IN2110: Språkteknologiske metoder

Vektorrom for IR

Erik Velldal

Språkteknologigruppen (LTG)

22. Januar, 2019

Agenda

Next weeks

- How to represent our data in a mathematical model.
- Vector space models
- Examples: representing documents, words and meaning.
- Vector space classification methods.

Today

- Vector space models for Information Retrieval (IR).
- Modeling the similarity of documents, queries, and topics.

- In a very high-level terms, classification and all machine learning models can be described as a learned mapping from inputs to outputs.

- The first step is to represent our data in a way that the model can take as input.
- Very often done by describing the data by a set of features.

Feature engineering

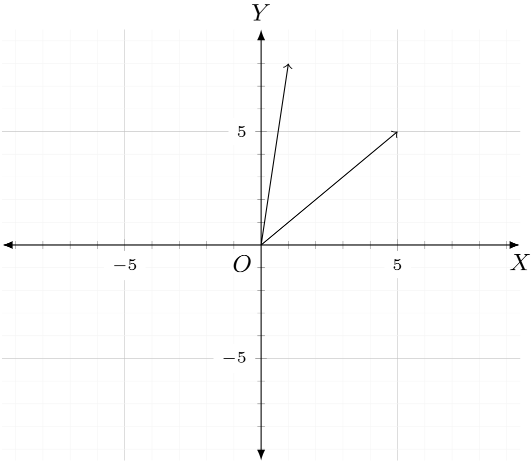
- Features record observable and relevant properties of the data.
- Every feature has a numerical value.
- Each object x to be modeled described as a tuple of d feature values:
- a feature vector: $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_d \rangle$
- Features are also numerically indexed.

Vector space models

- Once our data is represented as feature vectors, we often adopt a vector space model.
- Based on a spatial metaphor.
- Feature correspond to dimensions or coordinate axes in the space.
- Object correspond to points in this feature space.
- Each example x is a point or vector $\boldsymbol{x}$ in a space of d dimensions:
- $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_d \rangle$
- To measure the similarity of two objects, we can measure their geometrical distance / closeness in the model.

Vectors and vector spaces

- Simple example of a 2-dimensional space, $\mathbb{R}^2$.
- Two vectors: $v1 = [1, 8]$, $v2 = [5, 5]$
- In practice we only use the first (positive) quadrant.



Dimensionality

- For $d = 1$, examples are just points on a line.
- For $d = 2$, examples points in a plane.
- For $d = 3$, we have a three-dimensional space.
- For $d > 3$, it becomes difficult to visualize.
- High-dimensional spaces where d is the thousands or even millions not uncommon in ML/NLP.

Vector space models for Information Retrieval

- We want to be able to quantify how similar different documents are.
- Or how relevant documents are to a given query.
- A simple and wide-spread approach:
- The features representing a document =
- frequency counts of all the words that occur in the text.
- So-called bag-of-words (BoW) features.
- Each word type corresponds to a dimension.

Token vs. type

"Rose is a rose is a rose is a rose." *Gertrude Stein*

- How many words? (Assuming we ignore case and punctuation.)

Token vs. type

"Rose is a rose is a rose is a rose." *Gertrude Stein*

- How many words? (Assuming we ignore case and punctuation.)
- Three types and ten tokens.

The matrix view

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Conceptually, a vector space is often thought of as a matrix, often called co-occurrence matrix.
- For m documents and a vocabulary of n words, a BoW document representations would be called a $n \times m$ term–document matrix.
- Rows represent words (features) in the vocabulary, and columns represent the feature vectors of documents.

The matrix view

- We assume that documents that share many of the same words are semantically similar in terms of their content.
- Note that we can also view the rows as vectors representing words.
- Words that tend to co-occur in the same documents will tend to be semantically related.
- This is called the distributional hypothesis, and we will return to this later in the course!

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Can compute similarity (of either words or documents) based on distance in the space.
- Several ways this can be done.

