Advanced Methods in Natural Language Processing

Session 1: Introduction, Baselines, Evaluations & TF-IDF

Arnault Gombert

February 2024

Barcelona School of Economics

Today's Class

Class Overview: Introduction

- **Introduction to NLP**
  - Brief history and evolution
  - Importance in current technology landscape
  - Our class in a nutshell
- **Evaluation Metrics in NLP**
  - Overview of key metrics
  - Application and interpretation
- **Baselines in NLP**
  - Understanding baseline models
  - Importance and examples
- **TF-IDF and Its Improvements**
  - Deep dive into TF-IDF
  - Advanced techniques and applications
- **QA and Wrap-Up**
  - Open discussion
  - Summary of key takeaways

Introduction

Brief History and Evolution of NLP since 1950

- **1950s**: Turing Test introduction; early machine translation experiments.
- **1960s - 1970s**: Emergence of rule-based systems, ELIZA; focus on syntax and grammar.
- **1980s**: Computational advancements; shift towards statistical methods.
- **1990s**: Rise of statistical models; RNNs: early machine/deep learning approaches in NLP.
- **2000s**: Growth in machine learning techniques; algorithms like SVM, decision trees, Language Modeling.
- **2010s**: Deep learning revolution; models like Word2Vec, Transformer, BERT.
- **2020s**: Widespread application; advances in contextual understanding, sentiment analysis.

Natural Language Processing Today

- **Search Engines** (Google, Bing, Yahoo)

Natural Language Processing Today

-  **Search Engines** (Google, Bing, Yahoo)
  - *Example*: LexRank algorithm for page ranking (unsupervised learning).

Natural Language Processing Today

-  **Search Engines** (Google, Bing, Yahoo)
  - *Example*: LexRank algorithm for page ranking (unsupervised learning).
-  **Smartphone Keyboards**

Natural Language Processing Today

-  **Search Engines** (Google, Bing, Yahoo)
  - *Example*: LexRank algorithm for page ranking (unsupervised learning).
-  **Smartphone Keyboards**
  - *Example*: Predictive text and autocorrect (language modeling, supervised learning).

Natural Language Processing Today

-  **Search Engines** (Google, Bing, Yahoo)
  - *Example*: LexRank algorithm for page ranking (unsupervised learning).
-  **Smartphone Keyboards**
  - *Example*: Predictive text and autocorrect (language modeling, supervised learning).
-  **Translation Tools** (Google Translate, DeepL)

Natural Language Processing Today

-  **Search Engines** (Google, Bing, Yahoo)
  - *Example*: LexRank algorithm for page ranking (unsupervised learning).
-  **Smartphone Keyboards**
  - *Example*: Predictive text and autocorrect (language modeling, supervised learning).
-  **Translation Tools** (Google Translate, DeepL)
  - *Example*: Encoder/Decoder models for language translation (supervised learning).

Natural Language Processing Today

-  **Search Engines** (Google, Bing, Yahoo)
  - *Example*: LexRank algorithm for page ranking (unsupervised learning).
-  **Smartphone Keyboards**
  - *Example*: Predictive text and autocorrect (language modeling, supervised learning).
-  **Translation Tools** (Google Translate, DeepL)
  - *Example*: Encoder/Decoder models for language translation (supervised learning).
-  **Email Spam Detection**

Natural Language Processing Today

-  **Search Engines** (Google, Bing, Yahoo)
  - *Example*: LexRank algorithm for page ranking (unsupervised learning).
-  **Smartphone Keyboards**
  - *Example*: Predictive text and autocorrect (language modeling, supervised learning).
-  **Translation Tools** (Google Translate, DeepL)
  - *Example*: Encoder/Decoder models for language translation (supervised learning).
-  **Email Spam Detection**
  - *Example*: Classifying emails as spam or not (supervised learning).

Natural Language Processing Today

- **Search Engines** (Google, Bing, Yahoo)
  - *Example*: LexRank algorithm for page ranking (unsupervised learning).
- **Smartphone Keyboards**
  - *Example*: Predictive text and autocorrect (language modeling, supervised learning).
- **Translation Tools** (Google Translate, DeepL)
  - *Example*: Encoder/Decoder models for language translation (supervised learning).
- **Email Spam Detection**
  - *Example*: Classifying emails as spam or not (supervised learning).
- **Social Media** (Facebook, Twitter)

Natural Language Processing Today

- **Search Engines** (Google, Bing, Yahoo)
  - *Example*: LexRank algorithm for page ranking (unsupervised learning).
- **Smartphone Keyboards**
  - *Example*: Predictive text and autocorrect (language modeling, supervised learning).
- **Translation Tools** (Google Translate, DeepL)
  - *Example*: Encoder/Decoder models for language translation (supervised learning).
- **Email Spam Detection**
  - *Example*: Classifying emails as spam or not (supervised learning).
- **Social Media** (Facebook, Twitter)
  - *Example*: Content recommendations based on user interests (word embeddings, similarity algorithms).

Our Program

Part I - Good old fashioned NLP

1. *Session 1*: Baselines and Sparse representations - Baseline Models, Evaluations, TF-IDF and improvements
2. *Session 2*: Deep Learning - Backpropagation in Neural Networks, LSTM, Attention Processes, Language Models
3. *Session 3*: Word Embeddings - Static (Word2Vec, GloVe, FastText) and Contextual Embeddings (ELMo, BERT)
4. *Session 4*: **Practical Session + Homework** - Baseline Pipeline, Metrics Evaluation, LSTM-Pipeline, Training Own Embeddings

Part II - Almost Part of Good Old Fashioned NLP

5. *Session 5*: Transformer Architecture, Self-Attention, BERT Architecture
6. *Session 6*: Few Shot Learning, Transfer Learning - Fine-Tuning BERT, Leveraging Existing Knowledge, Prompts in Learning
7. *Session 7*: Injustice Biases in NLP - Detecting and Mitigating Biases, Large Language Models
8. *Session 8*: **Practical Session + Homework** - Fine-Tuning BERT, Data Requirements, Low Resource Solutions, Detecting Biases

Part III - LLMs, ChatGPT Others

9. *Session 9*: Prompt Engineering Fine-Tuning - Zero Shot Learning, Chain of Thoughts, Formatting Outputs
10. *Session 10*: Hallucinations Other Limitations - Detecting Hallucinations, Understanding Limitations

Class Evaluation Criteria

- **Participation** - *10%*
  - Active engagement in class discussions.
  - Attendance and involvement in interactive sessions.
- **Homework** - *20%*
  - 2 homework assignments.
  - Application of class concepts and timely submission.
- **Team Project** (3-4 Students) - *70%*
  - Collaborative team project.
  - Application of NLP concepts and techniques learned in class.
  - Final paper submission.

Today's class

Evaluation of models

Key Tasks in NLP and Related Metrics - I

1. **Text Classification**
   - *Tasks*: Sentimental Analysis, Spam Detection, Topic Assignment, Document Categorization.

