
Lecture - 10

# Content-based Methods

Arpit Rana

25th Feb 2022

# Definition

**Content-based methods** try to predict the ***utility*** of items for an ***active user*** based on ***item descriptions*** and her ***past preferences***.

In content-based systems, there are choices on the following

- **Item representation**: how items are represented,
- **User profile**: how user preferences are modeled, and
- **Filtering technique**: how items are matched with the user preferences.

# Filtering Technique

A ***filtering technique*** suggests relevant items from a set of candidate items.

These techniques are also split into the following categories -

- ***Memory-based techniques***: employ similarity measures to match the representations of candidate items against the profile
- ***Model-based techniques***: learn from the profile a model that can predict item relevance

# Keyword-based Vector Space Model

VSM is a spatial representation of text documents wherein -

- each document is represented by a vector in a *n*-dimensional space
- each dimension corresponds to a term from the overall vocabulary of a given document collection

# Keyword-based Vector Space Model

- Imagine each item (e.g. movie) is represented by a binary-valued (column) vector of dimension ***d***, e.g. ***d = 3***, where each element of the vector corresponds to a feature (e.g. movie genre).
- We can gather these vectors into a matrix, which we will refer to as ***Q***
  - So, ***Q*** is a ***d x |I|*** matrix.
  - If we want to refer to the column in ***Q*** that corresponds to item ***i***, we will write $Q_i$

| | $i_1$ | $i_2$ | $i_3$ | $i_4$ | $i_5$ | $i_6$ |
|---|---|---|---|---|---|---|
| **comedy** | 1 | 0 | 0 | 1 | 1 | 0 |
| **thriller** | 0 | 0 | 0 | 0 | 1 | 1 |
| **romance** | 1 | 0 | 1 | 0 | 1 | 0 |

# Keyword-based Vector Space Model

- Imagine each user is represented by a binary-valued row vector of her tastes. These vectors also have dimension ***d***, and the elements correspond to the ones used for items.
- We can gather these vectors into a matrix, which we will refer to as ***P***
  - So, ***P*** is a ***| U | x d*** matrix.
  - If we want to refer to the row in ***P*** that corresponds to user ***u***, we will write $P_u$

| | comedy | thriller | romance |
|---|---|---|---|
| $u_1$ | 0 | 1 | 0 |
| $u_2$ | 1 | 1 | 1 |
| $u_3$ | 0 | 0 | 0 |
| $u_4$ | 1 | 0 | 1 |

# Keyword-based Vector Space Model

- The score that capture the relevance to user ***u*** of item ***i*** is simply the similarity of vectors $Q_i$ and $P_u$
- We can use cosine similarity for this (ignoring normalization). This is simply the product of the two vectors.

$$sim(u,i) = P_u . Q_i$$

| | comedy | thriller | romance |
|---|---|---|---|
| $u_1$ | 0 | 1 | 0 |
| $u_2$ | 1 | 1 | 1 |
| $u_3$ | 0 | 0 | 0 |
| $u_4$ | 1 | 0 | 1 |

| | $i_1$ | $i_2$ | $i_3$ | $i_4$ | $i_5$ | $i_6$ |
|---|---|---|---|---|---|---|
| comedy | 1 | 0 | 0 | 1 | 1 | 0 |
| thriller | 0 | 0 | 0 | 0 | 1 | 1 |
| romance | 1 | 0 | 1 | 0 | 1 | 0 |

# Bag-of-Words Representation

## Sets

- A set is a collection of objects with two properties:
  - Order is not important, e.g. {a, c, f} = {c, f, a}
  - Duplicates are not allowed, e.g. {a, c, f} = {a, c, a, f}
- The set of all possible objects ***U*** is called the universal set, e.g. ***U*** = {a, b, c, d, e, f}
- But, we can describe a set by a ***binary valued vector,*** e.g. for the above universal set ***U*** we can represent {a, c, f}

| *a* | *b* | *c* | *d* | *e* | *f* |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 |

If ***U*** is large and the sets we store tend to be much smaller, then our vectors will be ***sparse*** (mostly zero).

# Bag-of-Words Representation

## Bags

- A bag is a collection of objects with one property:
  - Order is not important, e.g. {a, c, f} = {c, f, a}
  - Duplicates are allowed, e.g. {a, c, f} ≠ {a, c, a, f}
- We can describe a ***bag*** by a ***numeric-valued vector***, where the numbers are the ***frequencies*** with which the elements occur,

  e.g. for the universal set ***U*** we can represent {a, c, a, f}

| ***a*** | ***b*** | ***c*** | ***d*** | ***e*** | ***f*** |
|---|---|---|---|---|---|
| 2 | 0 | 1 | 0 | 0 | 1 |

We can represent each document by a ***bag-of-words***.

