Big Data Analytics

Lecture 10:

Word Embedding Models.

A non-technical introduction

References

- ✓ Rodriguez Pedro L. and Spirling Arthur (2021). Word Embeddings: What works, what doesn’t, and how to tell the difference for applied research, *Journal of Politics*, forthcoming
- ✓ Rudkowsky Elena, et al. (2018). More than Bags of Words: Sentiment Analysis with Word Embeddings. *Communication Methods and Measures*. 12:2-3, 140-157

Beyond bag-of-words

Bag-of-words representation of texts

- ✓ Focusing on frequency of features
- ✓ Ignoring context, grammar, word order

Bag-of-words approach discards therefore much linguistic information regarding the **surrounding syntactic and semantic context** of a given word in a sentence. This is wrong, but often useful!

How bringing back the context in which a word appears?

One alternative: **word embeddings**

What is word-embedding?

It’s all began with…: “*You shall know a word by the company it keeps*” (Firth, 1957)

More concretely, a word’s meaning can be garnered from its contextual information: literally, the other words that appear near it in text

A “context” or “window” here means a symmetric window of terms around the word of interest

What is word-embedding?

Let’s consider the following three sentences:

1. *I like deep learning.*
2. *I like NLP.*
3. *I enjoy flying.*

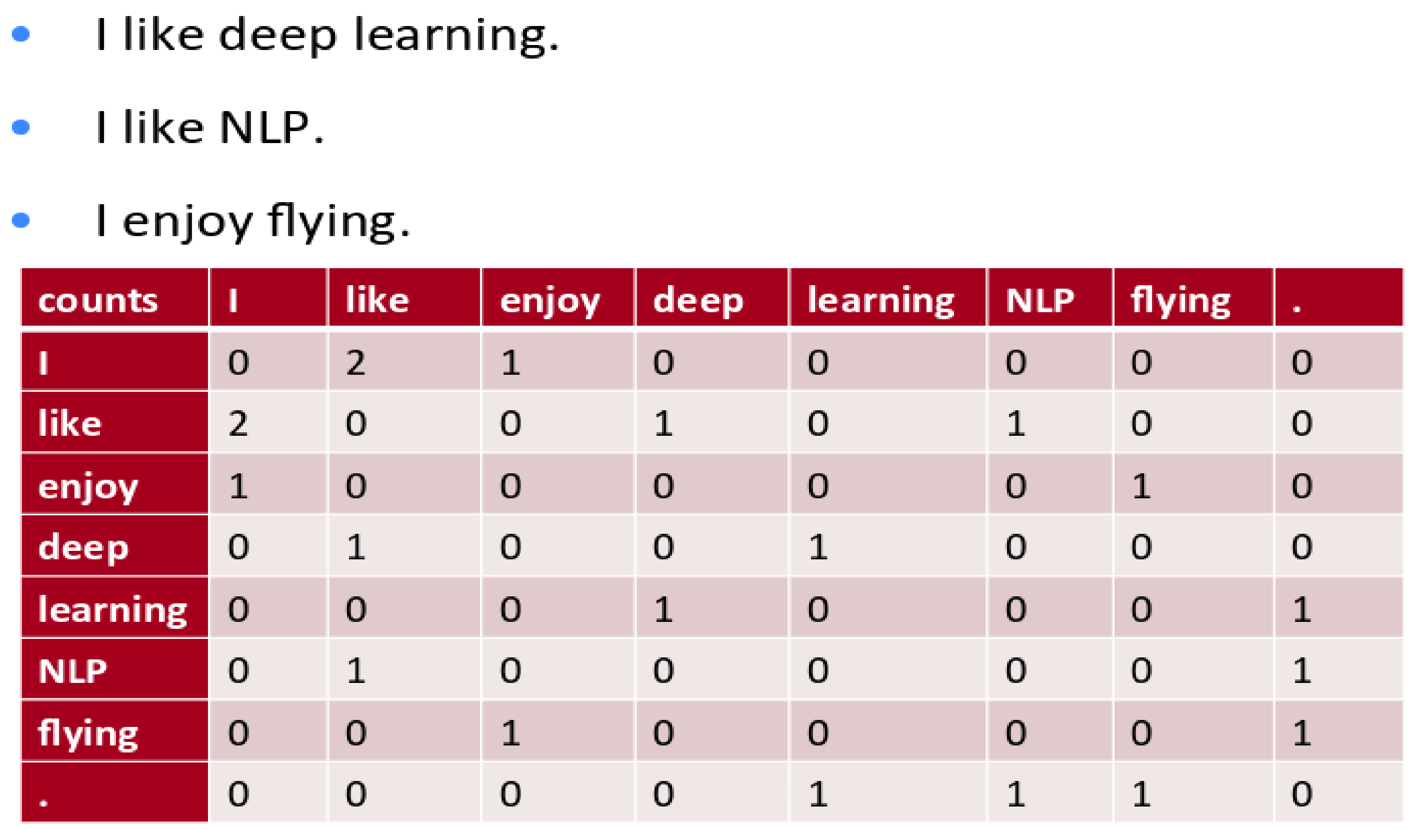
Let’s create a **co-occurrence matrix** with a *window-size*=1

