# Week 4 - Weighting Schemes

## Recap

In previous classes, we've identified a significant challenge: how to assign a weight to a term in a document. The weight of a term depends on its occurrences in the document, and we aim to compute a score between a query term $t$ and a document $d$, based on this weight. A standard approach is to assign a weight as a function of the number of occurrences of term $t$ in document $d$. Typically, we use a **bag of words model**, where the actual ordering of terms in a document is ignored, but the number of occurrences is important.

For instance, the document "**Mary is quicker than John**" is treated as identical to "**John is quicker than Mary**" in this model.

### Term Frequency

Using a term's raw frequency presents a problem: all terms are treated equally when assessing a document's relevance to a query. However, not all terms are equally discriminative. To address this, we use global features **(e.g., collection frequency or document frequency)** to adjust the weight of the term frequency. For instance, stopwords like "**the**" or "**and**" appear frequently but don't carry much meaning.

We looked at the **vector space model**, where documents and queries are represented as vectors:

$$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{n,j})$$
$$\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{n,q})$$

The similarity between a document and a query can be measured using **cosine similarity**:

$$\text{sim}(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}||\vec{q}|}$$

$$\text{sim}(d_j, q) = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$$

This metric essentially sums up the weights of co-occurring terms between a document and query, with normalization accounting for document length.

### TF-IDF

We also explored **Term Frequency - Inverse Document Frequency (TF-IDF)**. The **TF** component rewards frequently occurring terms within a document, while the **IDF** component penalizes terms that occur in many documents.

$$w_{i,j} = f_{i,j} \times \log\left(\frac{N}{n_i}\right)$$

**Where:**

- $f_{i,j}$ is some function of the frequency of term $t_i$ in document $d_j$
- $N$ is the number of documents in the collection
- $n_i$ is the number of documents in the collection that contain term $t_i$

This method provides a more reasonable weighting scheme, adjusting for both term frequency and document frequency.

# Text Properties

Not all words contribute equally to capturing the meaning of a document. Certain words, such as stopwords, appear frequently but offer little value for identifying relevant documents

Additionally, we're interested in how quickly the lexicon or vocabulary grows. For example, as we add documents, what is the probability that the index will grow? If we add a new document, how many new words will it contain? If we start with no documents and add the first one, every word encountered will be new. When we add a second document, some words will be new, but others are likely to have appeared in the first document. This trend continues as we add the 3rd, 1000th, or 1,000,000th document, with the probability of encountering new words diminishing considerably.

If we can estimate how fast our vocabulary is likely to grow, we can gain insights into how fast the index will grow. For instance, in a collection of thousands of documents with an average of 600 words per document, we could compute a bound on the index size. This estimation could also be used for index compression strategies.

## Word Frequency Distribution

Word distributions are oftern **heavy-tailed** of follow **Zipf's law**, meaning:

- A few words are very common (e.g., "the", "of").
- Many words are rare (e.g., over half of the words in a corpus may appear only once)

| Frequent Word | Number of Occurrences | Percentage of Total |
|---|---|---|
| the | 7,398,934 | 5.9 |
| of | 3,893,790 | 3.1 |
| to | 3,364,653 | 2.7 |
| and | 3,320,687 | 2.6 |
| in | 2,311,785 | 1.8 |
| is | 1,559,147 | 1.2 |
| for | 1,313,561 | 1.0 |
| The | 1,144,860 | 0.9 |
| that | 1,066,503 | 0.8 |
| said | 1,027,713 | 0.8 |

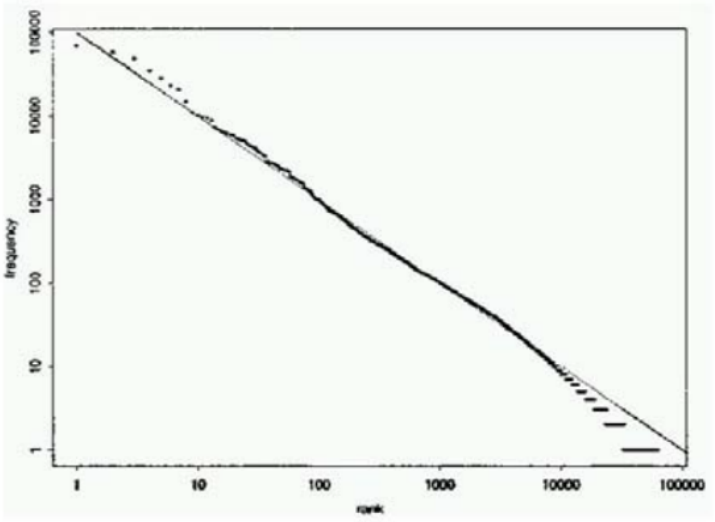
Frequencies from 336,310 documents in the 1GB TREC Volume 3 Corpus
125,720,891 total word occurrences; 508,209 unique words

