Word 2 Vec

Atul Dhingra

ODH

November 13, 2017

Motivation

- How similar are these sentences ?

Motivation

- How similar are these sentences ?
  - s1: Monday, Monday!

Motivation

- How similar are these sentences ?
  - s1: Monday, Monday!
  - s2: Today is a Monday

Motivation

- How similar are these sentences ?
    - s1: Monday, Monday!
    - s2: Today is a Monday
    - s3: Today is a Tuesday

Motivation

- How similar are these sentences ?
  - s1: Monday, Monday!
  - s2: Today is a Monday
  - s3: Today is a Tuesday
  - s4: Is today a Monday

Motivation

- How similar are these sentences ?
  - s1: Monday, Monday!
  - s2: Today is a Monday
  - s3: Today is a Tuesday
  - s4: Is today a Monday
- First order of business: Find a good representation of the text

Proposed Solution

- One-hot encoded vector for the entire vocabulary

Proposed Solution

- One-hot encoded vector for the entire vocabulary

  `vocab` $\sim$ `[Monday, Tuesday, is, a, today]`

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ~              [1, 0, 0, 0, 0]
```

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ~                [1, 0, 0, 0, 0]
s2:  Today is a Monday ~               [1, 0, 1, 1, 1]
```

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ∼                [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ∼              [1, 0, 0, 0, 0]
s2:  Today is a Monday ∼             [1, 0, 1, 1, 1]
s3:  Today is a Tuesday ∼            [0, 1, 1, 1, 1]
```

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ∼                [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ∼               [1, 0, 0, 0, 0]
s2:  Today is a Monday ∼              [1, 0, 1, 1, 1]
s3:  Today is a Tuesday ∼             [0, 1, 1, 1, 1]
```

- Similarity between sentences: Cosine distance

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ~               [1, 0, 0, 0, 0]
s2:  Today is a Monday ~              [1, 0, 1, 1, 1]
s3:  Today is a Tuesday ~             [0, 1, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $dist(s1, s2) = dist([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                [Monday, Tuesday, is, a, today]
s1: Monday, Monday! ~               [1, 0, 0, 0, 0]
s2: Today is a Monday ~             [1, 0, 1, 1, 1]
s3: Today is a Tuesday ~            [0, 1, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $dist(s1, s2) = dist([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$
  - $dist(s1, s3) = dist([1, 0, 0, 0, 0], [0, 1, 1, 1, 1]) = 1$

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ~              [1, 0, 0, 0, 0]
s2:  Today is a Monday ~             [1, 0, 1, 1, 1]
s3:  Today is a Tuesday ~            [0, 1, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $dist(s1, s2) = dist([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$
  - $dist(s1, s3) = dist([1, 0, 0, 0, 0], [0, 1, 1, 1, 1]) = 1$
  - $dist(s2, s3) = dist([1, 0, 1, 1, 1], [0, 1, 1, 1, 1]) = \mathbf{0.25}$

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                  [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ~              [1, 0, 0, 0, 0]
s2:  Today is a Monday ~             [1, 0, 1, 1, 1]
s3:  Today is a Tuesday ~            [0, 1, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $dist(s1, s2) = dist([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$
  - $dist(s1, s3) = dist([1, 0, 0, 0, 0], [0, 1, 1, 1, 1]) = 1$
  - $dist(s2, s3) = dist([1, 0, 1, 1, 1], [0, 1, 1, 1, 1]) = \mathbf{0.25}$

- Leads to data sparsity, therefore need more data

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~               [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ~             [1, 0, 0, 0, 0]
s2:  Today is a Monday ~            [1, 0, 1, 1, 1]
s3:  Today is a Tuesday ~           [0, 1, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $dist(s1, s2) = dist([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$
  - $dist(s1, s3) = dist([1, 0, 0, 0, 0], [0, 1, 1, 1, 1]) = 1$
  - $dist(s2, s3) = dist([1, 0, 1, 1, 1], [0, 1, 1, 1, 1]) = \mathbf{0.25}$

- Leads to data sparsity, therefore need more data
- Does not capture the context

