TF-IDF

Term Frequency / Inverse Document Frequency

LING 575
HW1
Einar Horn, Lu Liu, Avijit Vajpayee, Daniel Campos, Peter Schoener

Method introduction

TF-IDF

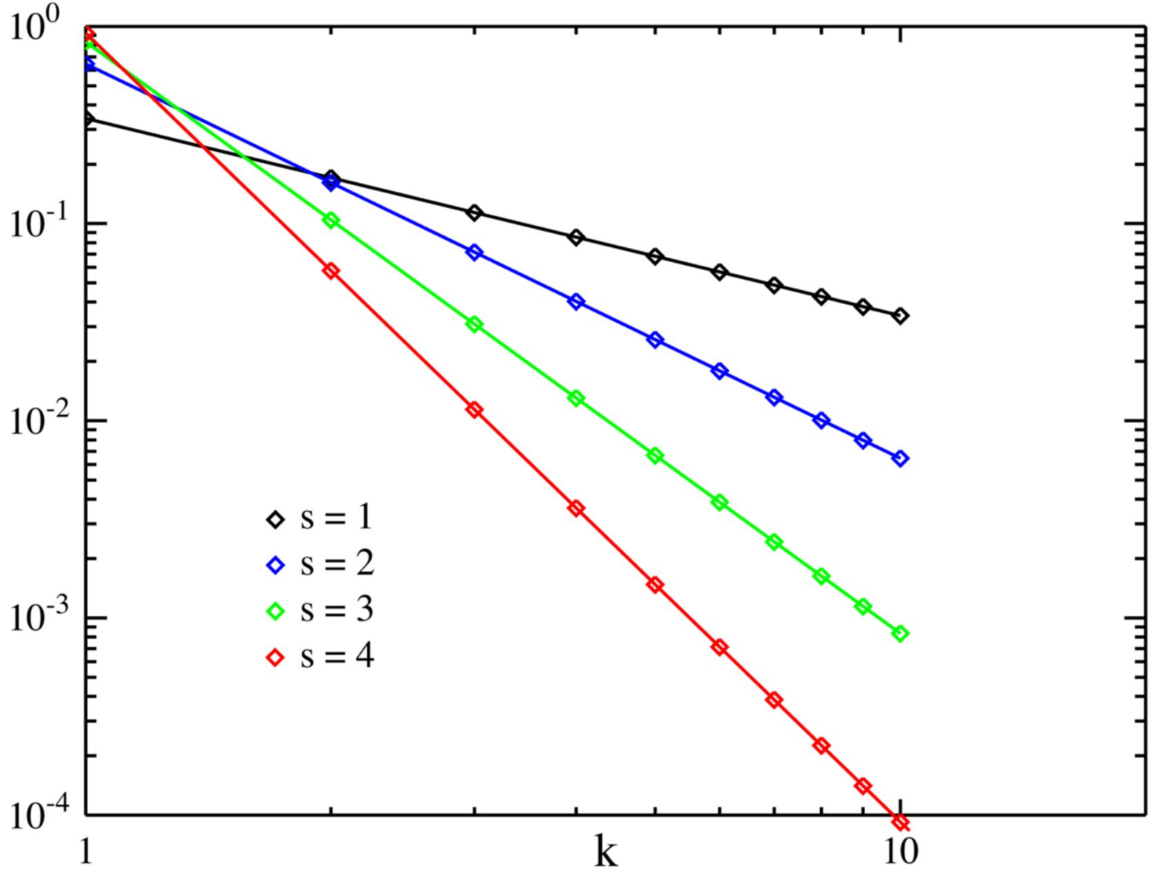
- A statistical method to represent how important an individual word is given a corpus of documents.
- The metric is a combination of components
    - **Term Frequency** : How common is the term in the given document
    - **Inverse Document Frequency** : How uncommon is the term across documents
- NOT a representation but a metric / numerical statistic
- Used as a weighting factor for representations of words in a corpus