2. **Named Entity Recognition (NER)**

3. **Topic Modeling**

Key Tasks in NLP and Related Metrics - I

1. **Text Classification**
   - *Tasks*: Sentimental Analysis, Spam Detection, Topic Assignment, Document Categorization.
   - *Metrics*: Accuracy, Precision, Recall, F1-Score.
2. **Named Entity Recognition (NER)**
3. **Topic Modeling**

Key Tasks in NLP and Related Metrics - I

1. **Text Classification**
   - *Tasks*: Sentimental Analysis, Spam Detection, Topic Assignment, Document Categorization.
   - *Metrics*: Accuracy, Precision, Recall, F1-Score.
2. **Named Entity Recognition (NER)**
   - *Tasks*: Identifying names, organizations, locations in text.
3. **Topic Modeling**

Key Tasks in NLP and Related Metrics - I

1. **Text Classification**
   - *Tasks*: Sentimental Analysis, Spam Detection, Topic Assignment, Document Categorization.
   - *Metrics*: Accuracy, Precision, Recall, F1-Score.
2. **Named Entity Recognition (NER)**
   - *Tasks*: Identifying names, organizations, locations in text.
   - *Metrics*: Precision, Recall, F1-Score, Entity-Level Accuracy.
3. **Topic Modeling**

Key Tasks in NLP and Related Metrics - I

1. **Text Classification**
   - *Tasks*: Sentimental Analysis, Spam Detection, Topic Assignment, Document Categorization.
   - *Metrics*: Accuracy, Precision, Recall, F1-Score.
2. **Named Entity Recognition (NER)**
   - *Tasks*: Identifying names, organizations, locations in text.
   - *Metrics*: Precision, Recall, F1-Score, Entity-Level Accuracy.
3. **Topic Modeling**
   - *Tasks*: Discovering topics in large text corpora.

Key Tasks in NLP and Related Metrics - I

1. **Text Classification**
   - *Tasks*: Sentimental Analysis, Spam Detection, Topic Assignment, Document Categorization.
   - *Metrics*: Accuracy, Precision, Recall, F1-Score.
2. **Named Entity Recognition (NER)**
   - *Tasks*: Identifying names, organizations, locations in text.
   - *Metrics*: Precision, Recall, F1-Score, Entity-Level Accuracy.
3. **Topic Modeling**
   - *Tasks*: Discovering topics in large text corpora.
   - *Metrics*: Coherence Score, Perplexity, Human Evaluation.

Key Tasks in NLP and Related Metrics - II

4. **Machine Translation**
   - *Tasks*: Translating text from one language to another.

5. **Text Generation**

6. **Question Answering**

Key Tasks in NLP and Related Metrics - II

4. **Machine Translation**
   - *Tasks*: Translating text from one language to another.
   - *Metrics*: BLEU, METEOR, TER.
5. **Text Generation**

6. **Question Answering**

Key Tasks in NLP and Related Metrics - II

4. **Machine Translation**
   - *Tasks*: Translating text from one language to another.
   - *Metrics*: BLEU, METEOR, TER.
5. **Text Generation**
   - *Tasks*: Automated content creation, dialogue generation.
6. **Question Answering**

Key Tasks in NLP and Related Metrics - II

4. **Machine Translation**
   - *Tasks*: Translating text from one language to another.
   - *Metrics*: BLEU, METEOR, TER.
5. **Text Generation**
   - *Tasks*: Automated content creation, dialogue generation.
   - *Metrics*: BLEU, ROUGE, Perplexity, Human Evaluation.
6. **Question Answering**

Key Tasks in NLP and Related Metrics - II

4. **Machine Translation**
   - *Tasks*: Translating text from one language to another.
   - *Metrics*: BLEU, METEOR, TER.
5. **Text Generation**
   - *Tasks*: Automated content creation, dialogue generation.
   - *Metrics*: BLEU, ROUGE, Perplexity, Human Evaluation.
6. **Question Answering**
   - *Tasks*: Building systems that automatically answer questions.

Key Tasks in NLP and Related Metrics - II

4. **Machine Translation**
   - *Tasks*: Translating text from one language to another.
   - *Metrics*: BLEU, METEOR, TER.
5. **Text Generation**
   - *Tasks*: Automated content creation, dialogue generation.
   - *Metrics*: BLEU, ROUGE, Perplexity, Human Evaluation.
6. **Question Answering**
   - *Tasks*: Building systems that automatically answer questions.
   - *Metrics*: F1-Score, Exact Match, BLEU.

A Closer Look at Key NLP Metrics

Note: Metrics are essential in NLP to assess problem difficulty and solution quality. They quantify success and guide improvements.

- **Text Classification Metrics**

A Closer Look at Key NLP Metrics

Note: Metrics are essential in NLP to assess problem difficulty and solution quality. They quantify success and guide improvements.

- **Text Classification Metrics**
    - *Recall*: Identifies relevant instances.

A Closer Look at Key NLP Metrics

Note: Metrics are essential in NLP to assess problem difficulty and solution quality. They quantify success and guide improvements.

- **Text Classification Metrics**
  - *Recall*: Identifies relevant instances.
  - *Precision*: Accuracy of identifying relevant instances.

A Closer Look at Key NLP Metrics

Note: Metrics are essential in NLP to assess problem difficulty and solution quality. They quantify success and guide improvements.

- **Text Classification Metrics**
  - *Recall*: Identifies relevant instances.
  - *Precision*: Accuracy of identifying relevant instances.
  - *F1-Score*: Balance of Precision and Recall.

A Closer Look at Key NLP Metrics

Note: Metrics are essential in NLP to assess problem difficulty and solution quality. They quantify success and guide improvements.

- **Text Classification Metrics**
  - *Recall*: Identifies relevant instances.
  - *Precision*: Accuracy of identifying relevant instances.
  - *F1-Score*: Balance of Precision and Recall.
- **General NLP Metrics**

A Closer Look at Key NLP Metrics

Note: Metrics are essential in NLP to assess problem difficulty and solution quality. They quantify success and guide improvements.

- **Text Classification Metrics**
  - *Recall*: Identifies relevant instances.
  - *Precision*: Accuracy of identifying relevant instances.
  - *F1-Score*: Balance of Precision and Recall.
- **General NLP Metrics**
  - *Model Loss*: Model's prediction accuracy.

A Closer Look at Key NLP Metrics

Note: Metrics are essential in NLP to assess problem difficulty and solution quality. They quantify success and guide improvements.

- **Text Classification Metrics**
  - *Recall*: Identifies relevant instances.
  - *Precision*: Accuracy of identifying relevant instances.
  - *F1-Score*: Balance of Precision and Recall.
- **General NLP Metrics**
  - *Model Loss*: Model's prediction accuracy.
  - *NPMI*: Association strength in topic modeling.

A Closer Look at Key NLP Metrics

Note: Metrics are essential in NLP to assess problem difficulty and solution quality. They quantify success and guide improvements.

- **Text Classification Metrics**
  - *Recall*: Identifies relevant instances.
  - *Precision*: Accuracy of identifying relevant instances.
  - *F1-Score*: Balance of Precision and Recall.
- **General NLP Metrics**
  - *Model Loss*: Model's prediction accuracy.
  - *NPMI*: Association strength in topic modeling.
  - *ROUGE*: Summarization and translation evaluation.