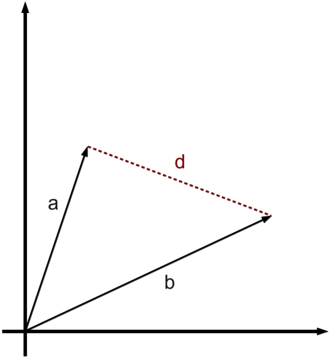
- We can now compute *document similarity* in terms of *spatial distance*.
- One standard metric for this is the *Euclidean distance*:

$$d(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\sum_{i=1}^{n} (\boldsymbol{a}_i - \boldsymbol{b}_i)^2}$$

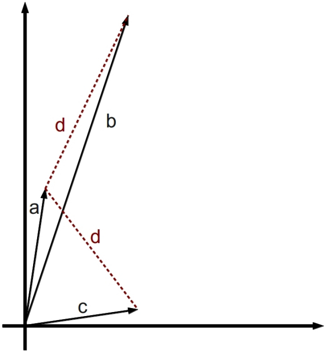
- Computes the norm (or *length*) of the *difference* of the vectors.
- The norm of a vector is:

$$\|\boldsymbol{x}\| = \sqrt{\sum_{i=1}^{n} \boldsymbol{x}_i^2} = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}}$$

- Intuitive interpretation: The distance between two points corresponds to the length of the straight line connecting them.



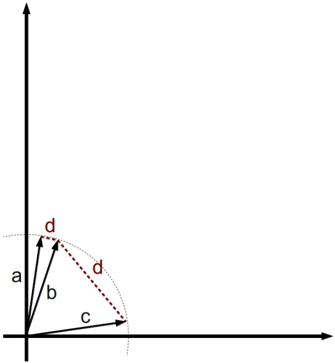
Euclidean distance and length bias



- Three document vectors, $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$
- $d(\boldsymbol{a}, \boldsymbol{b}) = 10$
- $d(\boldsymbol{a}, \boldsymbol{c}) = 7$

- However, a potential problem with Euclidean distance is that it is very sensitive to extreme values and the length of the vectors.
- As vectors of words with different *frequencies* will tend to have different length, the frequency will also affect the similarity judgment.

Overcoming length bias by normalization



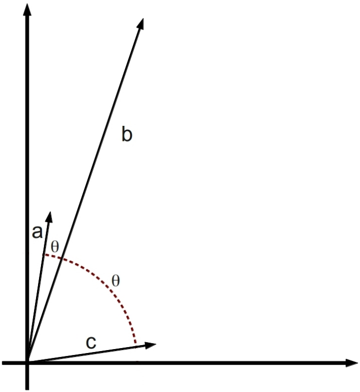
- One way to reduce frequency effects is to first normalize all our vectors to have unit length, i.e. $\|\boldsymbol{x}\| = 1$
- Can be achieved by simply dividing each element by the length: $\boldsymbol{x}\frac{1}{\|\boldsymbol{x}\|}$
- Amounts to all vectors pointing to the surface of a unit sphere.

Cosine similarity

- We can measure (cosine) *proximity* rather than (Euclidean) *distance*.
- Computes similarity as a function of tl

$$\cos(\boldsymbol{a}, \boldsymbol{b}) = \frac{\sum_i \boldsymbol{a}_i \boldsymbol{b}_i}{\sqrt{\sum_i \boldsymbol{a}_i^2}\sqrt{\sum_i \boldsymbol{b}_i^2}} = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{a}\|\|\boldsymbol{b}\|}$$

- Avoids the arbitrary scaling caused by dimensionality, frequency, etc.
- Constant range between 0 (for orthogonal vectors) and 1 (for vectors that point in the same direction).



Cosine similarity (cont'd)

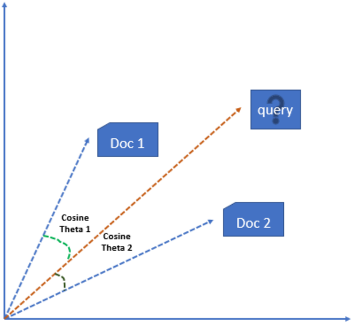
- For *normalized* (unit) vectors, the cosine is simply the *dot product*:

  $\cos(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{b}_i$

- Can be computed very efficiently.
- The *same relative rank order* as the Euclidean distance for unit vectors!