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ~               [1, 0, 0, 0, 0]
s2:  Today is a Monday ~              [1, 0, 1, 1, 1]
s3:  Today is a Tuesday ~             [0, 1, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $dist(s1, s2) = dist([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$
  - $dist(s1, s3) = dist([1, 0, 0, 0, 0], [0, 1, 1, 1, 1]) = 1$
  - $dist(s2, s3) = dist([1, 0, 1, 1, 1], [0, 1, 1, 1, 1]) = \mathbf{0.25}$

- Leads to data sparsity, therefore need more data
- Does not capture the context
  - Barack Obama said that George bush is a bad man
  - George Bush said that Barack Obama is a bad man

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ~              [1, 0, 0, 0, 0]
s2:  Today is a Monday ~             [1, 0, 1, 1, 1]
s3:  Today is a Tuesday ~            [0, 1, 1, 1, 1]
s4:  Is today a Monday ~             [1, 0, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $dist(s1, s2) = dist([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$
  - $dist(s1, s3) = dist([1, 0, 0, 0, 0], [0, 1, 1, 1, 1]) = 1$
  - $dist(s2, s3) = dist([1, 0, 1, 1, 1], [0, 1, 1, 1, 1]) = \mathbf{0.25}$

- Leads to data sparsity, therefore need more data
- Does not capture the context

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                  [Monday, Tuesday, is, a, today]
s1: Monday, Monday! ~               [1, 0, 0, 0, 0]
s2: Today is a Monday ~             [1, 0, 1, 1, 1]
s3: Today is a Tuesday ~            [0, 1, 1, 1, 1]
s4: Is today a Monday ~             [1, 0, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $\mathit{dist}(s1, s2) = \mathit{dist}([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$
  - $\mathit{dist}(s1, s3) = \mathit{dist}([1, 0, 0, 0, 0], [0, 1, 1, 1, 1]) = 1$
  - $\mathit{dist}(s2, s3) = \mathit{dist}([1, 0, 1, 1, 1], [0, 1, 1, 1, 1]) = \mathbf{0.25}$
  - $\mathit{dist}(s2, s4) = \mathit{dist}([1, 0, 1, 1, 1], [1, 0, 1, 1, 1]) = \mathbf{0}$
- Leads to data sparsity, therefore need more data
- Does not capture the context

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ~              [1, 0, 0, 0, 0]
s2:  Today is a Monday ~             [1, 0, 1, 1, 1]
s3:  Today is a Tuesday ~            [0, 1, 1, 1, 1]
s4:  Is today a Monday ~             [1, 0, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $dist(s1, s2) = dist([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$
  - $dist(s1, s3) = dist([1, 0, 0, 0, 0], [0, 1, 1, 1, 1]) = 1$
  - $dist(s2, s3) = dist([1, 0, 1, 1, 1], [0, 1, 1, 1, 1]) = \mathbf{0.25}$
  - $dist(s2, s4) = dist([1, 0, 1, 1, 1], [1, 0, 1, 1, 1]) = \mathbf{0}$
- Leads to data sparsity, therefore need more data
- Does not capture the context
- More frequent words should have more effect

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                  [Monday, Tuesday, is, a, today]
s1:  Monday, Monday!  ~              [1, 0, 0, 0, 0]
s2:  Today is a Monday ~             [1, 0, 1, 1, 1]
s3:  Today is a Tuesday ~            [0, 1, 1, 1, 1]
s4:  Is today a Monday ~             [1, 0, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $dist(s1, s2) = dist([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$
  - $dist(s1, s3) = dist([1, 0, 0, 0, 0], [0, 1, 1, 1, 1]) = 1$
  - $dist(s2, s3) = dist([1, 0, 1, 1, 1], [0, 1, 1, 1, 1]) = \mathbf{0.25}$
  - $dist(s2, s4) = dist([1, 0, 1, 1, 1], [1, 0, 1, 1, 1]) = \mathbf{0}$
- Leads to data sparsity, therefore need more data
- Does not capture the context
- More frequent words should have more effect
  - Use Bag of Words approach, where counts are used instead