A Closer Look at Key NLP Metrics

Note: Metrics are essential in NLP to assess problem difficulty and solution quality. They quantify success and guide improvements.

- **Text Classification Metrics**
  - *Recall*: Identifies relevant instances.
  - *Precision*: Accuracy of identifying relevant instances.
  - *F1-Score*: Balance of Precision and Recall.
- **General NLP Metrics**
  - *Model Loss*: Model's prediction accuracy.
  - *NPMI*: Association strength in topic modeling.
  - *ROUGE*: Summarization and translation evaluation.
- **Benchmarks**

A Closer Look at Key NLP Metrics

Note: Metrics are essential in NLP to assess problem difficulty and solution quality. They quantify success and guide improvements.

- **Text Classification Metrics**
  - *Recall*: Identifies relevant instances.
  - *Precision*: Accuracy of identifying relevant instances.
  - *F1-Score*: Balance of Precision and Recall.
- **General NLP Metrics**
  - *Model Loss*: Model's prediction accuracy.
  - *NPMI*: Association strength in topic modeling.
  - *ROUGE*: Summarization and translation evaluation.
- **Benchmarks**
  - *GLUE*: Variety of language tasks (Wang et al., 2019).

A Closer Look at Key NLP Metrics

Note: Metrics are essential in NLP to assess problem difficulty and solution quality. They quantify success and guide improvements.

- **Text Classification Metrics**
  - *Recall*: Identifies relevant instances.
  - *Precision*: Accuracy of identifying relevant instances.
  - *F1-Score*: Balance of Precision and Recall.
- **General NLP Metrics**
  - *Model Loss*: Model's prediction accuracy.
  - *NPMI*: Association strength in topic modeling.
  - *ROUGE*: Summarization and translation evaluation.
- **Benchmarks**
  - *GLUE*: Variety of language tasks (Wang et al., 2019).
  - *X-TREME*: Cross-lingual tasks (Hu et al., 2020).

Understanding Metrics Through Applications

1. **High Accuracy, Low Recall, and Precision**
   - *Scenario*: A spam filter mostly marks all emails as non-spam.
   - *Question*: Can we be sure the model is good despite high accuracy?

Understanding Metrics Through Applications

1. **High Accuracy, Low Recall, and Precision**
   - *Scenario*: A spam filter mostly marks all emails as non-spam.
   - *Question*: Can we be sure the model is good despite high accuracy?

Answer:

- High accuracy with low recall and precision can be misleading, especially in imbalanced datasets. It might not be a good model.

Missile Detection Algorithm

Scenario:

- *W*hat is the best metric to use for a **missile detection algorithm**?
- *Consider*: The implications of high recall vs. high precision.

Missile Detection Algorithm

Scenario:

- *W*hat is the best metric to use for a **missile detection algorithm**?
- *Consider*: The implications of high recall vs. high precision.

Answer:

- High Recall/Low Precision: Avoids missing detections, but may cause false alarms.
- Low Recall/High Precision: Reduces false alarms, but might miss actual threats.

Trial Algorithm in a Judicial System

Scenario:

- is the best metric to use for a **trial algorithm in a judicial system**?
- *Consider*: The implications of high recall vs. high precision.

Trial Algorithm in a Judicial System

Scenario:

- is the best metric to use for a **trial algorithm in a judicial system**?
- *Consider*: The implications of high recall vs. high precision.

Answer:

- *High Recall / Low Precision*: Prioritizes ensuring no guilty party is missed, but risks more false positives (wrongful accusations).
- *Low Recall / High Precision*: Focuses on minimizing wrongful accusations but might miss identifying some guilty parties.

Computational Metrics

1. **Computational Efficiency**
   - *Number of Texts Processed per Second*: Measures the model's speed, crucial for real-time applications.
   - *RAM Used*: Indicates the model's memory efficiency, important for deployment in limited-resource environments.

2. Environmental Impact

- *CO2 Equivalents (Strubell et al., 2019)*: Assesses the environmental footprint of training and running NLP models.
- *Software Carbon Intensity (Dodge et al., 2022)*: Measures the carbon efficiency of software, highlighting the need for greener algorithms.

3. Fairness and Bias

- *False Positive/Negative Rates by Demographic Segment*: Evaluates the model's fairness across different groups.
- *Qualitative Study of Outputs with Prompts (Sheng et al., 2019)*: Investigates subtle biases in model responses.

Baselines: The Best Tool to Explore from Intuition

Best Practices for ML Engineering

Based on Google's Best Practices for ML Engineering, Zinkevich et al. (2022)

- **Rule #1: Launch Products without ML Fearlessly**
  - Emphasizes starting simple. Advanced ML is not always the initial answer.
- **Rule #2: Prioritize Metrics Design and Implementation**
  - Stresses the importance of defining success metrics before ML integration.
- **Rule #3: Prefer ML to Complex Heuristics**
  - Recommends using ML for problems too complex for heuristic approaches.

Scenario: A Simple Decision Rule

- **Context**: Implementing a system to categorize emails as 'urgent' or 'non-urgent' based on a specific keyword.

Scenario: A Simple Decision Rule

- **Context**: Implementing a system to categorize emails as 'urgent' or 'non-urgent' based on a specific keyword.
- **Traditional Approach**: Using a simple keyword search algorithm to flag emails containing words like 'urgent' or 'immediate'.

Scenario: A Simple Decision Rule

- **Context**: Implementing a system to categorize emails as 'urgent' or 'non-urgent' based on a specific keyword.
- **Traditional Approach**: Using a simple keyword search algorithm to flag emails containing words like 'urgent' or 'immediate'.
- **Machine Learning Approach**: Training a complex NLP model to understand the context and categorize emails.

Scenario: A Simple Decision Rule

- **Context**: Implementing a system to categorize emails as 'urgent' or 'non-urgent' based on a specific keyword.
- **Traditional Approach**: Using a simple keyword search algorithm to flag emails containing words like 'urgent' or 'immediate'.
- **Machine Learning Approach**: Training a complex NLP model to understand the context and categorize emails.
- **Analysis**:

Scenario: A Simple Decision Rule

- **Context**: Implementing a system to categorize emails as 'urgent' or 'non-urgent' based on a specific keyword.
- **Traditional Approach**: Using a simple keyword search algorithm to flag emails containing words like 'urgent' or 'immediate'.
- **Machine Learning Approach**: Training a complex NLP model to understand the context and categorize emails.
- **Analysis**:
    - *Complexity*: ML adds unnecessary complexity for a task solvable with basic programming.

Scenario: A Simple Decision Rule

- **Context**: Implementing a system to categorize emails as 'urgent' or 'non-urgent' based on a specific keyword.
- **Traditional Approach**: Using a simple keyword search algorithm to flag emails containing words like 'urgent' or 'immediate'.
- **Machine Learning Approach**: Training a complex NLP model to understand the context and categorize emails.
- **Analysis**:
  - *Complexity*: ML adds unnecessary complexity for a task solvable with basic programming.
  - *Resource Efficiency*: ML requires more data, computational power, and maintenance.

