Lecture 03

- Content-based Methods
- Representation of the Text

- Content-based Methods
- Representation of the Text

Classic Mathematical Definition

Let $I = \{i_1, i_2, i_3, \ldots, i_n\}$ be a set of ***items*** and

$U = \{u_1, u_2, u_3, \ldots, u_m\}$ be a set of ***users***.

A recommender system attempts to find an item $i^* \in I$ for user $u \in U$ such that the *utility* of item i^* for user u, $utility(u, i^*)$, is maximum:

$$i^* = \underset{i \in I}{\arg\max}\ utility(u, i)$$

Content-based Methods: 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.

Item Representation

- **Structured**: a finite and typically small set of attributes
  - e.g., for products: size, weight, manufacturer, etc.
    for movies: director, length, language, guidance certificate, etc.
    for songs: artist, producer, record label, etc.

- **Unstructured**: no explicit structure, ***often processed to obtain meaningful information***
  - e.g., keywords extracted from a movie description or user reviews;
    user assigned tags to an item;

- **Semi-structured**: mixture of structured and unstructured information
  - e.g. movie genres (comedy, thriller, romance, …) with movie keywords

Item Representation

Source of Information for representing items -

- **Attribute-value pairs**: in terms of values for predefined attributes.
- **Item content**: content (textual descriptions) can be mined for *keywords*
- **User reviews**: user experiences or opinions about a product or service in the form of reviews, can be mined for *aspects, context,* etc.
- **User assigned tags**: can be characterized as *objective* (where they convey factual information about an item) or *subjective* (where they express the user's opinion about an item).
- **Linked data**: is inter-connected data that is published in a way that complies with the Linked Data principles. e.g., DBpedia

User Profile

A user's profile represents her ***preferences***.

User preferences may be -

- ***Persistent***, indicating a *user's long-term tastes and interests*, or
- ***Ephemeral***, reflecting her *transient (or short-term) requirements*

A user profile is most typically a representation of her persistent preferences.

User Profile

A user profile simply consists of a history of the user's interactions with the recommender system.

For example -

- ***items*** she has clicked or viewed or purchased, or ratings that she has given to the items;
- ***features*** that describe the user's tastes and interests (obtained from a sign-up form or aggregated from the items the user interacts with)

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

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 an ***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

Memory-based Recommendation

In general, steps to follow:

- The ***score*** can be calculated for all the candidate items for all the users.
- Arrange the candidates in a non-increasing order of their score for each user.
- Suggest top-k (e.g., k = 3 or 5 or 10) items to the user.

- Content-based Methods
- Representation of the Text
  - Text Vectorization

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"

Word	***Lemmatization***	***Stemming***
was	be	wa
studies	study	studi
studying	study	study

Bag-of-Words Representation

Stemming vs. Lemmatization

- Stemming is a process that stems or removes last few characters from a word based on a few rules.
- Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma.
    - e.g., stemming the word ‘Caring‘ would return ‘Car‘ while lemmatizing the word ‘Caring‘ would return ‘Care‘.
- Stemming is used in case of large dataset where performance is an issue, whereas Lemmatization is computationally expensive since it involves look-up tables or dictionary.

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

Term Weighting

The most commonly used term weighting scheme, ***TF-IDF (Term Frequency– Inverse Document Frequency)*** which relies on the following assumptions -

- rare terms are not less relevant than frequent terms (IDF assumption);
- multiple occurrences of a term in a document are not less relevant than single occurrences (TF assumption);
- long documents are not preferred to short documents (normalization assumption).