If there are kk features, a **co-occurrence matrix** is a KK x KK matrix, where each cell tells us how frequently word$_i$ occurs with word$_j$ in some window

In such instance the *window-size* determines the number of words, on either side of the focus word to be included in its context

Let’s consider a window size of 1

What is word-embedding?

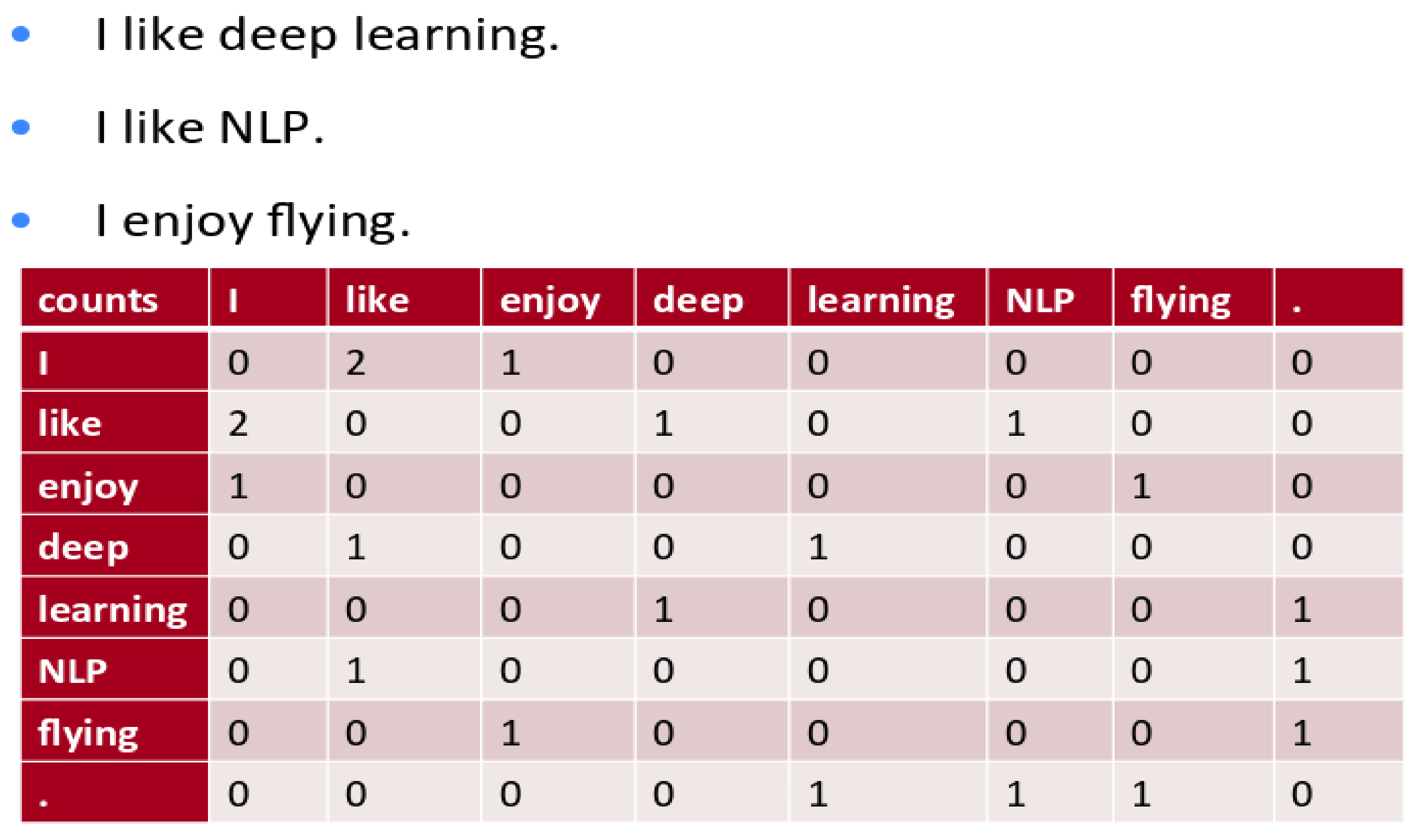


- I like deep learning.
- I like NLP.
- I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

As it is easy to guess, as we increase our corpus size, the semantically similar words will have similar looking vectors (i.e., similar looking rows) to each other (at least more similar looking than the semantically dissimilar words)

What is word-embedding?



- I like deep learning.
- I like NLP.
- I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Formalizing this idea, the “*distributional hypothesis*” suggests that words which appear in similar contexts are likely to share similar meanings (Harris, 1970)

What is word-embedding?

Compare it with a bag-of-words approach:

	I	like	enjoy	deep	learning	NLP	flying	.
Doc1	1	1	0	1	1	0	0	1
Doc2	1	1	0	0	0	1	0	1
Doc3	1	0	1	0	0	0	1	1

The main difference between these two matrices is the **type of contextual information** they use

- A DfM use *documents as contexts*
- A co-occurrence matrix instead use *words as contexts*, which can be considered as more natural from a linguistic perspective

But what should we do with a co-occurrence matrix?

What is word-embedding?

The goal is to map a **co-occurrence matrix** into a given number of dimensions, such that the **semantically similar** words are mapped close to each other

We could for example perform some matrix factorization technique (FA or PCA for example) to reduce the *dimensionality of the co-occurrence matrix*