Scenario: A Simple Decision Rule

- **Context**: Implementing a system to categorize emails as 'urgent' or 'non-urgent' based on a specific keyword.
- **Traditional Approach**: Using a simple keyword search algorithm to flag emails containing words like 'urgent' or 'immediate'.
- **Machine Learning Approach**: Training a complex NLP model to understand the context and categorize emails.
- **Analysis**:
  - *Complexity*: ML adds unnecessary complexity for a task solvable with basic programming.
  - *Resource Efficiency*: ML requires more data, computational power, and maintenance.
  - *Practicality*: A simple keyword-based approach is more straightforward, easier to implement, and maintain.

Scenario: A Simple Decision Rule

- **Context**: Implementing a system to categorize emails as 'urgent' or 'non-urgent' based on a specific keyword.
- **Traditional Approach**: Using a simple keyword search algorithm to flag emails containing words like 'urgent' or 'immediate'.
- **Machine Learning Approach**: Training a complex NLP model to understand the context and categorize emails.
- **Analysis**:
  - *Complexity*: ML adds unnecessary complexity for a task solvable with basic programming.
  - *Resource Efficiency*: ML requires more data, computational power, and maintenance.
  - *Practicality*: A simple keyword-based approach is more straightforward, easier to implement, and maintain.
  - **Conclusion**: In this case, a basic keyword search is more effective than applying an ML solution.

Regex Utilization: Detecting Urgent Emails

Python Implementation for Simple Pattern Matching

```
import re
def is_urgent(email_content, words):
    words = '|'.join(words)
    pattern = r'\\b(?:{})\\b'.format(words)
    if re.search(pattern, email_content, re.IGNORECASE):
        return "Urgent Email"
    return "Non-Urgent Email"

email = "Please review this document ASAP."
words = ['urgent', 'asap', 'immediate']
print(is_urgent(email))
```

Outcome: Efficiently identifies emails with urgent keywords.

Speed Comparison: Regex vs. Scikit-Learn

Inference Time Comparison

- **Task**: Classify emails as 'urgent' or 'non-urgent'.
- **Methods**:
  - Regex-based pattern matching.
  - A typical NLP classifier from scikit-learn. Needs data to train.
- **Performance Metrics**:
  - *Time Taken for Inference*:
    - Regex: Generally $\sim s$ for 10k mails.
    - Scikit-Learn: Between *10-100s* for 10k mails, may be larger.
  - *Efficiency*: Regex is often faster for simple pattern matching tasks.

Conclusion: For simple keyword-based tasks, Regex can be significantly faster and more resource-efficient than a full ML model. You can iterate fast to reach decent results.

spaCy Rule-Based Matching with POS Tagging

Note: Let's say we consider urgent only if action is needed. Action generally means a verb is present after one keyword.

Part 1: define matcher object

```
import spacy
from spacy.matcher import Matcher

# Load spaCy model
nlp = spacy.load("en_core_web_sm")

# Initialize Matcher
matcher = Matcher(nlp.vocab)
```

Part 2: Define the pattern

```
pattern = [{'TEXT':
        {'REGEX':'(?:urgent|asap)'}},
           {'POS':'PUNCT',
            'OP': '?'},
           {'POS':'VERB'}]
matcher.add("URGENT_ACTION_PATTERN",
            [pattern])
```

spaCy Rule-Based Matching with POS Tagging

```
# Process text
doc = nlp("It is urgent: please review.")
# Apply matcher to doc
matches = matcher(doc)

for match_id, start, end in matches:
    matched_span = doc[start:end]
    print(matched_span.text)
```

Note: And we could add much more rules.

Enhancing Rule-Based Matching with POS Tagging

Applications and Advantages

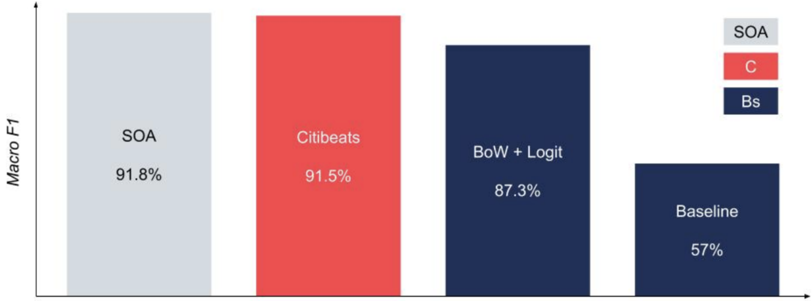
- **Precision**: Enhances pattern matching by considering word types and roles in sentences for more accurate entity and phrase recognition.
- **Versatility**: Enables the definition of complex patterns, facilitating a wider range of linguistic analyses.
- **Depth**: Offers deeper insights and more nuanced text analysis through contextual understanding.

Conclusion

- Rule-based matching with spaCy significantly improves the precision and depth of text analysis.

Compare SOA, Baseline & Random

Put perspective in results ! For instance a classification with 3 classes.



Limitations of Rule-Based Systems in NLP

- **Static Rules**

Limitations of Rule-Based Systems in NLP

- **Static Rules**
  - Require updates

Limitations of Rule-Based Systems in NLP

- **Static Rules**
  - Require updates
  - bootstrapping methods (Gupta et al., 2014) can help evolve rules.

Limitations of Rule-Based Systems in NLP

- **Static Rules**
  - Require updates
  - bootstrapping methods (Gupta et al., 2014) can help evolve rules.
- **Multilingual Scalability**

Limitations of Rule-Based Systems in NLP

- **Static Rules**
  - Require updates
  - bootstrapping methods (Gupta et al., 2014) can help evolve rules.
- **Multilingual Scalability**
  - Challenging for multiple languages

Limitations of Rule-Based Systems in NLP

- **Static Rules**
  - Require updates
  - bootstrapping methods (Gupta et al., 2014) can help evolve rules.
- **Multilingual Scalability**
  - Challenging for multiple languages
  - combining with translation tools and embedding alignment (Dou et al., 2016) can improve effectiveness.

Limitations of Rule-Based Systems in NLP

- **Static Rules**
  - Require updates
  - bootstrapping methods (Gupta et al., 2014) can help evolve rules.
- **Multilingual Scalability**
  - Challenging for multiple languages
  - combining with translation tools and embedding alignment (Dou et al., 2016) can improve effectiveness.
- **Too many rules**

Limitations of Rule-Based Systems in NLP

- **Static Rules**
  - Require updates
  - bootstrapping methods (Gupta et al., 2014) can help evolve rules.
- **Multilingual Scalability**
  - Challenging for multiple languages
  - combining with translation tools and embedding alignment (Dou et al., 2016) can improve effectiveness.
- **Too many rules**
  - Specialized rules may quickly become outdated

Limitations of Rule-Based Systems in NLP

- **Static Rules**
  - Require updates
  - bootstrapping methods (Gupta et al., 2014) can help evolve rules.
- **Multilingual Scalability**
  - Challenging for multiple languages
  - combining with translation tools and embedding alignment (Dou et al., 2016) can improve effectiveness.
- **Too many rules**
  - Specialized rules may quickly become outdated
  - continuous updates and monitoring are expensive.