Queries and relevance ranking

- Central task in information retrieval:
- Identifying and ranking relevant documents for a given query (i.e. search terms).
- Treat the query as a short document:
- Represent it as a vector and find its nearest neighbors.
- I.e. rank the documents based on the distance between the document vectors and the query vector.



Frequencies

- Problem: Raw frequency counts not always good indicators of relevance.
- The most frequent words will typically not be very discriminative.
- A weighting function is therefore usually applied to the raw counts.

TF-IDF

- The most commonly used weighting function is tf-idf:

TF-IDF

- The most commonly used weighting function is tf-idf:
  - The term frequency $\mathrm{tf}(t_i, d_j)$ denotes the number of times the term t_i occurs in document d_j.

TF-IDF

- The most commonly used weighting function is tf-idf:
  - The term frequency $\mathrm{tf}(t_i, d_j)$ denotes the number of times the term t_i occurs in document d_j.
  - The document frequency $\mathrm{df}(t_i)$ denotes the total number of documents in the collection that the term occurs in.

TF-IDF

- The most commonly used weighting function is tf-idf:
  - The term frequency $\mathrm{tf}(t_i, d_j)$ denotes the number of times the term t_i occurs in document d_j.
  - The document frequency $\mathrm{df}(t_i)$ denotes the total number of documents in the collection that the term occurs in.
  - The inverse document frequency is defined as $\mathrm{idf(t_i)} = \log\left(\frac{N}{df(t_i)}\right)$, where N is the total number of documents in the collection.

TF-IDF

- The most commonly used weighting function is tf-idf:
  - The term frequency $\mathrm{tf}(t_i, d_j)$ denotes the number of times the term t_i occurs in document d_j.
  - The document frequency $\mathrm{df}(t_i)$ denotes the total number of documents in the collection that the term occurs in.
  - The inverse document frequency is defined as $\mathrm{idf(t_i)} = \log\left(\frac{N}{df(t_i)}\right)$, where N is the total number of documents in the collection.
  - The weight given to term t_i in document d_j is then computed as

$$\text{tf-idf}(t_i, d_j) = \mathrm{tf}(t_i, d_j) \times \mathrm{idf}(t_i)$$

TF-IDF

- The most commonly used weighting function is tf-idf:
  - The term frequency $\mathrm{tf}(t_i, d_j)$ denotes the number of times the term t_i occurs in document d_j.
  - The document frequency $\mathrm{df}(t_i)$ denotes the total number of documents in the collection that the term occurs in.
  - The inverse document frequency is defined as $\mathrm{idf(t_i)} = \log\left(\frac{N}{df(t_i)}\right)$, where N is the total number of documents in the collection.
  - The weight given to term t_i in document d_j is then computed as

$$\text{tf-idf}(t_i, d_j) = \mathrm{tf}(t_i, d_j) \times \mathrm{idf}(t_i)$$

  - A high tf-idf is obtained if a term has a *high* frequency in the given *document* and a *low* frequency in the document *collection* as whole.
  - The weights hence tend to filter out common terms.

Text pre-processing. Or, what is a word?

Raw: “The programmer’s programs had been programmed.”

- Tokenization: Splitting a text into sentences and words or other units.
- Different levels of abstraction and morphological normalization:
  - What to do with case, numbers, punctuation, compounds, . . . ?
  - Full-form words vs. lemmas vs. stems . . .
- Stop-list: filter out closed-class words or function words.
  - The idea is that only *content words* provide relevant context.

Text pre-processing. Or, what is a word?

Raw:	“The programmer’s programs had been programmed.”
Tokenized:	the programmer ’s programs had been programmed .

- Tokenization: Splitting a text into sentences and words or other units.
- Different levels of abstraction and morphological normalization:
  - What to do with case, numbers, punctuation, compounds, . . . ?
  - Full-form words vs. lemmas vs. stems . . .
- Stop-list: filter out closed-class words or function words.
  - The idea is that only *content words* provide relevant context.

Text pre-processing. Or, what is a word?

Raw:	“The programmer’s programs had been programmed.”
Tokenized:	the programmer ’s programs had been programmed .
Lemmatized:	the programmer ’s program have be program .