What is word-embedding?

If, for example, we apply some matrix factorization technique to a co-occurrence matrix, we can reduce its dimensionality by a lot…

And if we do it, for each word we would have a *vector* of numeric values to represent them

For example, if we extract 4-dimensions out of a co-occurrence matrix, the word “like" now will be represented as a 4-dimensional vector which may look like (-0.4, 0.3, 0.6, -0.3). And this is true for each other word included in our corpus

The length of this vector corresponds to the nature and complexity of the multidimensional space in which we are seeking to **embed** the word (more on this later on)

What is word-embedding?

Word embeddings is therefore a way to **transform text into features**

Instead of using vectors of word counts (i.e., our traditional bag-of-words approach), words are now represented as **positions on a latent multidimensional vector space**

In such space, words that are mapped closer to each other have similar meaning (remember the *Distributional hypothesis*: words that appear in the same context share semantic meaning!)

Through that, we can give a more meaningful numeric representation of a text that contains both information regarding the word itself as well as information regarding the *linguistic context* of that word

What is word-embedding?

Newer techniques such as *word2vec* and *GloVe* (more on this in the Lab) use neural net approaches (i.e., specific ML algorithms) to construct word vectors rather than PCA or FA

`word2vec` model uses a neural network model to estimate the probability that a word is “close” to another given word

the `GloVe` model (“Global VEctors of words”) predicts surrounding words using a form of dynamic logistic regression

What is word-embedding?