Term Weighting

Term Frequency– Inverse Document Frequency, i.e., TF-IDF

$$\text{TF-IDF}(t_k, d_j) = \underbrace{\text{TF}(t_k, d_j)}_{\text{TF}} \cdot \underbrace{log\frac{N}{n_k}}_{\text{IDF}}$$

$$\text{TF}(t_k, d_j) = \frac{f_{k,j}}{max_z f_{z,j}}$$

Normalized TF-IDF

$$w_{k,j} = \frac{\text{TF-IDF}(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|} \text{TF-IDF}(t_s, d_j)^2}}$$

Term Weighting

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

Measuring Item Similarity

Similarity between two items represented by the constituent terms -

$$sim(d_i, d_j) = \frac{\sum_k w_{ki} \cdot w_{kj}}{\sqrt{\sum_k w_{ki}^2} \cdot \sqrt{\sum_k w_{kj}^2}}$$

Memory-based Recommendation

In general, steps to follow:

- The ***score*** can be calculated for all the candidate items for all the users.
- Arrange the candidates in a non-increasing order of their score for each user.
- Suggest top-k (e.g., k = 3 or 5 or 10) items to the user.

Bag-of-Words Representation Revisited

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.

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.

Bag-of-Words Representation

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

- 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.

Bag-of-Words Representation

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

- order of words in sentences,
- its stop-words, most notably negation,

This may not matter for some applications (e.g. spam detection) but will matter for others (e.g. machine translation)

- Content-based Methods
- Representation of the Text
  - Word Embeddings

Embeddings: Motivation

- So far, we have represented each document as a single vector (bag-of-words).
  - This is OK for some applications, e.g. spam filtering.

- But for many applications of natural language processing (NLP), we may need to treat ***documents as sequences (lists) of words*** (and maybe of punctuation symbols also):
  - Sentiment analysis, e.g. of movie reviews or tweets;
  - Machine translation;
  - Image captioning;
  - Question-answering and chatbots.

Embeddings: Motivation

There are other applications where each example is a ***sequence of features*** too, e.g.:

- processing speech;
- processing other audio, such as music;
- processing genomic data;
- time-series prediction;
- clickstream prediction

Embeddings: Motivation

Examples

- 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 dense vectors that can represent words.

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.

Embeddings

We could randomly assign numbers to words –

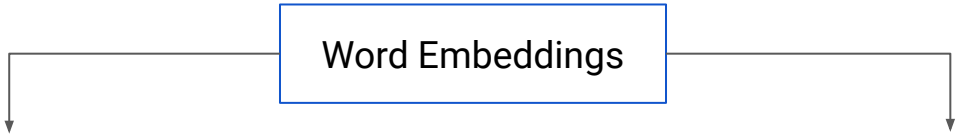
- This may not capture their semantic relationships –
  - e.g., ‘great’ and ‘awesome’ though are similar and used in similar ways; may be assigned 4.2 and -42.1 respectively.
- A word may be used in different contexts (i.e., nearby words) or in different forms (e.g., plural) or in different ways (e.g., positive or negative).
  - A multidimensional representation can be more beneficial to easily adjust to different contexts.

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



Frequency-based Embedding

Focus is on how many times a word appears in a given text.

- Term-document type
  - Count Vectorization
  - TF-IDF Vectorization
- Term-term type
  - Co-occurrence based Vectorization

Prediction-based Embedding

Focus is on predicting a word that best fits to a given context

Word2Vec (window size << |d|)

- Continuous Bag of Words (CBOW)
- Skip-gram

- GloVe (pre-trained) combines the two types — DIY!!

Co-occurrence based Vectorization

Co-occurrence Matrix

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

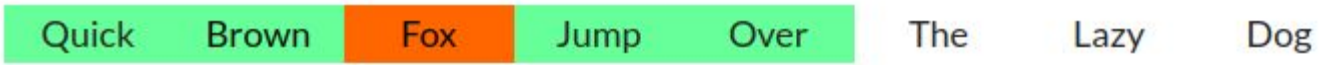
- For a given corpus, the co-occurrence of a pair of words say w_1 and w_2 is the number of times they have appeared together in a ***Context Window***.

Co-occurrence based Vectorization

- **Context Window**: Context window is specified by ***a number*** and the ***direction***.

  So what does a context window of 2 (around) means?

  - Context Window of 2 (around) for the word “Fox”



  - Context Window of 2 (around) for the word “Over”

Co-occurrence based Vectorization

Co-occurrence Matrix

- This is computed simply by counting how many times two or more words occur together in a given corpus.