# Bag-of-Words Representation

**Running Example**

- Suppose our dataset contain three documents (i.e. three tweets from Barack Obama, quoting Nelson Mandela)
  - **Tweet 0**: No one is born hating another person because of the color of his skin or his background or his religion.
  - **Tweet 1**: People must learn to hate, and if they can learn to hate, they can be taught to love.
  - **Tweet 2**: For love comes more naturally to the human heart than its opposite.

# Bag-of-Words Representation

**Tokenization**

- First, we must tokenize each document. This means splitting it into tokens.
- In our simple treatment, the tokens are just the words, ignoring punctuation and making everything lowercase.
- In reality, tokenization is surprisingly complicated,
  - e.g. is "don't" one token or two or three?
  - e.g. maybe pairs of consecutive words (so-called 'bigrams') could also be treated as if they were single tokens ("no one", "one is", "is born")
  - and so on.

# Bag-of-Words Representation

**Stop-words**

- Optionally, discard stop-words:
- Stop-words are common words such as "a", "the", "in", "on", "is, "are",…
- Sometimes discarding them helps, or does no harm, e.g. spam detection.
- Other times, you lose too much, e.g. web search engines ("To be, or not to be").

# Bag-of-Words Representation

**Running Example**

- After tokenization and stop-words
  - **Tweet 0**: born hating person color skin background religion
  - **Tweet 1**: people learn hate learn hate taught love
  - **Tweet 2**: love comes naturally human heart opposite

# Bag-of-Words Representation

**Stemming or Lemmatization**

- Optionally, apply stemming or lemmatization to the tokens.

  e.g. "hating" is replaced by "hate", "comes" is replaced by "come"

# Just FYI. . .

**Stemming vs. Lemmatization**

Stemmer operates on a single word without knowledge of the context, and therefore cannot discriminate between words which have different meanings depending on part of speech. For example,

- The word "better" has "good" as its lemma. This link is missed by stemming, as it requires a dictionary look-up.
- The word "walk" is the base form for word "walking", and hence this is matched in both stemming and lemmatization.
- The word "meeting" can be either the base form of a noun or a form of a verb ("to meet") depending on the context, e.g., "in our last meeting" or "We are meeting again tomorrow". Unlike stemming, lemmatization can in principle select the appropriate lemma depending on the context.

# Bag-of-Words Representation

## Count Vectorization

- Each document becomes a vector,
- each token becomes a feature,
- feature-values are frequencies (how many times that token appears in that document).

## Running example

| | background | born | color | comes | hate | hating | heart | human | learn | love | naturally | opposite | people | person | religion | skin | taught |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Tweet 0:** | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| **Tweet 1:** | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| **Tweet 2:** | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

# Bag-of-Words Representation

## TF-IDF Vectorization

- Optionally, replace the frequencies by TF-IDF scores.
- Variants of the formula might:
  - scale frequencies to avoid biases towards long documents;
  - logarithmically scale frequencies;
  - add 1 to part of the formula to avoid division-by-zero;
  - normalize the results (e.g. divide by the L2-norm)

## Running example

| | background | born | color | comes | hate | hating | heart | human | learn | love | naturally | opposite | people | person | religion | skin | taught |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Tweet 0:** | 0.38 | 0.38 | 0.38 | 0 | 0 | 0.38 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.38 | 0.38 | 0.38 | 0 |
| **Tweet 1:** | 0 | 0 | 0 | 0 | 0.61 | 0 | 0 | 0 | 0.61 | 0.23 | 0 | 0 | 0.31 | 0 | 0 | 0 | 0.31 |
| **Tweet 2:** | 0 | 0 | 0 | 0.42 | 0 | 0 | 0.42 | 0.42 | 0 | 0.32 | 0.42 | 0.42 | 0 | 0 | 0 | 0 | 0 |

# Bag-of-Words Representation

**Curse of Dimensionality**

- Reduce the number of features by:
  - discarding tokens that appear in too few documents;
  - discarding tokens that appear in too many documents;
  - keeping only the most frequent tokens.
- Use dimensionality reduction:
  - e.g. singular value decomposition (SVD) is suitable for bag-of-words, rather than PCA.

May be bcoz of sparsity..check??

# Bag-of-Words Representation

This representation is good for many applications in AI but it does have drawbacks too:

- It loses all the information that English conveys through the order of words in sentences,
  - e.g. "People learn to hate" and "People hate to learn" have very different meanings but end up with the same bag-of-words representation.
- It loses the information that English conveys using its stop-words, most notably negation,
  - e.g. "They hate religion" and "I do not hate religion" will have the same bag-of-words representation.
- This may not matter for some applications (e.g. spam detection) but will matter for others (e.g. machine translation)

# Embeddings

Embedding methods (alternatively referred to as “encoding”, “vectorising”, etc) convert -

- symbolic representations (i.e. words, emojis, categorical items, dates, times, other features, etc)
- into meaningful numbers (i.e. real numbers that capture underlying semantic relations between the symbols).
- e.g. representing colors in terms of colors of rainbow (one-hot encoding) representing colors in terms of vectors of RGB values
    - Different variations can be represented in a smaller size embedding
    - Red and Green will have large differences while Red and Orange will be relatively close.

