Language Model

Now build a bigram language model for each genre; to do this, we follow these steps:

1. Create the file BigramModel.py

- Complete the method “read_file_in_directory”; this method takes in the directory path, iterate on all songs, and returns two dictionaries (frequency of terms and pairs)

# Building Bigram Language Model

Now build a bigram language model for each genre; to do this, we follow these steps:

1. Create the file BigramModel.py
- Complete the method “read_file_in_directory”; this method takes in the directory path, iterate on all songs, and returns two dictionaries (frequency of terms and pairs)
2. Complete freq_to_prob method to calculate probability for a given pair with smoothing (Add-one)

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

# Building Bigram Language Model

Now build a bigram language model for each genre; to do this, we follow these steps:

1. Create the file BigramModel.py
- Complete the method “read_file_in_directory”; this method takes in the directory path, iterate on all songs, and returns two dictionaries (frequency of terms and pairs)
2. Complete freq_to_prob method to calculate probability for a given pair with smoothing (Add-one)
3. Using the same log-based approach, complete calculate_probablity method

# Building Bigram Language Model

Now build a bigram language model for each genre; to do this, we follow these steps:

1. Create the file BigramModel.py
- Complete the method “read_file_in_directory”; this method takes in the directory path, iterate on all songs, and returns two dictionaries (frequency of terms and pairs)
2. Complete freq_to_prob method to calculate probability for a given pair with smoothing (Add-one)
3. Using the same log-based approach, complete calculate_probablity method
4. Prepare your code for testing; you will be given input text, and you have find the genre for which its language model give the highest probability of input text being generated by its language model

# Next Session

# Vector Semantics

We will explore

- Text Classification
- Naïve Bayes Classifier
- Evaluation and significance testing