

INTRODUCTION TO WORD EMBEDDINGS

Yogesh Kulkarni

December 3, 2020

Need

Why do we use word embeddings?

- ▶ Computers can not process words
- ▶ They need numbers.
- ▶ Especially for Machine learning where we can apply mathematical rules and do matrix operations to them

Why?

- ▶ Another big advantage is that is that we can actually pre-train word embeddings that are applicable to many tasks.
- ▶ Only caution is that these embeddings are generic text corpus and carry general meanings.
- ▶ If you want domain specific meanings, then you will need to build your own word embeddings.
- ▶ “Shell” in zoology could mean creature houses you find on sea shore, but the same word in CAD means Hollowing operation.
- ▶ “Draft” in legal domain is a rough document, where as in CAD it is Tapering operation.

What is a word embedding?

- ▶ Word Embeddings are the texts converted into numbers
- ▶ There may be different numerical representations of same text.
- ▶ Many Machine Learning algorithms and almost all Deep Learning Architectures are incapable of processing strings or plain text in their raw form.
- ▶ They require numbers as inputs to perform any sort of job, be it classification, regression etc. in broad terms.
- ▶ So, for the computer to be able to "understand" a vector representation of a word is required.

What is a word embedding?

- ▶ Word embedding = vector representation of a word
- ▶ A word is 'embedded' in n dimensional space.
- ▶ Once the words are vectors, their properties can be utilized.
- ▶ One such property is dot product giving cosine similarity.
- ▶ Goal is to capture some sort of relationship in that space, be it meaning, morphology, context, or some other kind of relationship.

What is a word embedding?

- ▶ The choice of n in word embedding space, is our choice.
- ▶ It can be tens or hundreds instead of millions (like one-hot encoded vectors).
- ▶ A lot of word embeddings are created based on the notion introduced by Zellig Harris' "distributional hypothesis" which boils down to a simple idea that words that are used close to one another typically have the same meaning.
- ▶ Different word embeddings are created either in different ways or using different text corpora to map this distributional relationship

Good Vector Representation

- ▶ To have "Semantic" (meaning-wise) representation, the Similar words should be close to each other in the hyper dimensional space.
- ▶ Non-similar words should be far apart from each other in the hyper dimensional space.

Different types of Word Vectors

- ▶ (Traditional) Frequency based Embedding:
 - ▶ One-hot
 - ▶ Count Vector
 - ▶ TF-IDF Vector
- ▶ (Modern) Prediction based Embedding:
 - ▶ Word2vec (Google)
 - ▶ Global Vector Representations (GloVe) (Stanford)
 - ▶ BERT
 - ▶ ...

One-hot Encoding

- ▶ Also called as Count Vectorizing
- ▶ A vector that has as many dimensions as your corpora has unique words.
- ▶ Each unique word has a unique dimension and will be represented by a 1 in that dimension with 0s everywhere else.

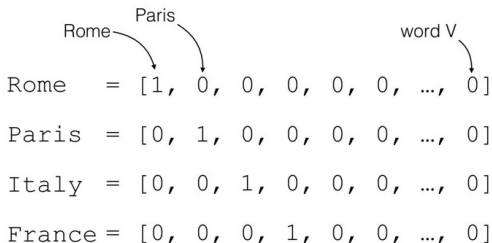


Diagram illustrating One-hot Encoding for words in a corpus. Each word is represented by a vector where only one dimension is 1 (the word's index) and all other dimensions are 0.

Labels above the vectors indicate the word corresponding to each 1 in the vector:

- Rome points to the first dimension of the first vector.
- Paris points to the second dimension of the first vector.
- word V points to the last dimension of the last vector.

The vectors are:

```

Rome    = [1, 0, 0, 0, 0, 0, ..., 0]
Paris   = [0, 1, 0, 0, 0, 0, ..., 0]
Italy   = [0, 0, 1, 0, 0, 0, ..., 0]
France  = [0, 0, 0, 1, 0, 0, ..., 0]
  
```

Result: A really huge and sparse vectors that capture absolutely no relational information.

One Hot

In traditional NLP, we regard words as discrete symbols: **hotel, conference, motel**

Means one 1, the rest 0s



Words can be represented by **one-hot** vectors:

motel = [0 0 0 0 0 0 0 0 0 1 0 0 0]

hotel = [0 0 0 0 0 0 1 0 0 0 0 0 0]

Vector dimension = number of words in vocabulary (e.g. 500,000)

(Ref: Word Embeddings - Elena Voita, Yandex Research)

One Hot: Problem

Example: in web search, if user searches for “**Seattle motel**”, we would like to match documents containing “**Seattle hotel**”.