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                 [Monday, Tuesday, is, a, today]
s1: Monday, Monday!  ~              [1, 0, 0, 0, 0]
s2: Today is a Monday ~             [1, 0, 1, 1, 1]
s3: Today is a Tuesday ~            [0, 1, 1, 1, 1]
s4: Is today a Monday ~             [1, 0, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $dist(s1, s2) = dist([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$
  - $dist(s1, s3) = dist([1, 0, 0, 0, 0], [0, 1, 1, 1, 1]) = 1$
  - $dist(s2, s3) = dist([1, 0, 1, 1, 1], [0, 1, 1, 1, 1]) = \mathbf{0.25}$
  - $dist(s2, s4) = dist([1, 0, 1, 1, 1], [1, 0, 1, 1, 1]) = \mathbf{0}$
- Leads to data sparsity, therefore need more data
- Does not capture the context
- More frequent words should have more effect
  - Use Bag of Words approach, where counts are used instead
  - e.g. `s1: Monday, Monday!  ~        [2, 0, 0, 0, 0]`
- Feature space grows with vocabulary size

Proposed Solution

- One-hot encoded vector for the entire vocabulary

```
vocab ~                [Monday, Tuesday, is, a, today]
s1: Monday, Monday!  ~             [1, 0, 0, 0, 0]
s2: Today is a Monday ~            [1, 0, 1, 1, 1]
s3: Today is a Tuesday ~           [0, 1, 1, 1, 1]
s4: Is today a Monday ~            [1, 0, 1, 1, 1]
```

- Similarity between sentences: Cosine distance
  - $dist(s1, s2) = dist([1, 0, 0, 0, 0], [1, 0, 1, 1, 1]) = 0.5$
  - $dist(s1, s3) = dist([1, 0, 0, 0, 0], [0, 1, 1, 1, 1]) = 1$
  - $dist(s2, s3) = dist([1, 0, 1, 1, 1], [0, 1, 1, 1, 1]) = \mathbf{0.25}$
  - $dist(s2, s4) = dist([1, 0, 1, 1, 1], [1, 0, 1, 1, 1]) = \mathbf{0}$
- Leads to data sparsity, therefore need more data
- Does not capture the context
- More frequent words should have more effect
  - Use Bag of Words approach, where counts are used instead
  - e.g. `s1: Monday, Monday!  ~        [2, 0, 0, 0, 0]`
- Feature space grows with vocabulary size
- Need to learn a low-dimensional representation: Word2Vec

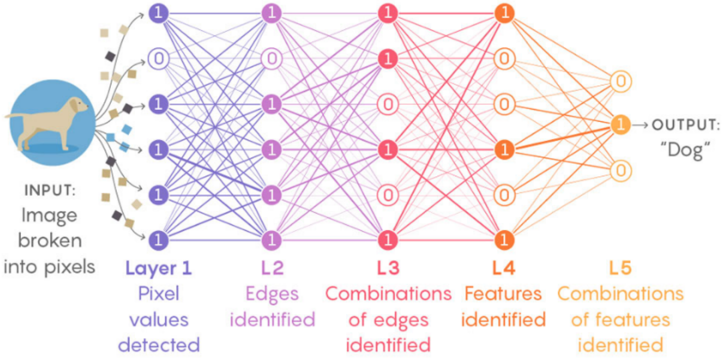
Neural Network

- Neural network as a black box

Neural Network as a black box

Neural Network

- Internal structure of a neural network is able to represent information



Internal representation of a Neural Network

Source: https://www.quantamagazine.org/new-theory-cracks-open-the-black-box-of-deep-learning-20170921/