## Zipf's Law

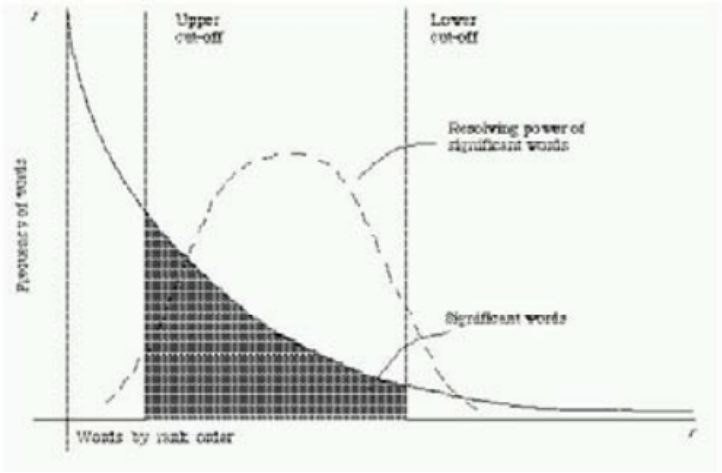
**Zipf's Law** models word frequency distribution in a text corpus. It states that the product of a word's **frequency** $f$ and its **rank** $r$ in the frequency list is approximately constant:

$$f \times r = \text{constant}$$

This represents a **power law**, which appears as a straight line on a log-log plot. Essentially, if you multiply a term's frequency by its rank, the result approximates a constant value.




## Luhn - resolving power




**Luhn's theory** proposes that there is an optimal range of word frequencies that provide the most discriminative power for information retrieval.
**Luuhn's Curve** The graph above visualizes Luhn's theory for determining the resolving power of words:

**Key insights:**

- **X-axis (Words by Rank Order):** This axis orders words from most frequent to least frequent
- **Y-Axis (Times Appeared):** This axis shows the frequency of each word's occurrence.

Luhn suggests that:

- **Upper Cut-Off:** Words that are too frequent (left side of the curve) are considered noise and offer little value for differentiating between documents. These include common words like "the", "of", etc.
- **Lower Cut-Off:** Words that are too rare (the right side of the curve) are unlikely to provide useful distinguishing power, as they may only appear in a handful of documents or queries.
- **Significant Words:** In the middle of the curve lies the sweet spot - words that occur frequent enough to be informative but not so frequently to be useless. These words contribute the most to resolving the relevance of a document to a query.

By focusing on the words between the upper and lower cut-off points, we can improve the efficiency and accuracy of information retrieval systems and we kind of already sort of do this.

- **Stop Word Removal:** Words to the left (most frequent) are typically filtered out to reduce noise in the dataset.
- **Term Weighting:** Words in the middle of the curve are given more weight, especially in models like TF-IDF, where term frequency and document frequency are key factors.

## Vocabulary Growth

As the number of documents in a collection increases, the vocabulary size grows. However, as we add more documents, the likelihood of encountering new terms decreases. Initially, when a collection is small, adding a document introduces many new terms. But as the collection grows, each additional document contributes fewer new terms to the vocabulary.

**Diminishing Growth of New Terms**
In certain cases, the vocabulary size is not strictly bounded and never reaches zero growth for several reasons:

- **Identifiers:** When creating documents (e.g., at work or online), identifiers such as project names or technical terms may be unique to that document.
- **Misspellings and Typos:** Misspelled words or typing errors also introduce new terms
- **Proper Nouns:** New words can come from names of people, places, chemicals, drugs, or medicines, which may not have appeared in the collection previously

Due to these factors, the growth of the vocabulary tends to slow down but never fully stops.