But:

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

These two vectors are orthogonal.

There is no natural notion of **similarity** for one-hot vectors!

These vectors do not contain information about a **meaning** of a word.

(Ref: Word Embeddings - Elena Voita, Yandex Research)

Count Vector

- ▶ Corpus:
 - ▶ D1: He is a lazy boy. She is also lazy.
 - ▶ D2: Neeraj is a lazy person.
- ▶ Dictionary is a list of unique tokens(words)
=['He','She','lazy','boy','Neeraj','person']
- ▶ Count Matrix:

	He	She	lazy	boy	Neeraj	person
D1	1	1	2	1	0	0
D2	0	0	1	0	1	1

Count Vector

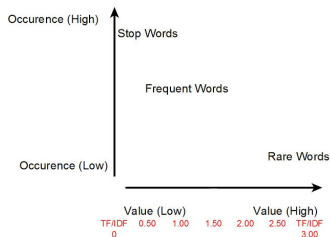
	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

← Word Vector (Passage Vector)

Document Vector

TF-IDF Transform

- ▶ Related to one-hot encoded vectors
- ▶ Words are represented by their term frequency multiplied by their inverse document frequency.
- ▶ Words that occur a lot but everywhere should be given very little weighting or significance.
- ▶ Words like “the” or “and” don’t provide a large amount of value.
- ▶ However, if a word appears very little or appears frequently, but only in one or two places, then its Imp and should be weighted High.



This suffers from the downside of very high dimensional representations that don’t capture semantic relatedness.

TF-IDF vectorization

- ▶ It takes into account not just the occurrence of a word in a single document but in the entire corpus.
- ▶ Down weigh the common words occurring in almost all documents and give more importance to words that appear in a subset of documents.

TF-IDF vectorization

- ▶ $TF = (\text{Number of times term } t \text{ appears in a document}) / (\text{Number of terms in the document})$
- ▶ So, $TF(\text{This}, \text{Document1}) = 1/8$ and $TF(\text{This}, \text{Document2}) = 1/5$
- ▶ $IDF = \log(N/n)$, where, N is the number of documents and n is the number of documents a term t has appeared in.
- ▶ So, $IDF(\text{This}) = \log(2/2) = 0$.
- ▶ $TFIDF = TF * IDF$
- ▶ Dictionary is made of a list of unique tokens(words)
- ▶ Similar to Count Matrix, TFIDF matrix is made with TFIDF values in it.

TF-IDF Inferencing

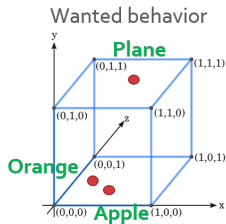
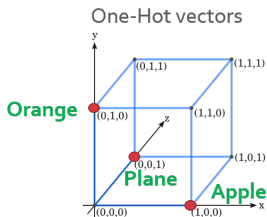
Once tf-idf model is trained on a corpus of documents, what happens when a test word or a document is given? How does it calculate tf-idf vector for it?

- ▶ For just one test word, its tf would be 1, then do we just multiply idf of that word from trained model? Are tf values in trained model irrelevant?
- ▶ Even for test document, does `inverse_transform` take into account tfs of current test document or tfs within trained model?
- ▶ Answer: the test word or document is 'ADDED' to the corpus and retraining/calculation is done to arrive at meaningful numbers.

(Ref: <https://stackoverflow.com/questions/55707577/how-does-tfidfvectorizer-compute-scores-on-test-data>)

Good Vector Representation

- ▶ Traditional One Hot Encoding:
 - ▶ Apple = [1, 0, 0]
 - ▶ Orange = [0, 1, 0]
 - ▶ Plane = [0, 0, 1]



- ▶ Very few cells participate in the representation.

Whats the solution? ... the 'Context'

Context

But, What is “meaning”?

What is “bardiwac”?

Anyone?

Lets try again, with examples

What is “bardiwac”?

- ▶ He handed her a glass of bardiwac.
- ▶ Beef dishes are made to complement the bardiwac.
- ▶ Nigel staggered to his feet, face flushed from too much bardiwac.
- ▶ Malbec, one of the lesser-known bardiwac grapes, responds well to Australia’s sunshine.
- ▶ I dined off bread and cheese and this excellent bardiwac.
- ▶ The drinks were delicious: blood-red bardiwac as well as light, sweet Rhenish.