Word Vectors

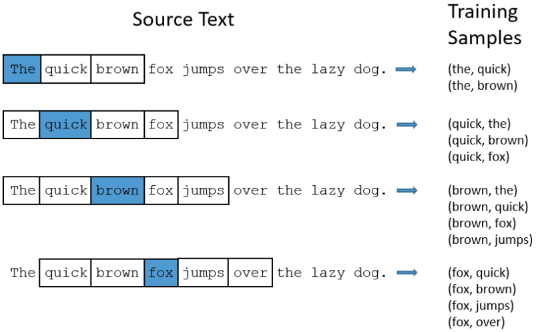
- Intuition: Train a neural network for an “auxiliary” task, and use the learned weights as a representation(word-vectors)

Word Vectors

- Intuition: Train a neural network for an “auxiliary” task, and use the learned weights as a representation(word-vectors)
- Auxiliary Task: Given a specific input word, compute probability for every word in our vocabulary of being the neighbor

Word Vectors

- Intuition: Train a neural network for an “auxiliary” task, and use the learned weights as a representation(word-vectors)
- Break the sentence into small windows(size=2), and create training set for each input word(in blue)

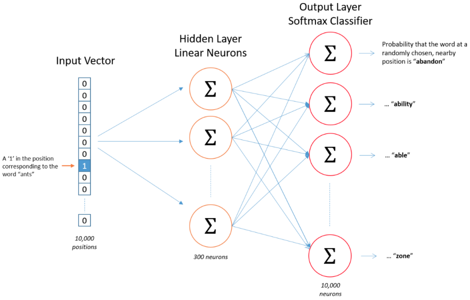


Generating Training Data

Source: *McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model*

Word Vectors

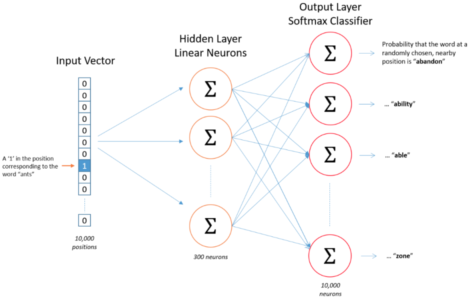
- Intuition: Train a neural network for an “auxiliary” task, and use the learned weights as a representation(word-vectors)
- Feed the training data as one-hot encoded vectors to the model, such that the output is a the probability of a word being the neighbor of target.



Neural Network for training the auxiliary task

Word Vectors

- Intuition: Train a neural network for an "auxiliary" task, and use the learned weights as a representation(word-vectors)
- Use the hidden layer as the word-representation



Word vectors as the hidden layer

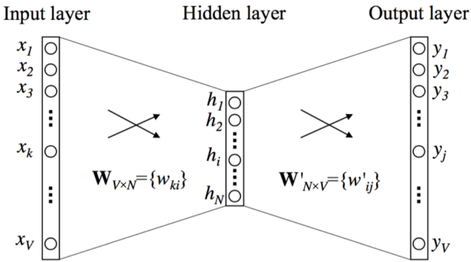
Source: McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model

How does Word2Vec work?

- Given an input word, predict single target word

How does Word2Vec work?

- Given an input word, predict single target word



Single context word model

Source: David Meyer, How exactly does word2vec work?

How does Word2Vec work?

- Given an input word, predict single target word
- $X = (x_1, x_2, ...x_V)$, $Y = (y_1, y_2...y_V)$, where V is the size of vocabulary, $x_i \in X$ and $y_i \in Y$ are one-hot encoded vectors

How does Word2Vec work?

- Given an input word, predict single target word
- $X = (x_1, x_2, ...x_V), Y = (y_1, y_2...y_V)$, where V is the size of vocabulary, $x_i \in X$ and $y_i \in Y$ are one-hot encoded vectors
- Hidden layer neurons can be computed by, $h = X^T.W$

How does Word2Vec work?

- Given an input word, predict single target word
- $X = (x_1, x_2, ...x_V), Y = (y_1, y_2...y_V)$, where V is the size of vocabulary, $x_i \in X$ and $y_i \in Y$ are one-hot encoded vectors
- Hidden layer neurons can be computed by, $h = X^T.W$
- For one-hot encoded $x_k = 1$, the above operation copies k^{th} row of W to h, i.e $h = X^T.W = W_{(k,:)} = v_{in}$, where v_{in} is the vector representation of input word

How does Word2Vec work?

