TF-IDF

Imagine you're running a popular blog that features a wide variety of recipes.

Your blog contains 1000s of recipes… from simple breakfasts to gourmet dishes

You have a search feature. But it's not good – people complain that when they search for "low carb breakfast", they get shown generic breakfast recipes.

Why? What's the problem that's happening here?

TF-IDF

Problem: The search is using basic string/keyword matching. It treats all words as equally important.

- Less relevant results are ranked as highly as more relevant ones

TF-IDF

Solution: Rank more important words as… more important

How?

TF-IDF

TF-IDF!

Consists of two terms: TF, and IDF

Term Frequency (TF): Measures how often a term occurs in a document.

- The more often a term appears in a document, the more important it is for that document.

Inverse Document Frequency (IDF): Measures how common or rare a term is across all documents in the corpus.

- Terms that appear in many different documents are less significant than those that appear in a smaller number of documents

TF-IDF

This means that a search for "low-carb breakfast" will prioritize recipes where "low-carb" is a significant term, rather than returning recipes with the more common "breakfast" term.

TF-IDF

This means that a search for "low-carb breakfast" will prioritize recipes where "low-carb" is a significant term, rather than returning recipes with the more common "breakfast" term.

TF-IDF

Term Frequency (TF):
Upweights words w that are more Important to d

Frequency of w occurring together with d

$$\mathrm{TF}_{w,d} = \frac{\mathrm{count}(w,d)}{\sum_{w'} \mathrm{count}(w',d)}$$

Inverse Document Frequency (IDF):
Downweights words that appear everywhere

Size of document collection

$$\mathrm{IDF}_{w,D} = log \frac{|D|}{|\{d \in D : w \in d\}|}$$

Number of documents in D that contain w

$$\text{TF-IDF}_{w,d,D} = \mathrm{TF}_{w,d}\mathrm{IDF}_{w,D}$$