Now, anyone?

At least one can guess now

What is “bardiwac”?

“Bardiwac is a red alcoholic beverage made from grapes ”

Context helps . . .

Distributed Semantics/Meaning

- ▶ A bottle of bardiwac is on the table.
- ▶ Everybody likes bardiwac.
- ▶ Don't have bardiwac before you drive.
- ▶ We make bardiwac out of corn.

Distributed Semantics/Meaning

- ▶ A bottle of xxxxxxxx is on the table.
- ▶ Everybody likes xxxxxxxx.
- ▶ Don't have xxxxxxxx before you drive.
- ▶ We make xxxxxxxx out of corn.

What other words fit into these places?

Won't they be similar to bardiwac?

Distributed Semantics/Meaning

- A bottle of _____ is on the table. (1)
- Everybody likes _____. (2)
- Don't have _____ before you drive. (3)
- We make _____ out of corn. (4)

What other words fit into these contexts?

	(1)	(2)	(3)	(4)	...
bardiwac	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
wine	1	1	1	0	
choices	0	1	0	0	

(Ref: Word Embeddings - Elena Voita, Yandex Research)

Distributed Semantics/Meaning

Closer ones are ...

Does vector similarity imply semantic similarity?



The **distributional hypothesis**, stated by Firth (1957):

"You shall know a word by the company it keeps."

(Ref: Word Embeddings - Elena Voita, Yandex Research)

Distributed Semantics/Meaning

Idea of Co-occurrence counts . . .

Corpus sentences

He also found five fish swimming in murky water in an old **bathub**.

We do abhor dust and dirt, and stains on the **bathub**, and any kind of filth.

Above At the far end of the garden room a **bathub** has been planted with herbs for the winter.

They had been drinking Cisco, a fruity, wine-based fluid that smells and tastes like a mixture of cough syrup and **bathub** gin.

Science finds that a surface tension on the water can draw the boats together, like toy boats in a **bathub**.

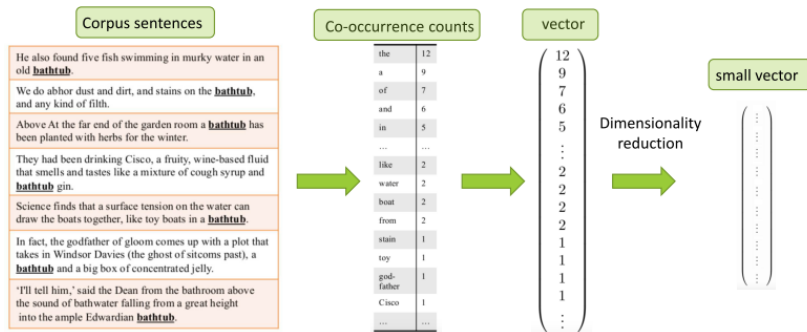
In fact, the godfather of gloom comes up with a plot that takes in Windsor Davies (the ghost of sitcoms past), a **bathub** and a big box of concentrated jelly.

'I'll tell him,' said the Dean from the bathroom above the sound of bathwater falling from a great height into the ample Edwardian **bathub**.

(Ref: Word Embeddings - Elena Voita, Yandex Research)

Distributed Semantics/Meaning

Calculate Co-occurrences for the context word, that itself becomes its own representation!!!



(Ref: Word Embeddings - Elena Voita, Yandex Research)

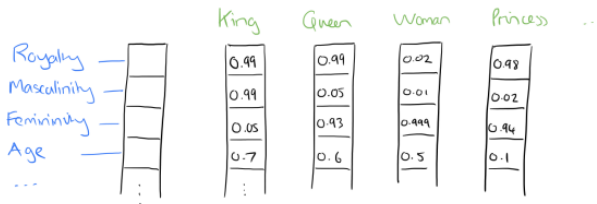
Modern Word Vectors

Modern Word Vectors

- ▶ How do we actually build these super-intelligent vectors, that seem to have such magical powers?
- ▶ How to find a word's friends?
- ▶ We will discuss the most famous methods to build such lower-dimension vector representations for words based on their context
 - ▶ Word2vec (Google)
 - ▶ Global Vector Representations (GloVe) (Stanford)
 - ▶ ...