Origin

- Inverse Document Frequency was first proposed by Karen Sparck Jones in 1972
  - First called term specificity
  - Proposed as a heuristic but proving theoretical foundations has been tricky
  - Loose Connection proposed to Zipf's law.
- Term Frequency was first proposed by Hans Peter Luhn in 1957

Zipf’s Law

- Zipf’s Law:
    - Many types of data studied in the physical and social sciences can be approximated with a Zipfian distribution, one of a family of related discrete power law probability distributions.
- Zipf’s Law for NLP:
    - The frequency of any word is inversely proportional to its rank in the frequency table.
    - The frequency of the n-th most frequent word is roughly proportional to 1/n
- Brown Corpus:
    - Thus "the" constitutes nearly 7%,"to" and "of" more than another 3% each; About half the total vocabulary of about 50,000 words are *hapax legomena*: words that occur only once in the corpus



10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
k
10
s = 1
s = 2
s = 3
s = 4

Zipf's Law Takeaways

- The most frequent words are not informative
    - *a* , *and* , *the, be, of, with* etc.
- Long tail of rare words that occur just once
- Informativeness : balance between two ends

Relevancy to 575

- TF-IDF is not a text representation in the modern sense but represents of a words salience for a given document in respect to a corpus.
- Frequency used as a weighting factor
- This representation allows for a scalable way of understanding relative relations of words in a corpus in a scalable way.
- TF-IDF can scale to virtually any size corpus with minimal memory requirements.

Algorithm details

Basic Equation

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

t = term
d = document
D = corpus

Term Frequency

A count of how many times a term t appears in document d with various normalization functions

Binary	0/1
Raw Count	$f_{t,d}$
Normalized	$\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$
Log Scaled	$log(1 + f_{t,d})$
Adjusted for Document Length	$0.5 + 0.5 \frac{f_{t,d}}{max_{t' \in d} f_{t',d}}$

Inverse Document Frequency

A measure of how much information the word provides(Is it common or rare across all documents).

n_t : # documents with term t

N : # documents

- IDF $\quad log\frac{N}{n_t}$
- Smooth $\quad log\frac{N}{1+n_t}$
- Max Normalized $\quad log\frac{max_{\{t' \in d\}} n_{t'}}{1+n_t}$
- Probabilistic $\quad log\frac{N-n_t}{1+n_t}$

TF-IDF Example

- "the"
  - tf("the", d_1) = 5/7
  - tf("the", d_2) = 6/9
  - idf("the", D) = log(2/2) = 0
  - tf-idf("the", d_1) = tf-idf("the", d_2) = 0
  - "The" is not an informative term
- "assignment"
  - tf("assignment", d_1) = 0/7
  - tf("assignment", d_2) = 2/9
  - idf("assignment", D) = log(2/1) = 1
  - tf-idf("assignment", d_1) = 0
  - tf-idf("assignment", d_2) = 2/9
  - "Assignment" is more informative in Document 2

