Duplicates methodology

19 de octubre de 2023

1. TF-IDF algorithm

TF-IDF, which stands for Term Frequency-Inverse Document Frequency, is a numerical statistic used in information retrieval and natural language processing to evaluate the importance of a word within a document relative to a collection of documents (corpus). It's a way to represent text data in a way that takes into account both the local and global significance of words.

Term Frequency (TF): Term Frequency measures how frequently a term (word) appears in a document. It's calculated as the number of times a term appears in a document divided by the total number of terms in that document.

$$tf(t,d) = \frac{f(t,d)}{\max f(t,d) : t \in d} \tag{1}$$

Where $tf(t,d)$ is a variable that gives 1 if the term "t appears in document d and 0 if not.

Inverse Document Frequency (IDF): Inverse Document Frequency measures the importance of a term across a collection of documents in our case across all the 'avisocuerpo' ads. It's calculated as the logarithm of the total number of documents divided by the number of documents containing the term.

$$idf(t,D) = \log \frac{|D|}{|d \in D : t \in d|} \tag{2}$$

Where D standas for the collection of documents, in our case a matrix containing each 'avisocuerpo' string in each year and $|d \in D : t \in d|$ the number of documents 'avisocuerpo' ads where the term t appears, if the term does not appear, the algorithm ads 1 in order to avoid division by zero $1 + |d \in D : t \in d|$

Finally, the TF-IDF score of a term in a document combines both the term's local importance (TF) and its global importance (IDF).

It's calculated as the product of TF and IDF for a given term in a document.

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \tag{3}$$

The key idea behind TF-IDF is that it assigns a higher score to terms that are important within a specific document (high TF) but relatively rare across the entire corpus (high IDF). Terms that are common across all documents receive lower scores. This is what we use to capture similarities accross ads.

Applications of TF-IDF:

Information Retrieval: TF-IDF is used in search engines to rank documents based on their relevance to a search query. Text Classification: It's used to classify documents into categories based on the importance of terms. Document Summarization: TF-IDF can help identify important sentences or passages in a document. Content-Based Recommendation: It's used in recommendation systems to find documents similar to the ones a user has shown interest in. In our case to identify similarities across documents (ads).

2. TF-IDF code

We're using The TfidfVectorizer() function from scikit-learn library, it is a tool for converting a collection of raw documents (in this case, 'avisocuerpo ads) into a matrix of TF-IDF features. that reflects the importance of a word or phrase within a document relative to a collection of documents.

Tokenization: The first step is to tokenize the text, which means breaking it down into individual words or terms. For example, the sentence "This is a job description"would be tokenized into the list of words ["This", ïs", .ª", "job", "description"].

Term Frequency - TF: For each document ('avisocuerpo' ad), it counts the number of times each term (word) appears in that document. This creates a matrix where each row represents a document, and each column represents a unique term from all the documents.

Inverse Document Frequency - IDF: IDF measures the importance of a term in the entire collection of documents. It's calculated as the logarithm of the total number of documents divided by the number of documents containing the term. Terms that appear in many documents have a lower IDF, while terms that appear in few documents have a higher IDF.

Term Frequency-Inverse Document Frequency - TF-IDF: This is the product of TF and IDF. It quantifies the importance of a term within a document relative to its importance in the entire corpus. TF-IDF is higher when the term occurs frequently in the document but is rare in other documents.

Normalization: By default, TfidfVectorizer normalizes the TF-IDF values for each document. This normalization ensures that the length of the document doesn't affect the TF-IDF values.

Vectorization: The result is a matrix where each row corresponds to a document, and each column corresponds to a unique term. Each entry in the matrix represents the TF-IDF score of a term in a document.

After this process we get a very large matrix that can be compared to itself with a cosine similarity algorithm that gives us a coefficient for each row that in this case represents each 'avisocuerpo' ad.

Now is neccesary to define a kind of similarity threshold for the cosine similarity coefficient above wich we can say that the string is "similar"to others, after some experiments, we define 0.89 as the similarity threshold.

3. References

- scikit-learn TfidfVectorizer Documentation
- scikit-learn Text Feature Extraction Documentation