Text Technologies

Vector Space Model of IR
Victor Lavrenko

Basic VS: dimensions = words
- Separate dimension for each distinct word
  - words ≠ coordinate vectors
- Value along dimension "cat" = number of times "cat" occurs

Comparing documents to queries
- Distance between points in space
  - Euclidean, or angle between vectors
    - Usually expressed as similarity
  - Distance: D(A, B) = √(Σ (aᵢ - bᵢ)^2)

Inverse Document Frequency
- Observation 4: Rare words carry more meaning
  - Overt, covert, adverbial, . . . , topical content
  - Said, went, of the big, etc. . . , linguistic glue
- Give more weight to rare words: Inverse Document Frequency
  - df(A) = number of documents in collection
  - df(A) = number of documents containing
  - New similarity formula:

Dot product
- Similarity of document vectors a and b:
  - a × b = [a₁b₁, a₂b₂, . . . , aₙbₙ]
  - a × b = ∑aᵢbᵢ
- Geometrically:
  - Length of projection of b onto a
  - 0 if a and b are orthogonal (no words in common)
  - Cosine of the angle between a and b
  - Relation to Euclidean distance between a and b:
  - d(a, b) = √(Σ (aᵢ - bᵢ)^2)
  - Equivalent if a and b are unit-length:
    - 1 - a × b = ∑[aᵢbᵢ]
  - Relation to intersection of sets a, b:
    - |a ∩ b| = ∑aᵢbᵢ
  - If word "cat" belongs to a, otherwise a
  - a × b = a₁b₁ + a₂b₂
  - Words that belong to b and a:

Inverse Document Frequency (idf)
- Very effective heuristic for picking out important words
- Sometimes idf used on the query weights qᵢ
- Logarithm to put idf on the same scale as the tf component

Bag-of-words matching
- Only in story 1
  - about
  - computer
  - only
  - each
  - space
  - our
- In both stories
  - binary
  - explaining
  - home
  - love
  - scale
  - through
  - are
  - in
- Only in story 2
  - astronomy
  - been
  - been
  - born
  - could
  - could
  - computer
  - either
  - mass
  - planar
  - others
  - prefer
  - radar
  - sophisticated
  - beam

How to combine all this into a similarity measure?

Vector Space Model
- Everything is a vector in some high-dimensional space
  - Words, documents, queries, user preferences
  - Issues to consider
    - What are the dimensions of that space (basis vectors)?
    - How to project words/documents/queries to that space?
    - How to compare documents and queries?
- Dimensions
  - Basis: set of line-independent (orthogonal) basis vectors
  - “Core” semantic concepts: works on toy datasets
  - Words in the corpus: one dimension per distinct word
    - Not orthogonal, huge dimensionality, constantly-growing

Term (word) Weighting
- Term weight: relative importance of word in a doc
  - D(A) coordinate of D along dimension w
  - Observation 1: presence / absence most important
    - Weight = 1 if word present, 0 otherwise
    - Document = binary vector = set
  - Observation 2: key words tend to be repeated in a doc
    - tf(w, D) = number of times w occurs in D
  - Observation 3: biased towards long documents
    - Long docs -> higher tf, spurious word occurrences
    - Normalize by document length |D|:

Frequency Normalization
- Observation 5:
  - Q = "angry aardvark"
  - D₁ = "... angry . . . aardvark . . ."
    - D₂ = "... aardvark . . . aardvark . . ."
    - Is D₁ or D₂ more relevant? Which will be ranked higher?
- Correction:
  - First occurrence more important than a repeat (why?)
    - Squash the growth of term frequency
      - Larger K = less (no squash)
      - Small K = step function
  - Observation 6:
    - Repetitions important in long docs
      - Make it reflect document length

tf-idf selects informative terms

DC-9 WITH 55 ABOARD CRASHES; AT LEAST 16 DEAD
CHARLOTTE, N.C. (AP) — A DC-9 with 55 people on board crashed and burned into trees during a thunderstorm after making an approach to Charlotte Douglas International Airport Saturday, killing at least 16 people. The flight, which originated in Columbia, S.C., and was on final approach, hit a house near the airport as it attempted to land, said...
tf.idf weighted sum

\[ s(Q, D) = \sum_{w \in D} \left( \frac{f_w}{|D|} \log \frac{1}{df_w} \right) \]

The more query words we match, the better. Over the vocabulary.
- rank documents in order of decreasing \( s(Q, D) \)
- state-of-the-art ranking formula for short queries
- variations actively used by many search engines

Vector similarity measures

\[ s(Q, D) = \sum_{w \in D} \left( \frac{f_w}{|D|} \log \frac{1}{df_w} \right) \]

- inner product
- Jaccard coefficient
- Cosine coefficient
- differences minor compared to how you set \( Q_w, D_w \)

Normalized to unit length \( \rightarrow \) all rank-equivalent to dot product

Example: weighted cosine

\[ D = 0.5 \ast \text{cat} + 0.8 \ast \text{dog} + 0.3 \ast \text{lion} \]
\[ Q = 0 \ast \text{lion} + 1.5 \ast \text{cat} + 0.1 \ast \text{dog} \]

Cosine coefficient:

\[ \cos(\theta) = \frac{Q \cdot D}{\|Q\| \cdot \|D\|} = \frac{0.5 \cdot 0.5 + 0.8 \cdot 0.8 + 0.3 \cdot 0.3}{\sqrt{0.5^2 + 0.8^2 + 0.3^2} \sqrt{0.5^2 + 0.5^2 + 0.3^2}} = 0.888 \]

More distance / similarity measures

\[ \rho \cdot \text{norm distance} = \rho \cdot \sqrt{\sum_{w \in D} (Q_w - D_w)^2} \]

p=2, Euclidean
p=1, Manhattan
p=\infty, max |Q_w-D_w| = logical OR
p=0, min |Q_w-D_w| = logical AND

- Treat documents / queries as word histograms
- \( Q_w, D_w \) non-negative, add up to 1
- \( k_w = \sum_{w \in D} \log Q_w \)

- Remember to convert distance to similarity (e.g. negate)
- Still need to set weights \( Q_w, D_w \)

Beyond bag-of-words

VSM: dimensions = words

- Can try phrases:
  - term: pair of adjacent words
  - or words in a k-word window
- or use NLPI:
  - noun phrases / verb phrases
  - head-modifier pairs
  - dependency chains
- Intuitively appealing, but:
  - very hard to beat words
  - weighting very important

Uses of VSM

- Not just ranking documents in response to query
- Any time you want to know if text A is similar to text B:
  - does this essay look like the writing of author B?
  - does patent A infringe on any part of patent portfolio B?
  - does email A look more like spam emails?
  - is piece of code similar to any part of system B?
- Determine if word/phrase A is similar to word/phrase B:
  - inverted list for a word is a vector over documents
  - similarity(\text{cat, lion}) = cosine of their inverted lists
  - need to customize weighting (no idf, length\(^2\), etc.)

Summary

- Everything is a vector, one dimension per word
- Rank by similarity of document vectors to query
  - dot product or cosine of the angle between vectors
- Term weighting: very important, \( tf.idf \) is universal

Heuristic in nature:
- easy to assimilate good ideas from other retrieval models
- components not interpretable \( \rightarrow \) no guide what to try next
- encourages ad-hoc engineering: tweak, test out, tweak...
- no notion of relevance (\( = \) similarity?)
- Very popular, hard to beat