## Predicting Vocabulary Growth

There has been extensive research on predicting how a vocabulary grows. One of the well-known models is **Heaps' Law**, which provides an approximation for vocabulary size growth.

**Heap's Law** defines the relationship between the size of the vocabulary $V$ and the size of the document collection $n$:

$$V = K \cdot n^{\beta}, \quad 0 < \beta < 1$$

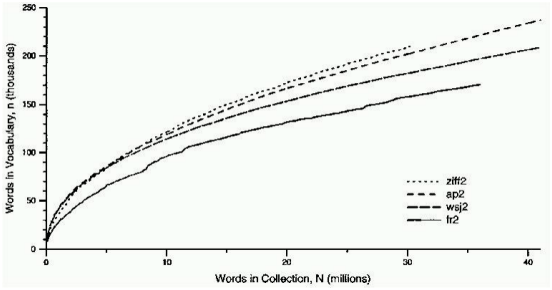
**Where:**

- $\boldsymbol{V}$ is the vocabulary size (number of unique terms),
- $\boldsymbol{K}$ is a constant,
- $\boldsymbol{n}$ is the total number of word occurrences in the collection,
- $\boldsymbol{\beta}$ is an exponent that typically ranges between 0.4 and 0.6 for natural languages, and for English, it's usually close to 0.5.

As the number of documents in the collection increases, the vocabulary size grows at a sublinear rate (since $0 < \beta < 1$). This means that after a certain point, adding more documents will increase the vocabulary, but at a slower pace.

For instance, if you have a collection of one million documents with an average length of 300 words per document, Heaps' Law allows you to estimate the vocabulary size, which gives insights into the growth of your index.

By using Heaps' Law, you can predict the vocabulary size and determine how much new information each additional document will bring, aiding in tasks like index compression and information retrieval optimization.




# Weighting schemes

We've already explored **TF-IDF** as a common weighting scheme, but it has some inherent limitations:

- **Term Independence**: TF-IDF assumes that terms are independent of each other and ignores word order. This is based on the ***bag of words*** model, where the sequence of words is disregarded. As a result, sentences such as ***"Mary is quicker than John"*** and ***"John is quicker than Mary"*** are treated as identical, even though they clearly convey different meanings in terms of who is faster. By assigning equal weights to all terms, TF-IDF loses some of the sentence's semantics.

## Term Frequency and Logarithmic Scaling

In terms of **term frequency (TF)**, we've traditionally considered the raw frequency of a term within a document. This makes sense to some extent because the more frequently a term appears, the more likely it represents the document's content. For example, if the term "**computer**" appears once in one document and 50 times in another, we're more inclined to believe that the second document is more related to computers.

However, there are diminishing returns to simply using raw counts. Seeing the word "**computer**" once suggests the document might be related to computers, and encountering it again strengthens that belief. But at a certain point—say, after seeing the word 30 times—each additional occurrence doesn't significantly increase our confidence in the topic. The first few occurrences carry more weight than later ones.

#### Logarithmic Term Frequency

One approach to address this is to take the **logarithm of the term frequency**. The idea is to recognize that term frequency should not grow linearly with the number of occurrences but rather reflect diminishing returns as the term appears more often.
The adjusted term frequency weighting can be defined as:

$$w_{i,d} = 1 + \log(t_{f_{i,d}}), \quad \text{if } t_{f_{i,d}} > 0$$

**Where:**

- $t_{f_{i,d}}$ is the term frequency of term $i$ in document $d$
- $w_{i,d}$ is the weight assigned to term $i$ in document $d$.

In cases where a term does not occur in the document, we assign it a weight of 0:

$$w_{i,d} = 0, \quad \text{if } t_{f_{i,d}} = 0$$

**Interpretation**

- The logarithmic function flattens the growth of term frequency weights. Initially, encountering a term increases our belief that the document is about the topic, but after several occurrences, the impact of additional appearances decreases.
- This approach balances the contribution of frequent terms without overemphasizing them. The goal is to capture the intuition that repeated occurrences of a term beyond a certain threshold do not proportionally increase the document's relevance to the term.