Limitations of Rule-Based Systems in NLP

- **Static Rules**
  - Require updates
  - bootstrapping methods (Gupta et al., 2014) can help evolve rules.
- **Multilingual Scalability**
  - Challenging for multiple languages
  - combining with translation tools and embedding alignment (Dou et al., 2016) can improve effectiveness.
- **Too many rules**
  - Specialized rules may quickly become outdated
  - continuous updates and monitoring are expensive.

Limitations of Rule-Based Systems in NLP

- **Static Rules**
  - Require updates
  - bootstrapping methods (Gupta et al., 2014) can help evolve rules.
- **Multilingual Scalability**
  - Challenging for multiple languages
  - combining with translation tools and embedding alignment (Dou et al., 2016) can improve effectiveness.
- **Too many rules**
  - Specialized rules may quickly become outdated
  - continuous updates and monitoring are expensive.

Once we have a strong baseline: We may think about Machine Learning !

Bag Of Words

Introduction to Bag of Words

- **What is Bag of Words (BoW)?**

Introduction to Bag of Words

- **What is Bag of Words (BoW)?**
  - A simple and foundational text representation technique in NLP.

Introduction to Bag of Words

- **What is Bag of Words (BoW)?**
  - A simple and foundational text representation technique in NLP.
  - Represents text data as a 'bag' (multiset) of words without considering grammar or word order but keeping multiplicity.

Introduction to Bag of Words

- **What is Bag of Words (BoW)?**
  - A simple and foundational text representation technique in NLP.
  - Represents text data as a 'bag' (multiset) of words without considering grammar or word order but keeping multiplicity.
- **How does it work?**

Introduction to Bag of Words

- **What is Bag of Words (BoW)?**
  - A simple and foundational text representation technique in NLP.
  - Represents text data as a 'bag' (multiset) of words without considering grammar or word order but keeping multiplicity.
- **How does it work?**
  - Texts are converted into a fixed-length vector of numbers.

Introduction to Bag of Words

- **What is Bag of Words (BoW)?**
  - A simple and foundational text representation technique in NLP.
  - Represents text data as a 'bag' (multiset) of words without considering grammar or word order but keeping multiplicity.
- **How does it work?**
  - Texts are converted into a fixed-length vector of numbers.
  - Each unique word in the text corpus corresponds to a feature (vector element).

Introduction to Bag of Words

- **What is Bag of Words (BoW)?**
  - A simple and foundational text representation technique in NLP.
  - Represents text data as a 'bag' (multiset) of words without considering grammar or word order but keeping multiplicity.
- **How does it work?**
  - Texts are converted into a fixed-length vector of numbers.
  - Each unique word in the text corpus corresponds to a feature (vector element).
  - The frequency or presence of each word is then used to fill the vector.

Introduction to Bag of Words

- **What is Bag of Words (BoW)?**
  - A simple and foundational text representation technique in NLP.
  - Represents text data as a 'bag' (multiset) of words without considering grammar or word order but keeping multiplicity.
- **How does it work?**
  - Texts are converted into a fixed-length vector of numbers.
  - Each unique word in the text corpus corresponds to a feature (vector element).
  - The frequency or presence of each word is then used to fill the vector.

Introduction to Bag of Words

- **What is Bag of Words (BoW)?**
  - A simple and foundational text representation technique in NLP.
  - Represents text data as a 'bag' (multiset) of words without considering grammar or word order but keeping multiplicity.
- **How does it work?**
  - Texts are converted into a fixed-length vector of numbers.
  - Each unique word in the text corpus corresponds to a feature (vector element).
  - The frequency or presence of each word is then used to fill the vector.

Note: BoW is often the first step in feature extraction for NLP tasks.

Term Frequency in Bag of Words

- **Term Frequency (TF)**

Term Frequency in Bag of Words

- **Term Frequency (TF)**
  - **Definition**: In the context of Bag of Words, Term Frequency measures how frequently a term occurs in a document.

Term Frequency in Bag of Words

- **Term Frequency (TF)**
  - **Definition**: In the context of Bag of Words, Term Frequency measures how frequently a term occurs in a document.
  - **Formal Representation**:

Term Frequency in Bag of Words

- **Term Frequency (TF)**
  - **Definition**: In the context of Bag of Words, Term Frequency measures how frequently a term occurs in a document.
  - **Formal Representation**:
    - Let d be a document in a corpus D.

Term Frequency in Bag of Words

- **Term Frequency (TF)**
  - **Definition**: In the context of Bag of Words, Term Frequency measures how frequently a term occurs in a document.
  - **Formal Representation**:
    - Let d be a document in a corpus D.
    - Let w be a term (word) in document d.

Term Frequency in Bag of Words

- **Term Frequency (TF)**
  - **Definition**: In the context of Bag of Words, Term Frequency measures how frequently a term occurs in a document.
  - **Formal Representation**:
    - Let d be a document in a corpus D.
    - Let w be a term (word) in document d.
    - The term frequency $TF(w, d)$ is calculated as:

$$TF(w, d) = \text{Number of times word } w \text{ appears in document } d$$

Term Frequency in Bag of Words

- **Term Frequency (TF)**
  - **Definition**: In the context of Bag of Words, Term Frequency measures how frequently a term occurs in a document.
  - **Formal Representation**:
    - Let d be a document in a corpus D.
    - Let w be a term (word) in document d.
    - The term frequency $TF(w, d)$ is calculated as:

$$TF(w, d) = \text{Number of times word } w \text{ appears in document } d$$

- **Significance in Bag of Words**

Term Frequency in Bag of Words

- **Term Frequency (TF)**
  - **Definition**: In the context of Bag of Words, Term Frequency measures how frequently a term occurs in a document.
  - **Formal Representation**:
    - Let d be a document in a corpus D.
    - Let w be a term (word) in document d.
    - The term frequency $TF(w, d)$ is calculated as:

$$TF(w, d) = \text{Number of times word } w \text{ appears in document } d$$

- **Significance in Bag of Words**
  - TF is a fundamental concept in converting text to numerical format in BoW, representing the importance of each term in the document.

Term Frequency in Bag of Words

- **Term Frequency (TF)**
  - **Definition**: In the context of Bag of Words, Term Frequency measures how frequently a term occurs in a document.
  - **Formal Representation**:
    - Let d be a document in a corpus D.
    - Let w be a term (word) in document d.
    - The term frequency $TF(w, d)$ is calculated as:

$$TF(w, d) = \text{Number of times word } w \text{ appears in document } d$$

- **Significance in Bag of Words**
  - TF is a fundamental concept in converting text to numerical format in BoW, representing the importance of each term in the document.
  - It's a simple way to quantify and compare the occurrence of terms across different documents in a corpus.

Term Frequency Bag of Words with Scikit-Learn

Generating a TF BoW Model

Using Python's scikit-learn library to vectorize text data.