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

- For a larger vocabulary, such vectorization may be quite sparse and may need large amount of storage capacity.

Co-occurrence based Vectorization

Co-occurrence Matrix

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

$\hat{X}$ is the best rank k approximation of X in terms of least squares.

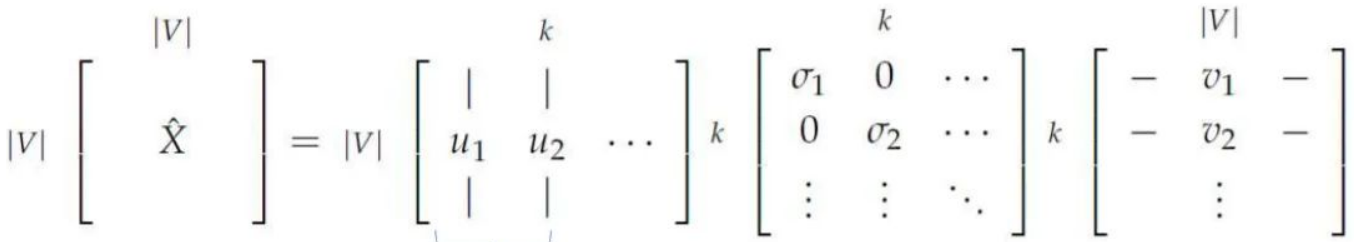
$$\underset{|V|}{\underset{|V|}{\begin{bmatrix} \\ X \\ \\ \end{bmatrix}}} = \underset{|V|}{\underset{|V|}{\begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix}}} \underset{|V|}{\underset{|V|}{\begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}}} \underset{|V|}{\underset{|V|}{\begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix}}}$$

$$\underset{|V|}{\underset{|V|}{\begin{bmatrix} \\ \hat{X} \\ \\ \end{bmatrix}}} = \underset{|V|}{\underset{k}{\begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix}}} \underset{k}{\underset{k}{\begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}}} \underset{k}{\underset{|V|}{\begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix}}}$$

Co-occurrence based Vectorization

Co-occurrence Matrix

- The *dot product of U and S gives the word vector representation* and *V gives the word context representation*.



$$\underset{|V|}{\begin{bmatrix} \\ \hat{X} \\ \\ \end{bmatrix}} = \underset{|V|}{\begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix}} \underset{k}{} \underset{k}{\begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}} \underset{k}{} \underset{|V|}{\begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix}}$$

$$\begin{bmatrix} u_{11}\sigma_1 & u_{12}\sigma_2 & \dots & u_{1k}\sigma_k \\ \vdots & & & \\ u_{m1}\sigma_1 & u_{m2}\sigma_2 & \dots & u_{mk}\sigma_k \end{bmatrix}$$

Each row of this matrix (U dot product S) is a word embedding

Co-occurrence based Vectorization

Improvements

- Ignore words that are too frequent, or too unique.
- ***Ramped windows*** that count closer words more
- Use Pearson correlation instead of counts, and set negative values to 0
- ...

Prediction-based Methods: Word2Vec

Learn word representations as a part of a simple neural network architecture for language modeling.

- We train a simple neural network with a single hidden layer to perform a certain task: predict the target word that best fits the given context.
- However, we are not actually going to use that neural network for the task we trained it on!
- Instead, the goal is just to learn the weights of the hidden layer — we will see that these weights are actually the “word vectors” that we are trying to learn.

Word2Vec: Skip-gram

Let’s take an example:

- “The man who passes the sentence should swing the sword.”