- Given an input word, predict single target word
- $X = (x_1, x_2, ...x_V)$, $Y = (y_1, y_2...y_V)$, where V is the size of vocabulary, $x_i \in X$ and $y_i \in Y$ are one-hot encoded vectors
- Hidden layer neurons can be computed by, $h = X^T.W$
- For one-hot encoded $x_k = 1$, the above operation copies k^{th} row of W to h, i.e $h = X^T.W = W_{(k,:)} = v_{in}$, where v_{in} is the vector representation of input word
- Compute the output score for each word in vocabulary V, based on h, $u_j = {v'_{w_j}}^T.h$, where v'_{w_j} is j^{th} column in W'

How does Word2Vec work?

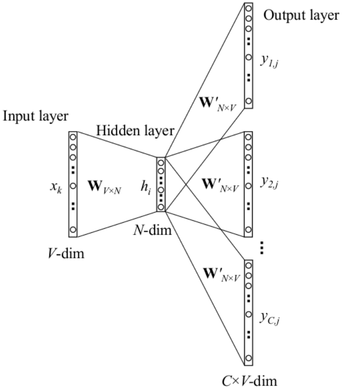
- Given an input word, predict single target word
- $X = (x_1, x_2, ...x_V), Y = (y_1, y_2...y_V)$, where V is the size of vocabulary, $x_i \in X$ and $y_i \in Y$ are one-hot encoded vectors
- Hidden layer neurons can be computed by, $h = X^T.W$
- For one-hot encoded $x_k = 1$, the above operation copies k^{th} row of W to h, i.e $h = X^T.W = W_{(k,:)} = v_{in}$, where v_{in} is the vector representation of input word
- Compute the output score for each word in vocabulary V, based on h, $u_j = {v'_{w_j}}^T.h$, where v'_{w_j} is j^{th} column in W'
- Compute the posterior probability using softmax, $p(w_j|w_{in}) = y_j = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u'_j)}$, where y_j is the output of the j^{th} unit in output layer

How does Word2Vec work?

- Given an input word, predict single target word
- $X = (x_1, x_2, ...x_V)$, $Y = (y_1, y_2...y_V)$, where V is the size of vocabulary, $x_i \in X$ and $y_i \in Y$ are one-hot encoded vectors
- Hidden layer neurons can be computed by, $h = X^T.W$
- For one-hot encoded $x_k = 1$, the above operation copies k^{th} row of W to h, i.e $h = X^T.W = W_{(k,:)} = v_{in}$, where v_{in} is the vector representation of input word
- Compute the output score for each word in vocabulary V, based on h, $u_j = {v'_{w_j}}^T.h$, where v'_{w_j} is j^{th} column in W'
- Compute the posterior probability using softmax, $p(w_j|w_{in}) = y_j = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u'_j)}$, where y_j is the output of the j^{th} unit in output layer
- The training objective, therefore, is to maximize the conditional probability of observing the actual output word, given the input context word

Word2Vec

- Skip-gram (SG): use a word to predict the surrounding ones in window.

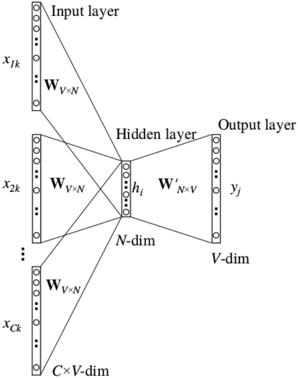


Structure of a Skip-gram model

Source: Xin Rong, Word2Vec Parameter Learning Explained

Word2Vec

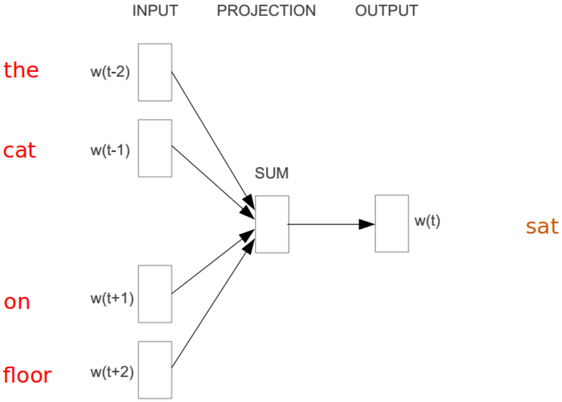
- Skip-gram (SG): use a word to predict the surrounding ones in window.
- Continuous Bag of Words (CBOW): use a window of word to predict the middle word