The details are not however (too much) important for applied users to benefit from them. The aim is always the same: building a low-dimensional vector representation from corpus of text, which preserves the contextual similarity of words

Any differences?

While `GloVe` explicitly underweights relatively rare terms, `Word2Vec` explicitly underweights high frequency terms. Consequently, `Word2Vec` often picks out relatively rare terms (including misspellings) as nearest neighbors

In practice this means `Word2Vec` is likely to be less “robust,” i.e., embeddings will tend to be more corpus specific, than `GloVe`

Parameter Choices

You should be called, at the very least, to decide about 3 parameters in any embedding model:

1. how large a **window size** you want the model to use
2. how large an **embedding** you wish to use to represent your words (i.e., the number of extracted dimensions)
3. whether to fit the embedding models **locally**, or to use pre-trained embeddings fit to some other (hopefully related) corpus

Window-size

Window-size determines the number of words, on either side of the focus word to be included in its context

Intuitively, larger windows provide more information to discriminate between different words

Take, for example, the following two sentences: “cows eat grass” and “lions eat meat”. A window-size of 1 does not provide enough information to distinguish between cows and lions (we know they both eat, but we don’t know what) whereas a window size of 2 does

Larger windows (usually 5 or above) tend therefore to produce better quality embeddings although with decreasing returns

Embedding dimensions

This parameter determines the **dimensions** of the embedding vectors which usually range between 50-450

We can think of these dimensions as capturing different aspects of “meaning” or semantics that can be used to organize words

Though none of the dimensions of the estimated vector are named, the “loading” of each term on the dimensions indeed often captures substantively important relationships

Embedding dimensions

For instance, the words “king” and “queen” might have a similar concentration on a dimension that seems to relate to the concept of royalty, but deviate on a dimension that seems to relate to gender

The coordinates of the dimensional-vector represent in other words **distinct topics**

Embedding dimensions

Too few dimensions - imagine the extreme of 1 - and there can be no meaningful separability of words; too many, and some dimensions are likely to be redundant (go unused)

Factors such as vocabulary size and topical specificity of the corpus are likely to play a role, although theoretical work in this area remains scant

Pre-Trained Versus Going Local

Embedding models can be data hungry, meaning they need a lot of data to produce ‘useful’ results

Consequently, researchers with small corpora rather than building word-embeddings **locally** out of their corpus, can decide to use generic pre-trained embeddings trained on much larger corpora

These off-the-shelf embedding corpora can be used like dictionaries, but instead of translations or meanings, they return vector embeddings for the requested words

Therefore, the usage of pre-trained embeddings does not require any further computing time

Pre-Trained Versus Going Local

For example, pre-trained word vectors are available that have been estimated from large corpora, such as that trained on six billion tokens from Wikipedia and the “Gigiword” corpus (Pennington, Socher and Manning, 2014)

This allows a researcher to represent his or her texts not just from the corpus at hand, but also augmented with quantitative measures of the words’ semantic representations fitted from other contexts

Pre-Trained Versus Going Local

However, there are trade-offs

Intuitively, we would want to use pre-trained embeddings trained on a corpus generated by a similar “language model” to that which generated our corpus of interest. The more similar the two language models, the more similar the underlying semantics

For a highly specific local corpus - a corpus in Old English for example - generic pre-trained embeddings may not be all that useful!

What for Word embedding models?

Operations with word-vectors are meaningful per-se, i.e., embeddings can be a **direct object of interest** for studying word usage and meaning

For example, the similarities between word embeddings may be substantively informative: if the distance between “immigrants” and “hardworking” is smaller for liberals than for conservatives, we learn something about their relative worldviews