- Tokenization: Splitting a text into sentences and words or other units.
- Different levels of abstraction and morphological normalization:
  - What to do with case, numbers, punctuation, compounds, . . . ?
  - Full-form words vs. lemmas vs. stems . . .
- Stop-list: filter out closed-class words or function words.
  - The idea is that only *content words* provide relevant context.

Text pre-processing. Or, what is a word?

Raw:	"The programmer's programs had been programmed."
Tokenized:	the programmer 's programs had been programmed .
Lemmatized:	the programmer 's program have be program .
W/ stop-list:	programmer program program

- Tokenization: Splitting a text into sentences and words or other units.
- Different levels of abstraction and morphological normalization:
  - What to do with case, numbers, punctuation, compounds, ...?
  - Full-form words vs. lemmas vs. stems ...
- Stop-list: filter out closed-class words or function words.
  - The idea is that only *content words* provide relevant context.

Text pre-processing. Or, what is a word?

Raw:	"The programmer's programs had been programmed."
Tokenized:	the programmer 's programs had been programmed .
Lemmatized:	the programmer 's program have be program .
W/ stop-list:	programmer program program
Stemmed:	program program program

- Tokenization: Splitting a text into sentences and words or other units.
- Different levels of abstraction and morphological normalization:
  - What to do with case, numbers, punctuation, compounds, . . . ?
  - Full-form words vs. lemmas vs. stems . . .
- Stop-list: filter out closed-class words or function words.
  - The idea is that only *content words* provide relevant context.

Practical comments: Sparsity

- BoW feature vectors will be extremely high-dimensional.
- The number of *non-zero* elements will be very low.
- Few active features per word.
- We say that the vectors are sparse.
- This has implications for how to implement our data structures and vector operations:
- Don't want to waste space representing and iterating over zero-valued features.

Next: Two categorization tasks in machine learning

Classification

- Supervised learning, requiring labeled training data.
- Train a classifier to automatically assign *new* instances to *predefined* classes, given some set of training examples.
- (Topic for next week.)

Next: Two categorization tasks in machine learning

Classification

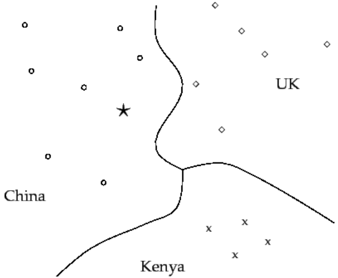
- Supervised learning, requiring labeled training data.
- Train a classifier to automatically assign *new* instances to *predefined* classes, given some set of training examples.
- (Topic for next week.)

Clustering

- Unsupervised learning from unlabeled data.
- Automatically group similar objects together.
- No predefined classes or structure, we only specify the similarity measure.
- (The topic for the week after.)

Classes and classification

- In our vector space model, objects are represented as points, so classes will correspond to collections of points; regions.
- Vector space classification is based on the contiguity hypothesis:



- Objects in the same class form a contiguous region, and regions of different classes do not overlap.
- Classification amounts to computing the boundaries in the space that separate the classes; *the decision boundaries*.

Vector space classification

- Classifiers based on vector space representations are well-suited for introducing the notion of classification:
- Little math required, easy to understand on the basis of geometrical intuitions.
- We will consider two very simple but powerful methods:
- K-Nearest Neighbor (KNN) classification
- Rocchio classification (a.k.a. Nearest centroid)
- Example task: text classification

Oblig 1a

- Theme for the 1st obligatory assignment:
- Topic classification of news articles (reviews in NoReC)
- using KNN
- and with BoW feature vectors using TF-IDF weighting.
- Deadline: 15/2
- Group work encouraged!

`https://www.uio.no/studier/emner/matnat/ifi/IN2110/v19/innleveringer.html`

Example applications of text classification

- Topic classification of news articles
- Authorship attribution
- Spam detection
- Polarity classification (sentiment analysis)
- Language identification
- Hate-speech / abusive language detection / threat detection
- Question type classification
- Content recommendation
- Political affiliation
- ...

Next week

- More on vector space models
- Classification algorithms: KNN-classification and c-means
- Reading: The chapter *Vector Space Classification* (sections 14-14.4) in Manning, Raghavan & Schütze (2008); `https://nlp.stanford.edu/IR-book/`.
- Want to learn more about IR? Take IN3120 (INF3800) – Search Technology.