Structure of a Continuous Bag-Of-Words Model

CBOW

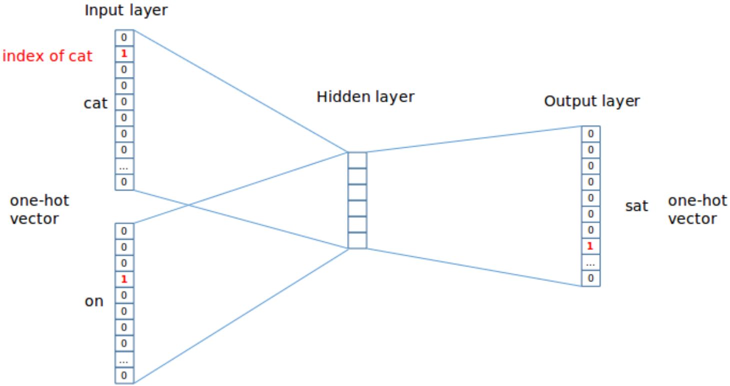
- e.g. "The cat sat on floor" (window = 2)



Sentence Structure

Source: Vagelis Hristidis, Vector Representation of Text

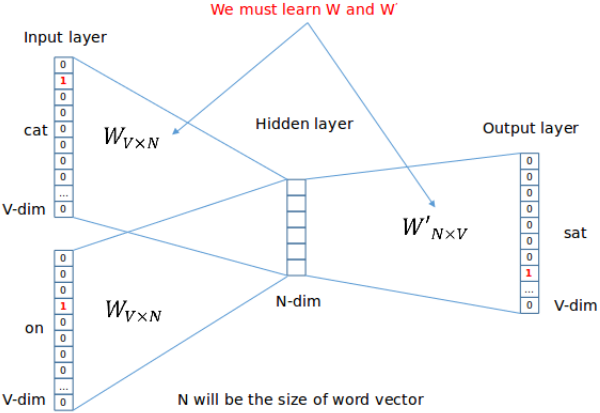
CBOW



One hot encoded input and output

Source: Vagelis Hristidis, Vector Representation of Text

CBOW



Learning W, W' matrices

Source: Vagelis Hristidis, Vector Representation of Text

Thrones2Vec

- Load all text from “Song of Ice and Fire” GoT books

Thrones2Vec

- Load all text from “Song of Ice and Fire” GoT books
  - “A Clash of Kings”, “A Storm of Swords”, “ A Song of Ice and Fire ”, “ A Feast for Crows”, “ A Game of Thrones”
- Convert the book into sentences by using tokenizer used in English such as period, question mark etc

Thrones2Vec

- Load all text from "Song of Ice and Fire" GoT books
  - "A Clash of Kings", "A Storm of Swords", " A Song of Ice and Fire ", " A Feast for Crows", " A Game of Thrones"
- Convert the book into sentences by using tokenizer used in English such as period, question mark etc
- Clean each sentence to remove unnecessary words, punctuations, hyphens etc and split into words

Thrones2Vec

- Load all text from “Song of Ice and Fire” GoT books
  - “A Clash of Kings”, “A Storm of Swords”, “ A Song of Ice and Fire ”, “ A Feast for Crows”, “ A Game of Thrones”
- Convert the book into sentences by using tokenizer used in English such as period, question mark etc
- Clean each sentence to remove unnecessary words, punctuations, hyphens etc and split into words
  - `Heraldic crest by Virginia Norey.`