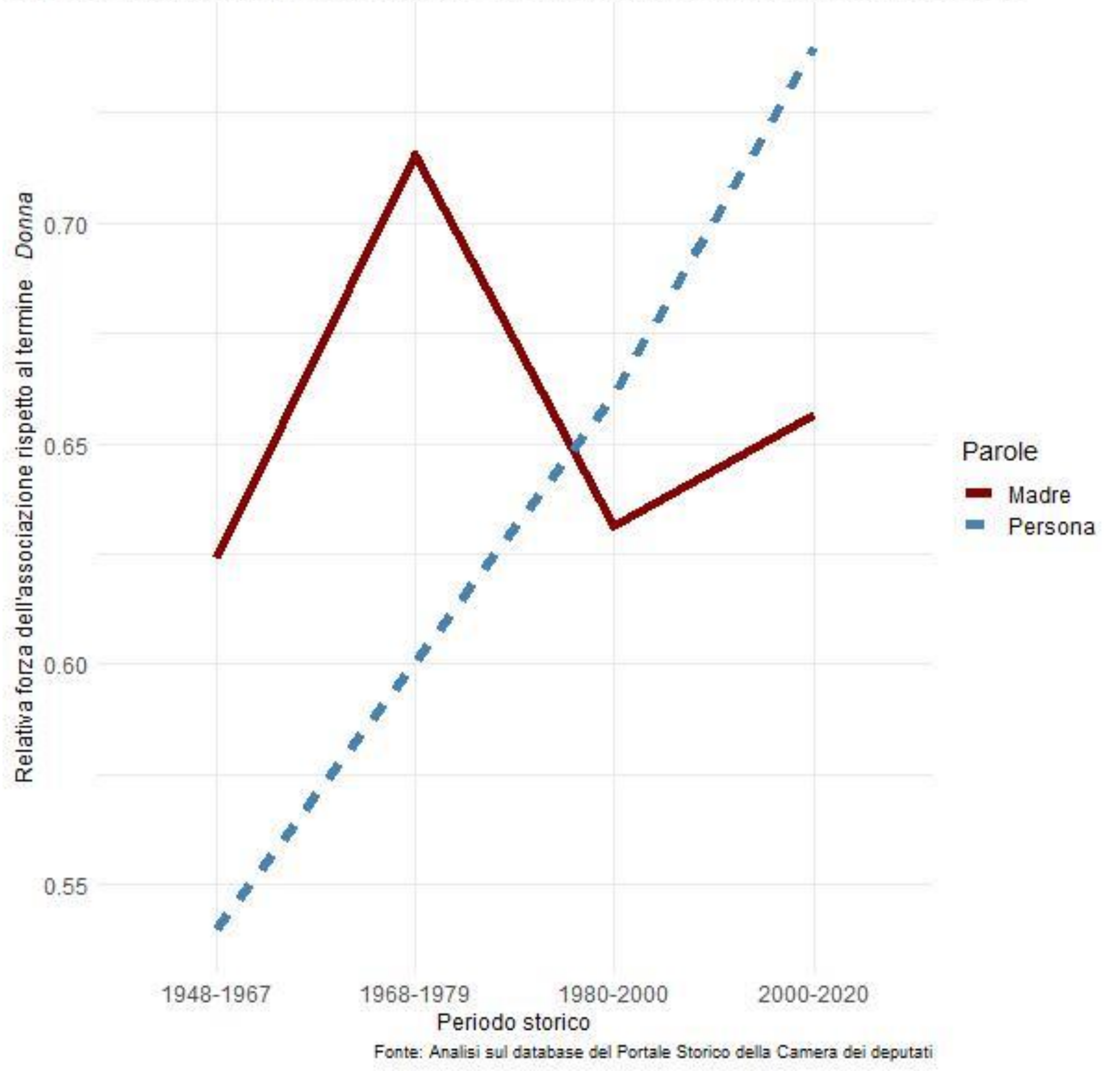
What for Word embedding models?

This also allows us to explore the possible existence of societal *stereotypes* encoded in the language

To take an example, if you analyze the Italian legislative speeches since 1948 to nowadays, the most similar word to “woman” in the ‘50s was “mother” and “wife”; then over the years it changed to “worker” and “person”

What for Word embedding models?

Fig.3: Associazione tra Donna vs. Madre e Persona nel Parlamento italiano 1948-2020



What for Word embedding models?