## Normalization

**Maximum Term Normalization:** The idea behind **maximum term normalization** is to account for document length when assessing term relevance. If we consider two documents:

- **Document A**: 50 words, with "**computer**" appearing twice.
- **Document B**: 10,000 words, with "**computer**" appearing twice.

Clearly, "**computer**" is more relevant in the shorter **document A** than in the much longer **document B**. Without normalization, these two documents might be treated as equally related to "**computer**" which would be misleading. Normalization ensures that the term's frequency is adjusted relative to the length of the document.

**Maximum term normalization is typically expressed as:**

$$\text{ntf} = a + (1 - a)\frac{t_{f_{i,d}}}{\text{tf}_{\max}(d)}$$

**Where:**

- $t_{f_{i,d}}$ is the term frequency of term $i$ in document $d$,
- $\text{tf}_{\max}(d)$ is the frequency of the most common term in document $d$,
- $a$ is a smoothing factor (with $0 \leq a \leq 1$) used to dampen the impact of the second term.

This formula adjusts the term frequency relative to the maximum frequency of any term in the document, normalizing based on document length.

The **smoothing factor** $a$ allows us to control the extent to which this normalization affects term weights. For example:

- $a = 1$: No normalization is applied (the weight is just the term frequency).
- $a = 0$: Full normalization (weights are strictly based on the relative frequency compared to the most common term).

This idea of **max term normalization** can be applied beyond text processing, such as in image analysis (e.g., scaling based on the most frequent color).

## Criticisms of Maximum Term Normalization

Although maximum term normalization helps in addressing document length variations, it has its limitations and potential pitfalls:

1. **Impact of Stopword Removal**: Stopword removal can affect the distribution of terms, potentially destabilizing the normalization process. For instance, if some documents in the collection have had stopwords removed while others have not, the normalization results may vary unpredictably.
2. **Outliers**: Documents with unusually high frequencies of certain terms (outliers) can distort the normalization process. Such outliers might cause misleading weightings in some cases.

3. **Distribution of Terms**: Documents with a more even distribution of term frequencies should be treated differently from those with skewed distributions. For example, documents dominated by one or two terms will behave differently under this normalization compared to documents with a broader distribution of terms.

4. **Sophisticated Normalization Techniques**: More sophisticated normalization techniques exist and may be more suitable in specific contexts. Will cover these different approaches later

## General Need for Normalization

Normalization is crucial for several reasons:

1. **Higher Frequencies in Longer Documents**: Longer documents tend to repeat words more frequently, but this doesn't necessarily mean they are more relevant to a query. For instance, simply concatenating a document with itself (to create a longer version of the same content) doesn't make the new document more relevant to a query, but without normalization, it would appear more relevant.

2. **Vector Space Model Example**: Consider a document $d$ and a new document $d'$ created by concatenating $d$ with itself. The vector space similarity score between $d'$ and any query $q$ would be higher than between $d$ and $q$, even though $d'$ contains no new information. Without normalization, $\text{sim}(d', q) \geq \text{sim}(d, q)\ \forall q$

3. **Avoiding Redundancy**: In practice, if a user encounters document $d'$ (which is just a repeated version of $d$), they wouldn't find it more useful than the original document. Normalization helps to correct this issue by ensuring that longer documents don't unfairly gain higher relevance scores simply due to length.

Normalization ensures that documents of different lengths are treated fairly, preventing artificially inflated relevance scores due to word repetition.

# Weighting Schemes in Information Retrieval

We have examined several weighting schemes, including Boolean models and TF-IDF. Most modern weighting schemes can be represented in the following general format:

$$\text{sim}(q, d) = \sum_{t \in q \cap d} (\text{ntf}(D) \times \text{gwt}(C) \times \text{qwt}(Q))$$

**Where**:

- ntf($D$) is the normalized term frequency in document $D$,
- gwt($C$) is the global weight of a term across the collection $C$,
- qwt($Q$) is the query weight of a term in query $Q$.

## Explanation

- ntf($D$) is a local measure at the document level, often normalized using logarithmic scaling.
- gwt($C$) is a global feature, such as IDF, that reflects the term's importance across the entire collection.
- qwt($Q$) is the weight assigned to the term in the query. This can be based on frequency, previous queries, or other available information.