Python Code

```
from sklearn.feature_extraction.text import CountVectorizer

# Example sentences
sentences = ["The quick brown fox jumps over the lazy dog",
             "Never jump over the lazy dog quickly",
             "The fox is quick and brown"]

# Initialize CountVectorizer
vectorizer = CountVectorizer()

# Fit and transform the sentences
BoW_matrix = vectorizer.fit_transform(sentences)
print(BoW_matrix.toarray())
```

Term Frequency Bag of Words with Scikit-Learn

Python Code, results

```
sentences = ["The quick brown fox jumps over the lazy dog",
             "Never jump over the lazy dog quickly",
             "The fox is quick and brown"]

Vocabulary: {'the': 12, 'quick': 10, 'brown': 1, 'fox': 3, 'jumps': 6,
              'over': 9,  'lazy': 7, 'dog': 2, 'never': 8,  'jump': 5,
              'quickly': 11, 'is': 4, 'and': 0}

tf_matrix:[[0 1 1 1 0 0 1 1 0 1 1 0 2]
           [0 0 1 0 0 1 0 1 1 1 0 1 1]
           [1 1 0 1 1 0 0 0 0 1 0 1]]
```

- The output matrix represents the term frequencies of each word in the sentences.
- Each column corresponds to a unique word in the combined sentences.
- Rows represent each sentence's word frequency vector.

Limitations of Term Frequency BoW

- **Overemphasis on Frequent Words**:

Limitations of Term Frequency BoW

- **Overemphasis on Frequent Words**:
  - Common words like 'the' and 'is' may appear frequently but offer little value in understanding the unique context of each document.

Limitations of Term Frequency BoW

- **Overemphasis on Frequent Words**:
  - Common words like 'the' and 'is' may appear frequently but offer little value in understanding the unique context of each document.
  - *Example*: In our sentences, 'the' and 'quick' are frequent but not necessarily informative.

Limitations of Term Frequency BoW

- **Overemphasis on Frequent Words**:
  - Common words like 'the' and 'is' may appear frequently but offer little value in understanding the unique context of each document.
  - *Example*: In our sentences, 'the' and 'quick' are frequent but not necessarily informative.
- **Ignoring Word Importance Across Documents**:

Limitations of Term Frequency BoW

- **Overemphasis on Frequent Words**:
  - Common words like 'the' and 'is' may appear frequently but offer little value in understanding the unique context of each document.
  - *Example*: In our sentences, 'the' and 'quick' are frequent but not necessarily informative.
- **Ignoring Word Importance Across Documents**:
  - TF BoW counts words in each document independently, not accounting for their importance or rarity across the entire document set.

Limitations of Term Frequency BoW

- **Overemphasis on Frequent Words**:
  - Common words like 'the' and 'is' may appear frequently but offer little value in understanding the unique context of each document.
  - *Example*: In our sentences, 'the' and 'quick' are frequent but not necessarily informative.
- **Ignoring Word Importance Across Documents**:
  - TF BoW counts words in each document independently, not accounting for their importance or rarity across the entire document set.
  - *Example*: Words like 'fox' and 'dog', which might be key to understanding the specific content, are treated the same as common words.

Term Frequency-Inverse Document Frequency (TF-IDF)

- **What is TF-IDF?**
  It enhances the basic TF by considering not only the frequency of a word in a single document but also its frequency across multiple documents.

Term Frequency-Inverse Document Frequency (TF-IDF)

- **What is TF-IDF?**
  It enhances the basic TF by considering not only the frequency of a word in a single document but also its frequency across multiple documents.
- **Formal Definition**

Term Frequency-Inverse Document Frequency (TF-IDF)

- **What is TF-IDF?**

  It enhances the basic TF by considering not only the frequency of a word in a single document but also its frequency across multiple documents.
- **Formal Definition**
  - Let w be a word, d a document, and D the corpus of documents.

Term Frequency-Inverse Document Frequency (TF-IDF)

- **What is TF-IDF?**
  It enhances the basic TF by considering not only the frequency of a word in a single document but also its frequency across multiple documents.
- **Formal Definition**
  - Let w be a word, d a document, and D the corpus of documents.
  - The TF-IDF value is calculated as:

$$\text{TF-IDF}(w, d, D) = TF(w, d) \times IDF(w, D)$$

Term Frequency-Inverse Document Frequency (TF-IDF)

- **What is TF-IDF?**
  It enhances the basic TF by considering not only the frequency of a word in a single document but also its frequency across multiple documents.
- **Formal Definition**
  - Let w be a word, d a document, and D the corpus of documents.
  - The TF-IDF value is calculated as:

$$\text{TF-IDF}(w, d, D) = TF(w, d) \times IDF(w, D)$$

  - Where $IDF(w, D)$ (Inverse Document Frequency) is defined as:

$$IDF(w, D) = \log\left(\frac{\text{Total number of documents in } D}{\text{Number of documents containing word } w}\right)$$

Term Frequency-Inverse Document Frequency (TF-IDF)

Differences from TF BoW

- **Word Significance**: TF-IDF decreases the weight of words that occur frequently across many documents (common words), emphasizing words that are unique to specific documents.

Term Frequency-Inverse Document Frequency (TF-IDF)

Differences from TF BoW

- **Word Significance**: TF-IDF decreases the weight of words that occur frequently across many documents (common words), emphasizing words that are unique to specific documents.
- **Contextual Importance**: Unlike TF, which treats all terms equally, TF-IDF provides a way to assess the relevance of terms in the context of the entire corpus.

Term Frequency-Inverse Document Frequency (TF-IDF)

Differences from TF BoW

- **Word Significance**: TF-IDF decreases the weight of words that occur frequently across many documents (common words), emphasizing words that are unique to specific documents.
- **Contextual Importance**: Unlike TF, which treats all terms equally, TF-IDF provides a way to assess the relevance of terms in the context of the entire corpus.

Term Frequency-Inverse Document Frequency (TF-IDF)

Differences from TF BoW

- **Word Significance**: TF-IDF decreases the weight of words that occur frequently across many documents (common words), emphasizing words that are unique to specific documents.
- **Contextual Importance**: Unlike TF, which treats all terms equally, TF-IDF provides a way to assess the relevance of terms in the context of the entire corpus.

TF-IDF is widely used in information retrieval and text mining to reflect how important a word is to a document in a collection.

TF-IDF with Scikit-Learn

Generating a TF-IDF Model

Using Python's scikit-learn library to apply TF-IDF vectorization to data.