You can also do addition in a word embeddings framework to explore *analogies*

Analogies are **equations** of the form *A is to B as C is to D*

Given the terms A, B, C, the model must return the word that correctly stands for D in the given analogy

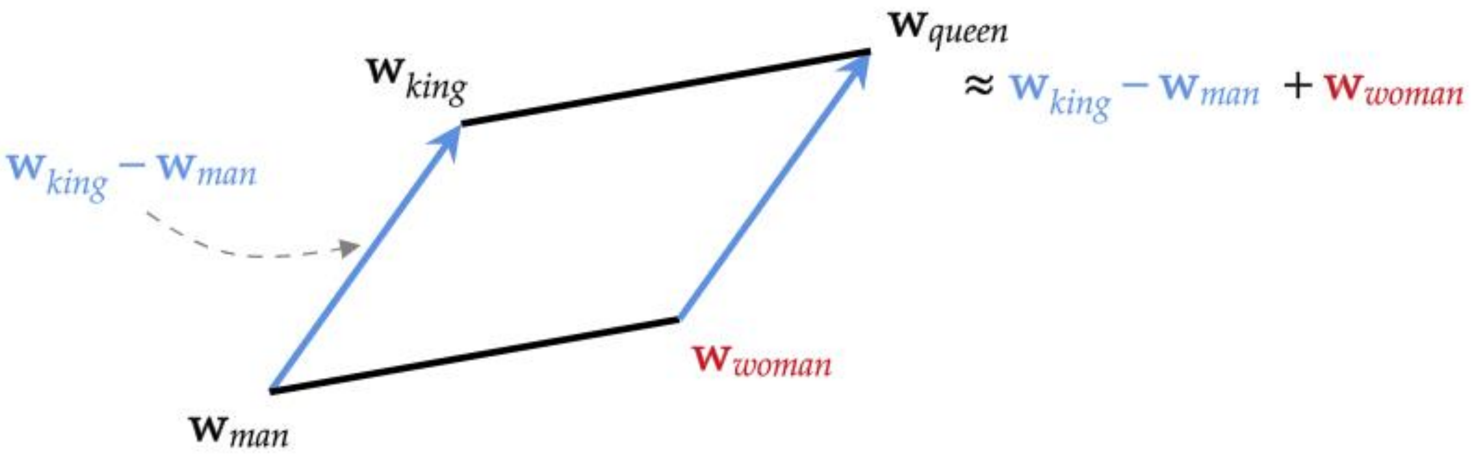
A classic example is the following one: «King - male + female ~ queen». That is, “man is to king as woman is to queen”. Why?

What for Word embedding models?

As the difference between vector *king* and vector *queen* is similar to the difference between vector *male* and vector *female*, then "King - queen ~ male - female", therefore, "King - male + female ~ queen"

Subtracting the **man** vector from the **king** vector and adding **female**, the most similar word to this would be **queen**

What for Word embedding models?



$\mathbf{w}_{queen}$
$\approx \mathbf{w}_{king} - \mathbf{w}_{man} + \mathbf{w}_{woman}$
$\mathbf{w}_{king}$
$\mathbf{w}_{king} - \mathbf{w}_{man}$
$\mathbf{w}_{woman}$
$\mathbf{w}_{man}$

word	D_1	D_2	D_3	...	D_N
man	0.46	0.67	0.05	...	...
woman	0.46	-0.89	-0.08	...	...
king	0.79	0.96	0.02	...	...
queen	0.80	-0.58	-0.14	...	...

What for Word embedding models?

It may be surprising that you are able to get these results, without any label on the meaning of words

"*Co-occurrence*" is however the key here! It is in fact the only information that you are using

By building a sense of one word’s proximity to other similar words, which do not necessarily contain the same letters, we can move beyond hard tokens to a smoother and more general sense of meaning

What for Word embedding models?

Word embedding models can be however thought not simply as a **method on their own** for analyzing text as data, but also as extremely **useful complements to representations of text as data based on word counts**

How?