***Similarly, we can create fixed-length vectors that can represent words.***

# Word Embeddings

Suppose, a word vector of 100 values can represent 100 unique features (a feature can be anything that relates words to one another), for example -

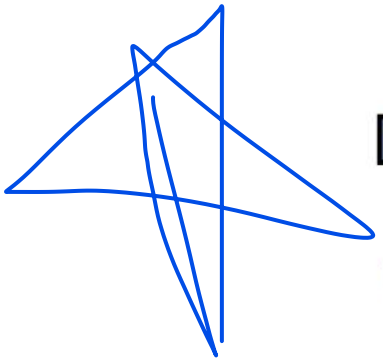
- A common Theme / Topic such as“Royalty”, “Masculinity”, “Femininity”, “Age”, etc.).
- Words are then represented as a distribution of its membership to each of the features.

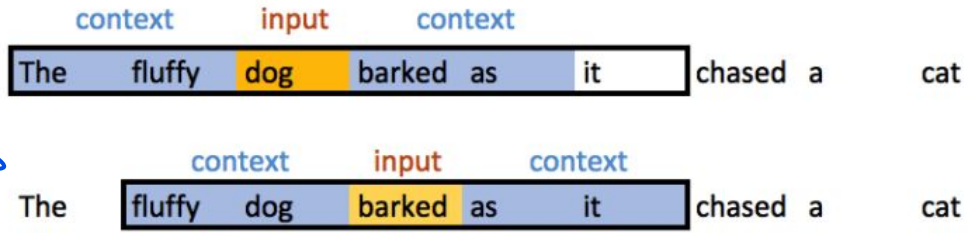
# Word Embeddings

## Co-occurrence Matrix

Words co-occurrence matrix describes how words occur together that in turn captures the relationships between words.

- This is computed simply by counting how two or more words occur together in a given corpus (within a fixed-size window).
- This would mean that the word vector's features are also words themselves.





# Word Embeddings

## Co-occurrence Matrix

Words co-occurrence matrix describes how words occur together that in turn captures the relationships between words.

- This is computed simply by counting how two or more words occur together in a given corpus.
- This would mean that the word vector's features are also words themselves.

| Terms | Terms data | examples | introduction | mining | network | package |
|---|---|---|---|---|---|---|
| data | 53 | 5 | 2 | 34 | 0 | 7 |
| examples | 5 | 17 | 2 | 5 | 2 | 2 |
| introduction | 2 | 2 | 10 | 2 | 2 | 0 |
| mining | 34 | 5 | 2 | 47 | 1 | 5 |
| network | 0 | 2 | 2 | 1 | 17 | 1 |
| package | 7 | 2 | 0 | 5 | 1 | 21 |

Co-occurrence matrix is decomposed using techniques like PCA, SVD etc. into factors and combination of these factors forms the word vector representation.

# Word Embeddings

## Prediction-based Methods

***CBOW (Continuous Bag-of-Words)*** is learning to predict the word by the context.

- A context may be single word or multiple word for a given target word.
  e.g *“The cat jumped over the puddle.”*
- So CBOW treats {*“The”, “cat”, ’over”, “the’, “puddle”*} as a context and from these words, it predicts or generates the center word *“jumped”*.

# Word Embeddings

## Prediction-based Methods

***Skip-gram Model*** is learning to predict or generate the surrounding words *“The”, “cat”, “over”, “the”, “puddle”* given iven the word *“jumped”*.

- Here the word *“jumped”* is the context.

# Word Embeddings

## Learning Word Embeddings

We learn the word embeddings from the dataset of documents.

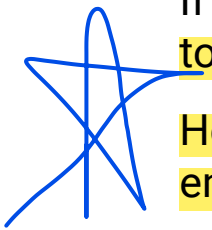
Conceptually,

- The values in the vectors are initialized randomly;
- Then they are adjusted during learning.

For example, **Keras** does it using a network layer:

- During learning, the weights of the layer are adjusted.
- The activations of the units in the layer are the word embeddings.

If we learn embeddings on our own dataset, they will be tailored to helping us to perform our specific task.



However, the dataset should be sizable enough to learn really powerful word embeddings.

# Word Embeddings

## Pre-trained Word Embeddings

To some extent, word embeddings are fairly generic, so it can make sense to reuse pretrained embeddings from very large datasets.

- **word2vec** This is Google's famous algorithm for learning word embeddings – pretrained embeddings learned from news articles.
- **GloVe (Global Vectors for Word Representation)**: This is a Stanford University algorithm – pretrained embeddings learned from Wikipedia.

It takes a large body of text (e.g. Wikipedia) and builds a model (a two-layer neural network classifier) that predicts words.

**Next lecture -**

Content-based Recommendations