Document 1

Word	Freq
the	5
homework	2
assignment	0

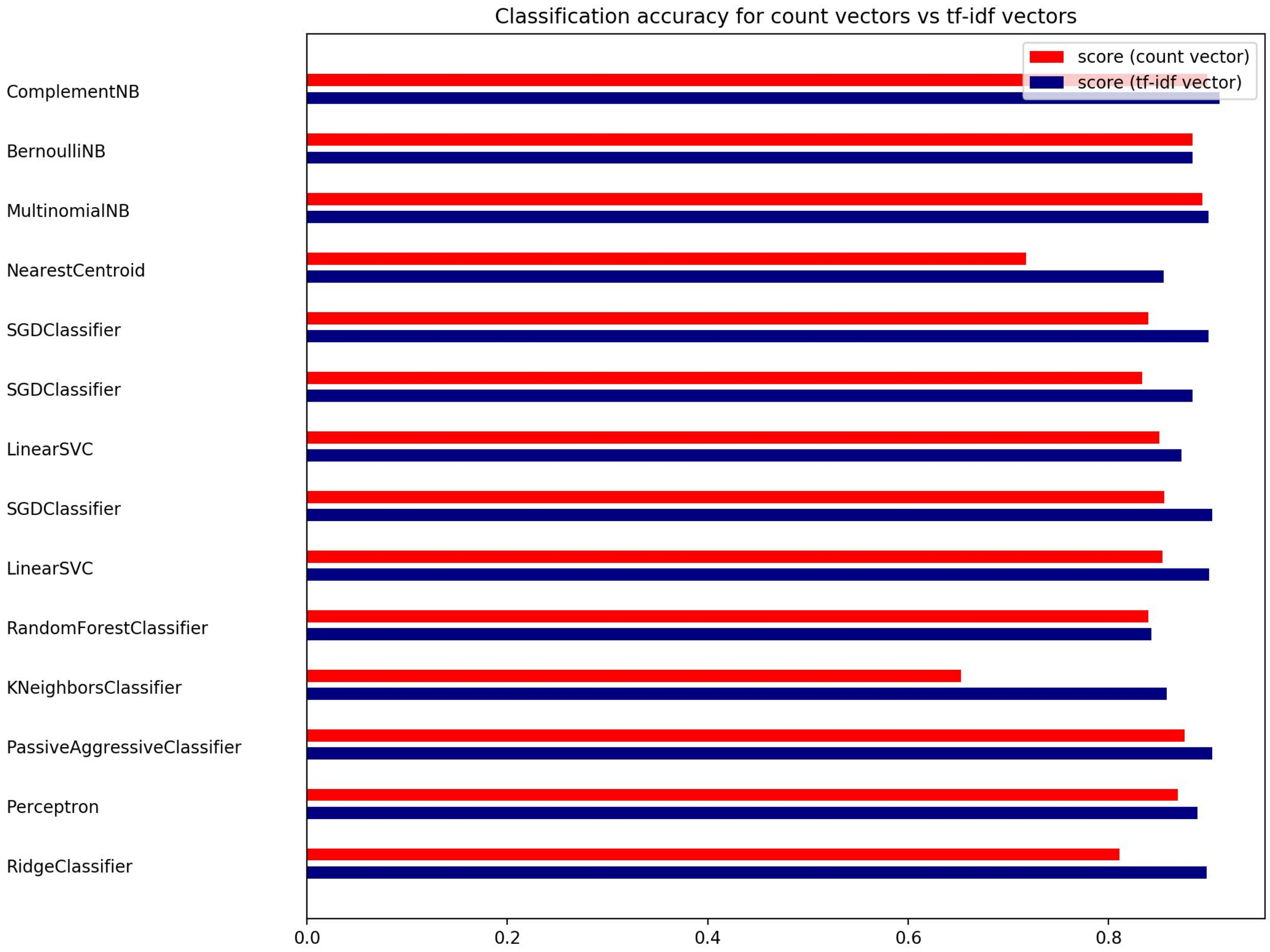
Document 2

Word	Freq
the	6
homework	1
assignment	2

Performance

Count vectors vs TF-IDF vectors

- Task: Classify text documents by topic using various classifiers
- Compare results of using count vectors and tf-idf vectors
- Corpus: 18000 newsgroups posts on 20 topics
    - Topics: baseball, computers, religion, etc



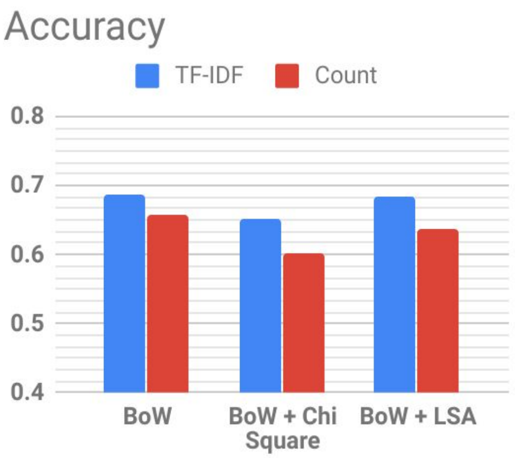
Takeaway: TF-IDF outperforms count vectors in every classifier