Thus, many term weighting schemes can be represented as a product of these three components: normalized term frequency, global weight, and query weight. Numerous approaches to term weighting, document normalization, and global weighting schemes (such as IDF) have been developed and empirically tested across collections.

## BM25/Okapi

One example of such a standard weighting scheme is **BM25**, which can be expressed as:

$$\text{BM25}(Q, D) = \sum_{t \in Q \cap D} \left( \frac{tf_t^D \cdot \log\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right) \cdot tf_t^Q}{tf_t^D + k_1 \cdot \left((1-b) + b \cdot \frac{dl}{dl_{\text{avg}}}\right)} \right)$$

**Where:**

- $tf_t^D$ is the term frequency of term $t$ in document $D$,
- $tf_t^Q$ is the term frequency of term $t$ in query $Q$,
- $N$ is the total number of documents in the collection,
- $df_t$ is the document frequency of term $t$,
- $dl$ is the document length of $D$,
- $dl_{\text{avg}}$ is the average document length in the collection,
- $k_1$ and $b$ are tuning parameters.

**Key Characteristics of BM25**

- It is a standard benchmark weighting scheme with relatively good performance.
- Requires tuning parameters $k_1$ and $b$ to adjust the formula for specific collections.
- The denominator helps normalize term frequency based on document length, using the tuning parameters $k_1$ and $b$ to control the impact of document length and term frequency.

Although we will not delve into the detailed derivation of BM25 here, it fits perfectly into the general template of local term weighting, global term weighting, and query weighting.

## Pivoted Normalization

Another popular weighting scheme is **Pivoted Normalization**, which is defined as:

$$\text{piv}(Q, D) = \sum_{t \in Q \cap D} \left( \frac{1 + \log(1 + \log(tf_t^D))}{(1-s) + s \cdot \frac{dl}{dl_{\text{avg}}}} \times \log\left(\frac{N+1}{df_t}\right) \times tf_t^Q \right)$$

**Where:**

- $tf_t^D$ is the term frequency of term $t$ in document $D$,
- $dl$ is the document length of $D$,
- $dl_{\text{avg}}$ is the average document length in the collection,
- $N$ is the total number of documents in the collection,
- $df_t$ is the document frequency of term $t$,
- $tf_t^Q$ is the term frequency of term $t$ in query $Q$,
- $s$ is a tuning parameter for document length normalization.

**Key Characteristics of Pivoted Normalization**

- It is a standard benchmark for information retrieval.
- Requires tuning to adjust for different collections (e.g., the parameter $s$).
- There are potential issues with normalization, as this approach can sometimes overcompensate for document length differences.

This weighting scheme is derived in a different manner compared to others, but it still fits into the framework of local term weighting, global term weighting, and query weighting.

# Axiomatic Approaches to Weighting Schemes

The idea behind **axiomatic approaches** is to develop a principled method for designing weighting schemes by defining a set of constraints (or axioms) that all good or correct weighting schemes should follow. If we are assigning weights to terms, the weighting schemes should adhere to these constraints to ensure consistency and correctness.

## Constraints of Axiomatic Approaches

- **Constraint 1: Adding a Query Term Increases the Score**
  Adding a query term to a document must always increase the score of the document. The intuition is that having an extra query term in the document increases the likelihood that the document is relevant to the query.
- **Constraint 2: Adding a Non-Query Term Decreases the Score**
  Adding a non-query term to a document must always decrease the score. The idea is that adding a term not present in the query increases the length of the document without making it more relevant to the query.
- **Constraint 3: Sub-Linear Term Frequency Growth**
  Adding successive occurrences of a term to a document must increase the score less with successive occurrences. This means that any term-frequency factor should be sub-linear. Repeated occurrences of a term indicate relevance, but the likelihood does not grow linearly with occurrences. For example, encountering a term 30 times does not increase relevance 30-fold.
- **Constraint 4: Document Length Normalization**
  Using the vector length should serve as a better normalization factor for retrieval. However, using the vector length alone may violate one of the existing constraints. To avoid this, the document length factor must be used in a sub-linear function to ensure that repeated appearances of non-query terms are weighted less.

New weighting schemes that adhere to these constraints have been shown to outperform best-known benchmarks.