Word2Vec

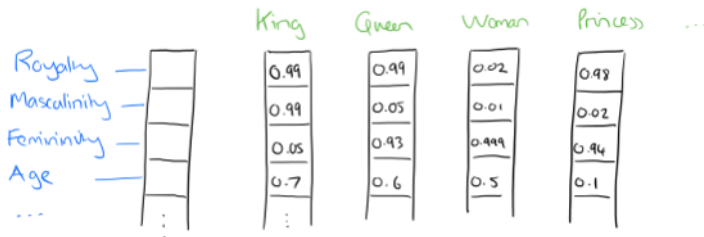
Word2vec (Google): a distributed representation of a word is used and not sparse like One-Hot.



Represent in some abstract way the 'meaning' of a word.

Word Distributed Representation - Word2Vec

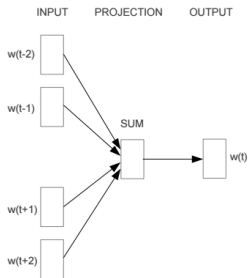
- ▶ All vector cells participate in representing each word.
- ▶ Words are represented by real valued dense vectors of significantly smaller dimensions (e.g. 100 - 1000).
- ▶ Intuition: consider each vector cell as a representative of some feature.



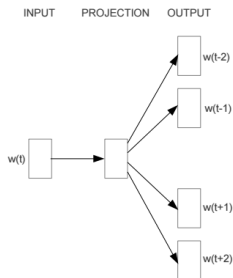
Word2Vec

Two major learning approaches:

- ▶ Continuous Bag-of-Words (CBOW): Learns an embedding by predicting the current words based on the context. The context is determined by the surrounding words.
- ▶ Continuous Skip-Gram: Learns an embedding by predicting the surrounding words given the context. The context is the current word.



CBOW



Skip-gram

Word2Vec

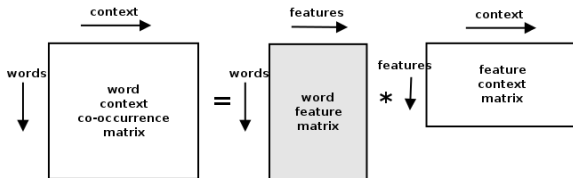
- ▶ Both of these learning methods use local word usage context (with a defined window of neighboring words).
- ▶ The larger the window is, the more topical similarities that are learned by the embedding. Forcing a smaller window results in more semantic, syntactic, and functional similarities to be learned.

Benefits

- ▶ Low space and low time complexity to generate a rich representation
- ▶ The larger the dimensionality, the more features we can have in our representation
- ▶ Allows us to efficiently generate something like a billion word corpora,
- ▶ But encompass a bunch of generalities and keep the dimensionality small.

Glove

- ▶ An extension of word2vec, and a much better one at that.
- ▶ Contribution was the addition of global statistics in the language modeling task to generate the embedding.
- ▶ There is no window feature for local context.
- ▶ Instead, there is a word-context/word co-occurrence matrix that learns statistics across the entire corpora.



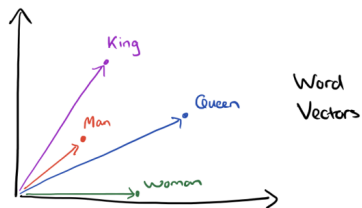
FastText

- ▶ To incorporate sub-word information by splitting all words into a bag of n-gram characters (typically of size 3-6).
- ▶ It would add these sub-words together to create a whole word as a final feature.
- ▶ Power: support out-of-vocabulary words!
- ▶ In other approaches, if the system encounters a word that it doesn't recognize, it just has to set it to the unknown word. With FastText, we can give meaning to words like circumnavigate if we only know the word navigate, because our semantic knowledge of the word navigate can help use at least provide a bit more semantic information to circumnavigate, even if it is not a word our system learned during training.

Examples

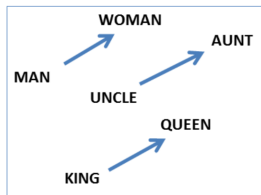
Examples

Vectors for King, Man, Queen, & Woman:

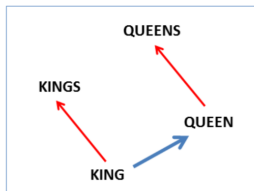


Examples

Gender relation:



Plural relation:



Examples

Word pair relationships:

Table 8: Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Country-capital city relationship:

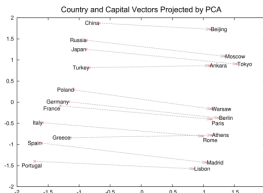


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Word Representations Comparison