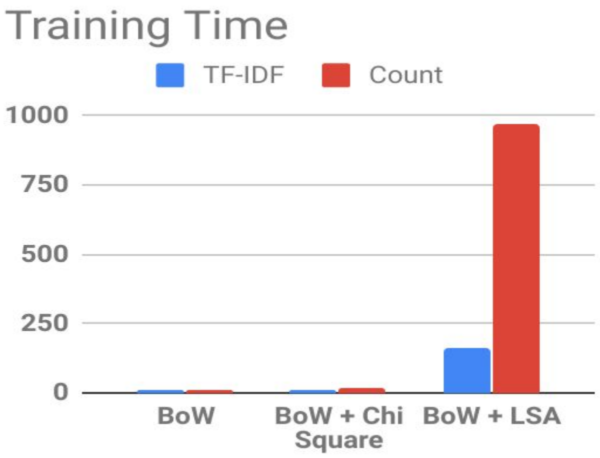
Code: https://github.com/einarhorn/tf-idf-classification

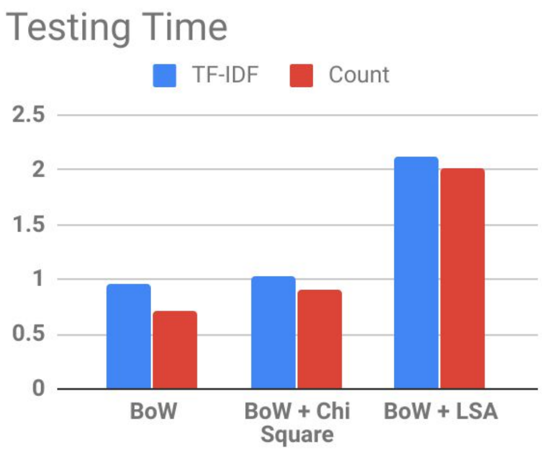
Space and Time Complexity

- **Count Vectors :** $O(V)$ time , $O(V^2)$ space
- **TF IDF Vectors :** $O(V^2)$ time , $O(V^2)$ space
- Collecting counts/collocation statistics : Fully parallelizable
- Space Complexity a bigger issue
  - Large sparse vectors not as ideal for "most" downstream ML estimators
  - Sparse vectors generally used with dimensionality reduction techniques
    - Top Terms selection through Chi-Square
    - Matrix Factorization (SVD / NMF)
    - TF-IDF weighted average of neural word embeddings

- TF-IDF vs. Count Sparse Vectors in 3 settings
- Linear SVM Classifier
- 20 News Groups Dataset







Code : https://github.com/einarhorn/tf-idf-classification/blob/master/lsa_weighting_comparison.py

Applications

Stopwords Filtering

- TF-IDF to filter stopwords
- Frequent words are weighted higher.
- If the frequent words are frequent in all documents, then their weight is lowered.
- Stopwords tend to have low IDF and TF-IDF.

Search Engines

- Ranking function, to rank matching documents according to their relevance to a given search query, simplest one is to sum the TF-IDF for each query term.
- The cores of ranking function - BM25 and TF-IDF.
- BM25 is the next generation of TF-IDF: it improves upon TF-IDF.

- $f(q_i, D)$ is q_i's term frequency in document D, IDF(q_i) is the IDF weight of the query term q_i.
- |D| is the length of the document D in words.
- Avgdl is the average document length in the text collection from which documents are drawn.
- k_1 and b are free parameters, usually k_1 is between [1.2, 2.0] and b = 0.75.

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

Recommendation Engines

- Content based recommendation system
- System recommends similar items according to keywords or properties of items with a distance measure.
  - Convert texts or words into vectors
  - TF-IDF to weight and construct a TF-IDF matrix.
  - Cosine similarity

Text Summarization

- Assign scores based on TF-IDF to sentences, and take the top scoring *n* sentences as a summary
- How to assign score to sentence?
  - Sum of TF-IDF values
  - Paper by Seki suggests the following:
    - sum noun terms
    - Biase the weights of words in sentence that are in title of the document
    - multiply this sum by the position value of the sentence in the document

Seki, Yohei. "Sentence Extraction by tf/idf and position weighting from Newspaper Articles." (2002).