Sliding window (size = 2)	Target word	Context
[The man who]	the	man, who
[The man who passes]	man	the, who, passes
[The man who passes the]	who	the, man, passes, the
[man who passes the sentence]	passes	man, who, the, sentence
…	…	…
[sentence should swing the sword]	swing	sentence, should, the, sword
[should swing the sword]	the	should, swing, sword
[swing the sword]	sword	swing, the

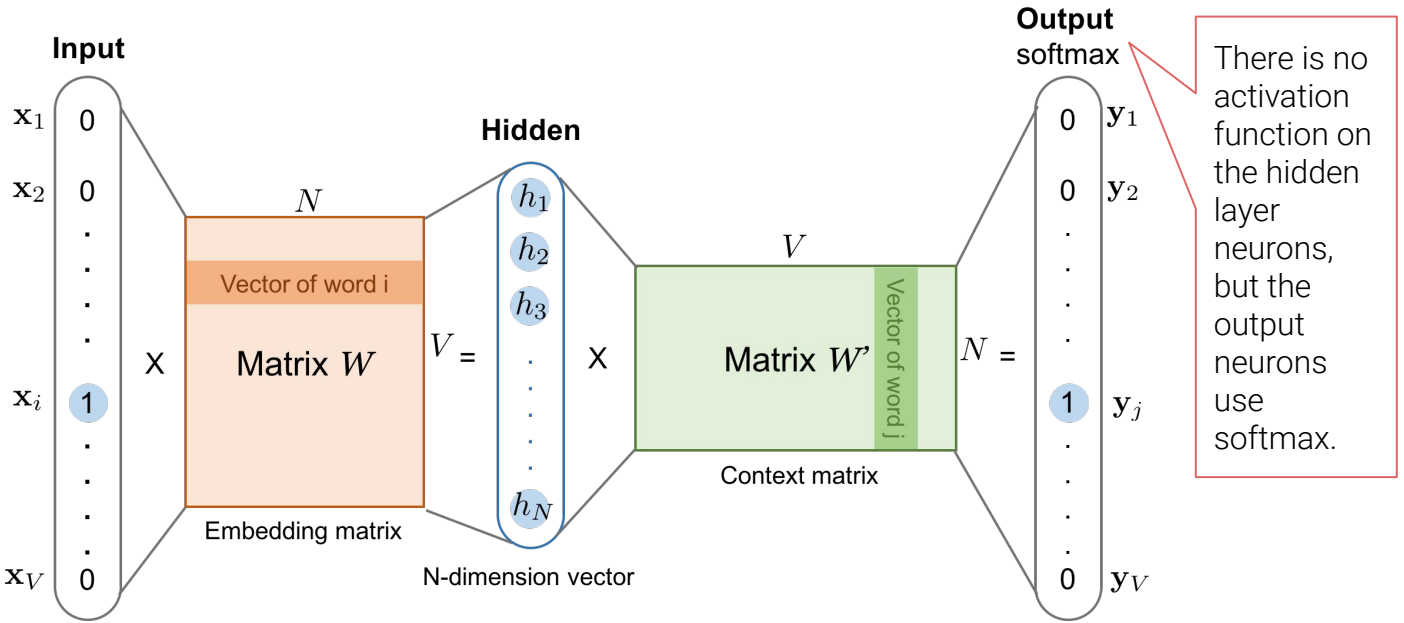
{:.info}

Word2Vec: Skip-gram

- Each ***context-target*** pair is treated as a new observation in the data.
- For example, the target word “swing” in the above case produces four training samples:
  - (“swing”, “sentence”),
  - (“swing”, “should”),
  - (“swing”, “the”), and
  - (“swing”, “sword”).
- The model learns to predict one context word (output) using one target word (input) at a time.

Word2Vec: Skip-gram

In the following skip-gram model, both the input vector ***x*** and the output ***y*** are one-hot encoded word representations. The hidden layer is the word embedding of size ***N***.



Word2Vec: Skip-gram

That one-hot vector is almost all zeros… what's the effect of that?"

- For example, If you multiply a 1 x 5 one-hot vector by a 5 x 3 matrix, it will effectively just select the matrix row corresponding to the "1".

$$\begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix} \times \begin{bmatrix}17 & 24 & 1\\ 23 & 5 & 7\\ 4 & 6 & 13\\ 10 & 12 & 19\\ 11 & 18 & 25\end{bmatrix} = \begin{bmatrix}10 & 12 & 19\end{bmatrix}$$

- This means that the hidden layer of this model is really just operating as a lookup table.
- The output of the hidden layer is just the "word vector" for the input word.