Python Code

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Example sentences
sentences = ["The quick brown fox jumps over the lazy dog",
             "Never jump over the lazy dog quickly",
             "The fox is quick and brown"]

# Initialize TfidfVectorizer
vectorizer = TfidfVectorizer()

# Fit and transform the sentences
tfidf_matrix = vectorizer.fit_transform(sentences)

# Print the resulting matrix
print(tfidf_matrix.toarray())
```

TF-IDF with Scikit-Learn

Python Code, results

```
sentences = ["The quick brown fox jumps over the lazy dog",
             "Never jump over the lazy dog quickly",
             "The fox is quick and brown"]

Vocabulary: {'the': 12, 'quick': 10, 'brown': 1, 'fox': 3, 'jumps': 6,
              'over': 9,  'lazy': 7, 'dog': 2, 'never': 8,  'jump': 5,
              'quickly': 11, 'is': 4, 'and': 0}

tf_matrix:[[0 1 1 1 0 0 1 1 0 1 1 0 2]
           [0 0 1 0 0 1 0 1 1 1 0 1 1]
           [1 1 0 1 1 0 0 0 0 0 1 0 1]]
tf_idf_matrix:
[[0.  0.3 0.3 0.3 0.  0.  0.4 0.3 0.  0.3 0.3 0.  0.5]
 [0.  0.  0.3 0.  0.  0.4 0.  0.3 0.4 0.3 0.  0.4 0.3]
 [0.5 0.4 0.  0.4 0.5 0.  0.  0.  0.  0.  0.4 0.  0.3]]
```

Limitations of TF-IDF and Motivation for BM25

- **Term Saturation**

Limitations of TF-IDF and Motivation for BM25

- **Term Saturation**
  - Overemphasis on frequent terms due to linear term frequency assumption.

Limitations of TF-IDF and Motivation for BM25

- **Term Saturation**
  - Overemphasis on frequent terms due to linear term frequency assumption.
  - *E.g.*, "space" dominates over "exoplanets" in space-related texts.

Limitations of TF-IDF and Motivation for BM25

- **Term Saturation**
  - Overemphasis on frequent terms due to linear term frequency assumption.
  - *E.g.*, "space" dominates over "exoplanets" in space-related texts.
- **Document Length Bias**

Limitations of TF-IDF and Motivation for BM25

- **Term Saturation**
  - Overemphasis on frequent terms due to linear term frequency assumption.
  - *E.g.*, "space" dominates over "exoplanets" in space-related texts.
- **Document Length Bias**
  - Favors longer documents, overlooking shorter content's relevance.

Limitations of TF-IDF and Motivation for BM25

- **Term Saturation**
  - Overemphasis on frequent terms due to linear term frequency assumption.
  - *E.g.*, "space" dominates over "exoplanets" in space-related texts.
- **Document Length Bias**
  - Favors longer documents, overlooking shorter content's relevance.
  - *E.g.*, "Quantum computing" appears more relevant in longer papers.

Limitations of TF-IDF and Motivation for BM25

- **Term Saturation**
  - Overemphasis on frequent terms due to linear term frequency assumption.
  - *E.g.*, "space" dominates over "exoplanets" in space-related texts.
- **Document Length Bias**
  - Favors longer documents, overlooking shorter content's relevance.
  - *E.g.*, "Quantum computing" appears more relevant in longer papers.
- **Query-Document Mismatch**

Limitations of TF-IDF and Motivation for BM25

- **Term Saturation**
  - Overemphasis on frequent terms due to linear term frequency assumption.
  - *E.g.*, "space" dominates over "exoplanets" in space-related texts.
- **Document Length Bias**
  - Favors longer documents, overlooking shorter content's relevance.
  - *E.g.*, "Quantum computing" appears more relevant in longer papers.
- **Query-Document Mismatch**
  - Lacks specific tuning for matching queries with document relevance.

Limitations of TF-IDF and Motivation for BM25

- **Term Saturation**
  - Overemphasis on frequent terms due to linear term frequency assumption.
  - *E.g.*, "space" dominates over "exoplanets" in space-related texts.
- **Document Length Bias**
  - Favors longer documents, overlooking shorter content's relevance.
  - *E.g.*, "Quantum computing" appears more relevant in longer papers.
- **Query-Document Mismatch**
  - Lacks specific tuning for matching queries with document relevance.
  - *E.g.*, "Python" ambiguously interpreted in diverse contexts.

Limitations of TF-IDF and Motivation for BM25

- **Term Saturation**
  - Overemphasis on frequent terms due to linear term frequency assumption.
  - *E.g.*, "space" dominates over "exoplanets" in space-related texts.
- **Document Length Bias**
  - Favors longer documents, overlooking shorter content's relevance.
  - *E.g.*, "Quantum computing" appears more relevant in longer papers.
- **Query-Document Mismatch**
  - Lacks specific tuning for matching queries with document relevance.
  - *E.g.*, "Python" ambiguously interpreted in diverse contexts.

Limitations of TF-IDF and Motivation for BM25

- **Term Saturation**
  - Overemphasis on frequent terms due to linear term frequency assumption.
  - *E.g.*, "space" dominates over "exoplanets" in space-related texts.
- **Document Length Bias**
  - Favors longer documents, overlooking shorter content's relevance.
  - *E.g.*, "Quantum computing" appears more relevant in longer papers.
- **Query-Document Mismatch**
  - Lacks specific tuning for matching queries with document relevance.
  - *E.g.*, "Python" ambiguously interpreted in diverse contexts.

Conclusion: TF-IDF's simplistic approach leads to biased term

BM25

- **What is BM25?**
  It enhances TF-IDF balancing the term frequency with document length and query-document relevance.

BM25

- **What is BM25?**
  It enhances TF-IDF balancing the term frequency with document length and query-document relevance.
- **Formal Definition**

BM25

- **What is BM25?**

  It enhances TF-IDF balancing the term frequency with document length and query-document relevance.

- **Formal Definition**
  - Given a sentence W containing terms $w_1, w_2, ..., w_n$, the BM25 score of a document d is:

$$\text{Score}(d, W) = \sum_{i=1}^{n} IDF(w_i) \times \frac{tf(w_i, d) \times (k_1 + 1)}{tf(w_i, D) + k_1 \times (1 - b + b \times \frac{|d|}{\text{avgdl}})}$$

BM25

- **What is BM25?**
  It enhances TF-IDF balancing the term frequency with document length and query-document relevance.
- **Formal Definition**
  - Given a sentence W containing terms $w_1, w_2, ..., w_n$, the BM25 score of a document d is:
    $$\text{Score}(d, W) = \sum_{i=1}^{n} IDF(w_i) \times \frac{tf(w_i, d) \times (k_1 + 1)}{tf(w_i, D) + k_1 \times (1 - b + b \times \frac{|d|}{\text{avgdl}})}$$
  - Where $tf(w_i, d)$ is w_i's frequency in d, $|d|$ is the length of the document, and avgdl is the average document length in the corpus. k_1 and b are free parameters, usually chosen empirically.

BM25

- **What is BM25?**

  It enhances TF-IDF balancing the term frequency with document length and query-document relevance.

- **Formal Definition**
  - Given a sentence W containing terms $w_1, w_2, ..., w_n$, the BM25 score of a document d is:

    $$\text{Score}(d, W) = \sum_{i=1}^{n} IDF(w_i) \times \frac{tf(w_i, d) \times (k_1 + 1)}{tf(w_i, D) + k_1 \times (1 - b + b \times \frac{|d|}{\text{avgdl}})}$$

  - Where $tf(w_i, d)$ is w_i's frequency in d, $|d|$ is the length of the document, and avgdl is the average document length in the corpus. k_1 and b are free parameters, usually chosen empirically.
  - Where $IDF(w_i) = ln(\frac{N - n(w_i) + 0.5}{n(w_i) + 0.5} + 1)$, N is the number of documents and $n(w_i)$ the number of documents containing w_i

Differences with TF-IDF

Key Differences Between BM25 and TF-IDF

- **TF / (TF + k), the backbone of BM25**
  - *Term saturation*: now limited by 1. the higher k the lower it reaches 1.
  - *Document length*: let k depends on length of the document as k $=|d|/$avgdl, the longer the document, the more it will penalize the score. The value of b wgives the speed of the growth.
- **Document Length Normalization**
  - *BM25*: k depends on length of the document as k $=|d|/$avgdl

Conclusion: BM25 addresses several key limitations of TF-IDF, making it more suitable for modern information retrieval systems.

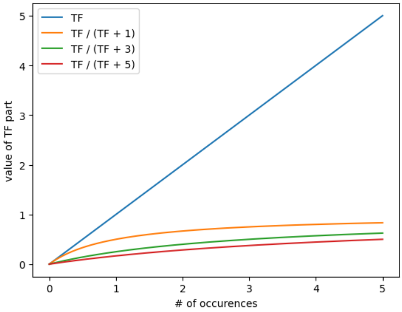
Illustrations of TF / (TF + k)

BM25 Formula

$$\text{Score}(d, W) = \sum_{i=1}^{n} IDF(w_i) \times \frac{f(w_i, d) \times (k_1 + 1)}{f(w_i, D) + k_1 \times (1 - b + b \times \frac{|d|}{\text{avgdl}})}$$

TF-IDF Formula

$$\text{TF-IDF}(w, d, D) = TF(w, d) \times IDF(w, D)$$



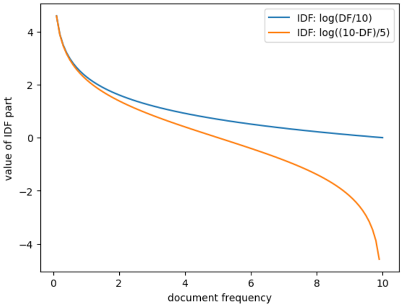
Illustrations of IDFs

BM25 Formula

$$IDF(w_i) = ln(\frac{N - n(w_i) + 0.5}{n(w_i) + 0.5} + 1)$$

TF-IDF Formula

$$IDF(w, D) = \log\left(\frac{\text{Total number of documents in } D}{\text{Number of documents containing word } w}\right)$$



BM25 with Scikit-Learn

Generating a BM25 representation

Using Python's scikit-learn library and some functions to compute BM25.

Python Code