The advantages of Embeddings

In a bag-or-words approach (i.e., in a context-blind approach) features are completely independent from one another (remember?). The feature word “dog” is as dissimilar to word “thinking” than it is to word “cat”

One the contrary, using a word embeddings approach, each feature is embedded into a *d* dimensional space, and represented as a vector in that space

And as we have already discussed, similar features will have similar vectors – i.e., information is shared between similar features (“dog” will be spatially closer to “cat” than to “thinking”)

The advantages of Embeddings

The main benefit of a dense representations lies therefore in **generalization power**: if we believe some features may provide similar clues, it is worthwhile to provide a representation that is able to capture these similarities

Think about the advantages of that!

New words in the test-set that were missing in training texts can now still be useful for classification! How? Through *words similarity* of course!

That is, using a set of word embeddings (either defined locally or using pre-trained embeddings) can allow new words encountered in the test-set to be classified according to their similarity to words that were present in the training-set allowing a ML model to share statistical strength between the two events

The advantages of Embeddings

For example, suppose that the word “cat” is included in both the training and the test-set. While the word “dog” is included just in the test-set

Further assume that according to your SVM, the word cat is very important to classify a text as “Positive” (as it arises in the training-set)

If “cat” and “dog” are (reasonably) spatially close to each other in the embedding world, *now* SVM can take advantage of the word “dog” (by exploiting its dimensional vector) during the prediction stage to classify a text as “Positive”, although the word “dog” wasn’t included in the training set!

The advantages of Embeddings

Through that, word embeddings may increase the accuracy of classification models as they also provide information (i.e., vector representations) on words **that are not represented** in the training data based on their similarity with other words

This is of course impossible in a bag-of-words approach in which new words encountered in the texts of the test-set are a *nuisance*

What for Word embedding models?

Word embeddings can also be very useful to expand *existing dictionaries*

Imagine you want to expand a dictionary of uncivil words

By looking for other words with semantic similarity to each of the terms included in the original dictionary, we can identify words that we may not have thought of in the first place, either because they’re slang, new words or just misspellings of existing words

What for Word embedding models?

An example:

Douglas R. Rice and Christopher Zorn. Corpus-based dictionaries for sentiment analysis of specialized vocabularies, *Political Science Research and Methods* (2021), 9, 20–35

Replication data: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/4EKHFM

What for Word embedding models?

Aim: to develop a sentiment dictionary that reflects the emotional valence of the words as they are used in a specific context

Method: selecting a small set of terms (“seeds”) uncontroversially positive or negative

As positive terms: “superb”, “brilliant”, “great”, “terrific”,“wonderful”, “splendid”, “good”, “fantastic”, “excellent”, and “enjoy”

As negative terms: “bad”, “awful”, “badly”,“dumb”, “horrible”, “wrong”, “terrible”, “poorly”, and “incorrect.”

The method: `GloVe`

What for Word embedding models?

After having estimated the vector representations for each token in their corpus, they have identified positively-valenced tokens by finding the token vector representations which were **most similar** to a vector constructed from the sum of the set of 10 uncontroversially positive tokens minus the sum of the set of 10 uncontroversially negative tokens

These new words - semantically similar to the uncontroversially positive and negative seed words within the domain - are then extracted in order to construct the dictionary

Advice to Practitioners

- ✓ Window-size and embedding dimensions: with the possible exception of small corpora, one should avoid using very few dimensions (below 100) and small window-sizes (< 5), unless one cares about syntactic relationships. In such instance, small windows may be desirable
- ✓ While performance improves with larger window-sizes and more dimensions, both exhibit decreasing returns: i.e., improvements are marginal beyond 300 dimensions and window-size of 6
- ✓ Given the tradeoff between more dimension/larger window-size and computation time, the choice of 5 or 6 (window-size) and between 100 and 300 (dimensions) seems therefore reasonable

R pakcages to install

install.packages("text2vec", repos='http://cran.us.r-project.org')

install.packages("lsa", repos='http://cran.us.r-project.org')

install.packages("readr", repos='http://cran.us.r-project.org')