Document Classification and Clustering

- Classify or cluster the documents - numerical representation of the sentences or documents.
- Represent the documents into mutually comparable vectors.
- The documents can be represented using the TF-IDF scores.
- Then we can represent each document as a vector of terms using a global ordering of each unique term found throughout all the documents.

Variants

Proportional Document Frequency (PDF)

- Measures the difference of how often a term occurs in different domains.
- Proportional Document Frequency is computed as $exp(n_{jc}/N_c)$

$$W_j = \sum_{c=1}^{c=D} |F_{jc}| \exp(\frac{n_{jc}}{N_c}),$$

$$|F_{jc}| = \frac{F_{jc}}{\sqrt{\sum_{k=1}^{k=K} {F_{kc}}^2}},$$

IDuF

- The TF component in TF-IDuF is the same with TF-IDF.
- Classic IDF is calculated using the document frequencies in the recommendation corpus.
- IDuF is calculated using the document frequencies in a user’s personal document collection c_u.

Sparse Word Vector Representations

Information Theoretic Background

Information Content / Self Information : (for an event) amount of information gained when sampled

$$I(E) = -log(Pr(E))$$

Entropy (H) : Expected value of information content for a random variable

Mutual Information : amount of information about one R.V. through observing another R.V.

$$MI(X,Y) = \sum p(x,y) log \frac{p(x,y)}{p(x)p(y)}$$

Pointwise Mutual Information

- *Strength of association* : word to word ([Church and Hanks 1990])
- Distributional Hypothesis of Semantics
- Define a window of context length C
- Each word represented as a sparse vector of length |V| (size of vocab)
    - How many times each word in vocab appears within context of given word
    - Mutual Information between 2 R.V. is the expected value of PMI

$$PMI = log_2 \frac{P(word_1, word_2)}{P(word_1) * P(word_2)}$$

$$PPMI = max(PMI, 0)$$

Extensions to PMI

- **Normalized PMI (NPMI) :**
  - Bound between [-1, +1]
- **Context Smoothing (Vector Semantics) :**
  - PMI biased towards infrequent events
  - Slightly shifts probability to rare context words
- **Shifted PPMI**
  - Related to SkipGram model of Word2Vec trained with Negative Sampling

$$\frac{PMI(w,c)}{I(w,c)} = \frac{PMI(w,c)}{-log(p(w,c))}$$

$$PPMI_\alpha(w,c) = max(log\frac{P(w,c)}{P(w)P_\alpha(c)},0)$$

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha} \qquad \alpha = \frac{3}{4}$$

$$SPPMI(w,c) = max(PMI(w,c) - logk, 0)$$

Connection to Dense Neural Embeddings

- GloVe
  - Objective directly minimizes the difference between dot product of word and context vectors and log of co-occurrences.
- word2Vec : Neural Word Embeddings as Implicit Matrix Factorization (Levy et. al. 2014)
  - SkipGram with Negative Sampling (SGNS) implicitly factorizes word context matrix
  - SGNS superior for word analogy
  - SVD on PMI matrix at least as good for word similarity
  - Excellent Blog - Sebastian Ruder

Appendix

Relevant Papers

- Understanding inverse document frequency: on theoretical arguments for IDF
- A Statistical Approach to Mechanized Encoding and Searching of Literary Information*
- Sentence Extraction by tf/idf and Position Weighting from Newspaper Articles
- From RankNet to LambdaRank to LambdaMART: An Overview
- Relevance weighting of search terms
- A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL

Contributions

- Daniel Campos : Introduction to TF-IDF, Zipf's Law, Variants
- Einar Aleksander Horn : TF-IDF Calculations, Performance Benchmarking, Text Summarization
- Lu Liu: Applications, Variants
- Avijit Vajpayee : TF-IDF Formulations, Performance Benchmark, PPMI
- Peter Schoener : Applications, PPMI