Word2Vec: Skip-gram

- The multiplication of the hidden layer and the word context matrix ***W'*** of size **N x V** produces the output one-hot encoded vector ***y***.
- The output context matrix ***W'*** encodes the meanings of words as context, different from the embedding matrix ***W***.

Note that despite the name, ***W'*** is independent of ***W***, not a transpose or inverse.

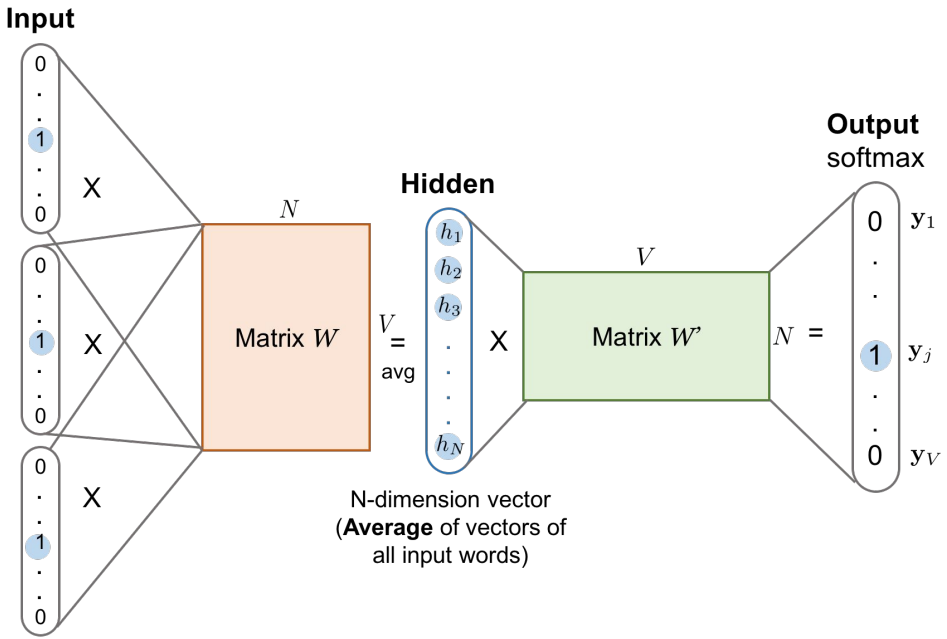
Word2Vec: Continuous Bag-of-Words (CBOW)

The Continuous Bag-of-Words (CBOW) is another similar model for learning word vectors.

- It predicts the target word (i.e., “swing”) from source context words (i.e., “sentence should the sword”).

Word2Vec: Continuous Bag-of-Words (CBOW)

Because there are multiple contextual words, we average their corresponding word vectors, constructed by the multiplication of the input vector and the matrix ***W***.



Word2Vec

- Actually, Word2Vec was trained on the Google News articles comprising of 3M words vocabulary that resulted in word vectors of size 300.
- In this case, the size of the embedding matrix is ***W (3000000 x 300)*** and of the context matrix is ***W' (300 x 3000000),*** therefore, in total **1.8B weights** need to be updated in each iteration of the Backpropagation — slows down the training!
- To speed up the training, they applied ***Negative Sampling***: randomly selecting a subset of words we don't want to predict for optimization.

Word2Vec

Example of Negative Sampling:

- There are two “problems” with common words like “the”:
  - When looking at word pairs, (“passes”, “the”) doesn’t tell us much about the meaning of “passes”. “the” appears in the context of pretty much every word.
  - We will have many more samples of (“the”, …) than we need to learn a good vector for “the”.
- If we have a window size of 10, and we remove a specific instance of “the” from our text:
  - As we train on the remaining words, “the” will not appear in any of their context windows.
  - We’ll have 10 fewer training samples where “the” is the input word.

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

- Examples of Information Extraction from the Text