```
from sklearn.feature_extraction.text import CountVectorizer

# Example sentences
sentences = ["The quick brown fox jumps over the lazy dog",
             "Never jump over the lazy dog quickly",
             "The fox is quick and brown"]

# Initialize CountVectorizer
vectorizer = CountVectorizer()

# Fit and transform the sentences
tf_matrix = vectorizer.fit_transform(sentences)
```

BM25 with Scikit-Learn

```
from math import log
import numpy as np

def compute_idf(corpus):
    N = len(corpus)
    idf_dict = {}
    for document in corpus:
        for term in set(document.split()):
            idf_dict[term] = idf_dict.get(term, 0) + 1
    for term, count in idf_dict.items():
        idf_dict[term] = log(N / float(count))
    return idf_dict

def bm25(tf, idf, avgdl, dl, b=0.75, k1=1.5):
    return idf * (tf * (k1 + 1))/(tf + k1 * (1 - b + b * (dl / avgdl)))
```

BM25 with Scikit-Learn

```
# Calculate IDF
idf_dict = compute_idf(sentences)
avgdl = np.mean([len(doc.split()) for doc in sentences])

# Calculate BM25
bm25_matrix = np.zeros((len(sentences), len(terms)))
for i, sentence in enumerate(sentences):
    dl = len(sentence.split())
    for j, term in enumerate(terms):
        tf = X[i, j]
        idf = idf_dict.get(term, 0)
        bm25_matrix[i, j] = bm25(tf, idf, avgdl, dl)
```

BM25 with Scikit-Learn

Python Code, results

```
sentences = ["The quick brown fox jumps over the lazy dog",
             "Never jump over the lazy dog quickly",
             "The fox is quick and brown"]

Vocabulary: {'the': 12, 'quick': 10, 'brown': 1, 'fox': 3, 'jumps': 6,
              'over': 9,  'lazy': 7, 'dog': 2, 'never': 8,  'jump': 5,
              'quickly': 11, 'is': 4, 'and': 0}

tf_idf_matrix:
[[0.  0.3 0.3 0.3 0.  0.  0.4 0.3 0.  0.3 0.3 0.  0.5]
 [0.  0.  0.3 0.  0.  0.4 0.  0.3 0.4 0.3 0.  0.4 0.3]
 [0.5 0.4 0.  0.4 0.5 0.  0.  0.  0.  0.  0.4 0.  0.3]]

bm25_matrix:
[[0.  0.4 0.4 0.4 0.  0.  1.  0.4 0.  0.4 0.4 0.  0.5]
 [0.  0.  0.4 0.  0.  1.1 0.  0.4 0.  0.4 0.  1.1 0.4]
 [1.2 0.4 0.  0.4 1.2 0.  0.  0.  0.  0.  0.4 0.  0.4]]
```

Limitations of TF-IDF

TF-IDF Limitations

- **Term Frequency Bias**: Overemphasizes words that appear frequently, potentially overshadowing rare yet significant terms.
- **Document Length**: Fails to normalize for document length, potentially biasing towards longer documents.
- **Lack of Context and Semantics**: Treats each word independently without considering context or word meanings.

Limitations of BM25

BM25 Limitations

- **Parameter Sensitivity**: The effectiveness of BM25 depends on the tuning of its parameters k_1 and b, which may not be straightforward.
- **Still Context-Agnostic**: Like TF-IDF, BM25 does not account for word order, semantics, or the overall context of the query or document.
- **Complexity in Large Scale Applications**: Computationally more complex than TF-IDF, especially for very large document collections.

Note: Both methods, while foundational in information retrieval, have been partly superseded by more advanced NLP techniques that better understand context and semantics, like word embeddings and neural network models.

QA

Open Discussion

- Feel free to ask questions or share your thoughts about today's topics.
- Any insights, experiences, or perspectives you'd like to discuss are welcome.

Summary of Key Takeaways

- **Metrics and Evaluation**: Discussed various metrics and evaluation strategies to judge model quality.
- **Baselines in NLP**: Emphasized the importance of establishing baselines for comparison and model assessment.
- **Term Frequency's Role**: Understood its significance in text representation and limitations in context and semantics.
- **TF-IDF and BM25**: Explored their concepts, applications, and limitations.
- **BM25's Advantages**: Improved handling of term frequency and document length, but with its own set of challenges.
- **Evolving Landscape of NLP**: Recognized advancements beyond TF-IDF and BM25 towards more context-aware and semantic approaches in NLP.