Traditional Method - Bag of Words Model

- ▶ Uses one hot encoding
- ▶ Each word in the vocabulary is represented by one bit position in a HUGE vector.
- ▶ For example, with a vocabulary of 10000 words, and "Hello" is the 4th word in the dictionary: 0 0 0 1 0 0 .
 0 0 0 0
- ▶ Context information is not utilized

Modern - Word Vectors

- ▶ Stores each word in as a point in space, represented by a vector of fixed number of dimensions (generally 300)
- ▶ Unsupervised, built just by reading huge corpus
- ▶ For example, "Hello" might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02]
- ▶ Context information is utilized

Applications

Word Similarity

- ▶ Classic Methods : Edit Distance, WordNet, Porter's Stemmer, Lemmatization using dictionaries
- ▶ Easily identifies similar words and synonyms since they occur in similar contexts
- ▶ Stemming (thought → think)
- ▶ Inflections, Tense forms
- ▶ eg. Think, thought, ponder, pondering,
- ▶ eg. Plane, Aircraft, Flight

-

Part-of-Speech and Named Entity Recognition

- Classic Methods : Sequential Models (MEMM , Conditional Random Fields), Logistic Regression

	POS WSJ (acc.)	NER CoNLL (F1)
State-of-the-art*	97.24	89.31
Supervised NN	96.37	81.47
Unsupervised pre-training followed by supervised NN**	97.20	88.87
+ hand-crafted features***	97.29	89.59

Sentiment Analysis

- ▶ Classic Methods : Naive Bayes, Random Forests/SVM
- ▶ Classifying sentences as positive and negative
- ▶ Building sentiment lexicons using seed sentiment sets
- ▶ No need for classifiers, we can just use cosine distances to compare unseen reviews to known reviews.

```
Enter word or sentence (EXIT to break): sad
Word: sad Position in vocabulary: 4067
```

Word	Cosine distance
saddening	0.727309
Sad	0.661083
saddened	0.660439
heartbreaking	0.657351
disheartening	0.650732
Meny_Friedman	0.648706
parishioner_Pat_Patello	0.647586
saddens_me	0.640712
distressing	0.639909
reminders_bobbing	0.635772
Turkoman_Shites	0.635577
saddest	0.634551
unfortunate	0.627209
sorry	0.619405
bittersweet	0.617521
tragic	0.611279
regretful	0.603472

Other Applications

- ▶ Co-reference Resolution: Chaining entity mentions across multiple documents - can we find and unify the multiple contexts in which mentions occurs?
- ▶ Clustering: Words in the same class naturally occur in similar contexts, and this feature vector can directly be used with any conventional clustering algorithms (K-Means, agglomerative, etc). Human doesn't have to waste time hand-picking useful word features to cluster on.
- ▶ Topic modeling: Semantic Analysis of Documents: Build word distributions for various topics, etc.

Evaluation

Efficacy

Intrinsic: evaluation on a specific/intermediate subtask

- ▶ word analogies: “a is to b as c is to xxxx?”
- ▶ word similarity: correlation of the rankings

Extrinsic: evaluation on a real task

- ▶ take some task (MT, NER, coreference resolution, ...) or several tasks
- ▶ train with different pretrained word embeddings
- ▶ if the task quality is better -j win!

Advantages

- ▶ They provide a fresh perspective to ALL problems in NLP, and not just solve one problem.
- ▶ Technological Improvement
- ▶ Rise of deep learning since 2006 (Big Data + GPUs + Work done by Andrew Ng, Yoshua Bengio, Yann Lecun and Geoff Hinton)
- ▶ Application of Deep Learning to NLP - led by Yoshua Bengio, Christopher Manning, Richard Socher, Tomas Mikalov
- ▶ The need for unsupervised learning . (Supervised learning tends to be excessively dependent on hand-labeled data and often does not scale)

Disadvantages

- ▶ Single vector for a word (whats problem with that?)
- ▶ Very vulnerable, and not a robust concept
- ▶ Can take a long time to train
- ▶ Non-uniform results
- ▶ Hard to understand and visualize

Summary

Machines like humans need four things to understand language

- ▶ Understand the words (semantics)
- ▶ Build the ability to guess (language model)
- ▶ Parse language specific rules and patterns (encoder-decoder, transformers)
- ▶ Build on the experience (pre-training)

Ability to include non-language information (culture, visuals, etc) will improve language models.

Thanks ... yogeshkulkarni@yahoo.com