Thrones2Vec

- Load all text from “Song of Ice and Fire” GoT books
  - “A Clash of Kings”, “A Storm of Swords”, “ A Song of Ice and Fire ”, “ A Feast for Crows”, “ A Game of Thrones”
- Convert the book into sentences by using tokenizer used in English such as period, question mark etc
- Clean each sentence to remove unnecessary words, punctuations, hyphens etc and split into words
  - `Heraldic crest by Virginia Norey.`
  - `[‘Heraldic’, ‘crest’, ‘by’, ‘Virginia’, ‘Norey’]`

Thrones2Vec

- Load all text from “Song of Ice and Fire” GoT books
  - “A Clash of Kings”, “A Storm of Swords”, “ A Song of Ice and Fire ”, “ A Feast for Crows”, “ A Game of Thrones”
- Convert the book into sentences by using tokenizer used in English such as period, question mark etc
- Clean each sentence to remove unnecessary words, punctuations, hyphens etc and split into words
  - `Heraldic crest by Virginia Norey.`
  - `[‘Heraldic’, ‘crest’, ‘by’, ‘Virginia’, ‘Norey’]`
- Build the vocabulary using the window size of 7 units, and minimum word count of 3 units

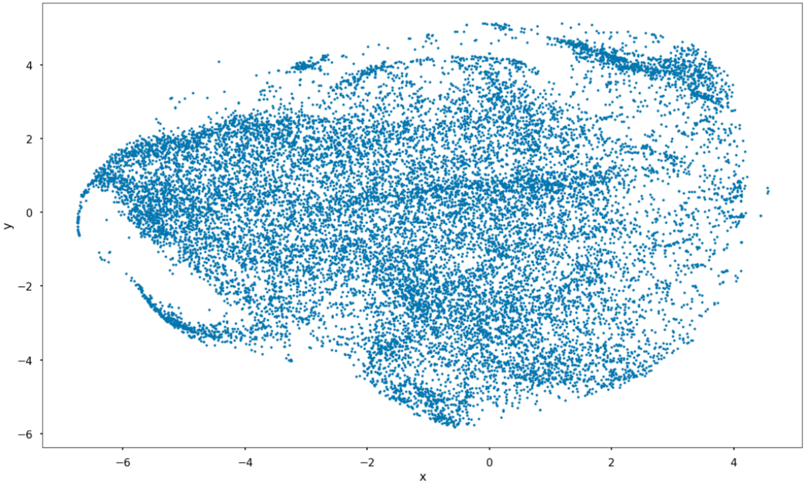
Thrones2Vec

- Load all text from "Song of Ice and Fire" GoT books
  - "A Clash of Kings", "A Storm of Swords", " A Song of Ice and Fire ", " A Feast for Crows", " A Game of Thrones"
- Convert the book into sentences by using tokenizer used in English such as period, question mark etc
- Clean each sentence to remove unnecessary words, punctuations, hyphens etc and split into words
  - `Heraldic crest by Virginia Norey.`
  - `['Heraldic', 'crest', 'by', 'Virginia', 'Norey']`
- Build the vocabulary using the window size of 7 units, and minimum word count of 3 units
- Train a skip-gram model using gensim on the entire vocabulary to obtain a 300-dimensional feature(word)-vector

Thrones2Vec

- Load all text from “Song of Ice and Fire” GoT books
  - “A Clash of Kings”, “A Storm of Swords”, “ A Song of Ice and Fire ”, “ A Feast for Crows”, “ A Game of Thrones”
- Convert the book into sentences by using tokenizer used in English such as period, question mark etc
- Clean each sentence to remove unnecessary words, punctuations, hyphens etc and split into words
  - `Heraldic crest by Virginia Norey.`
  - `['Heraldic', 'crest', 'by', 'Virginia', 'Norey']`
- Build the vocabulary using the window size of 7 units, and minimum word count of 3 units
- Train a skip-gram model using gensim on the entire vocabulary to obtain a 300-dimensional feature(word)-vector
- Compress the word-vectors into a 2D space for visualization

The Big Picture



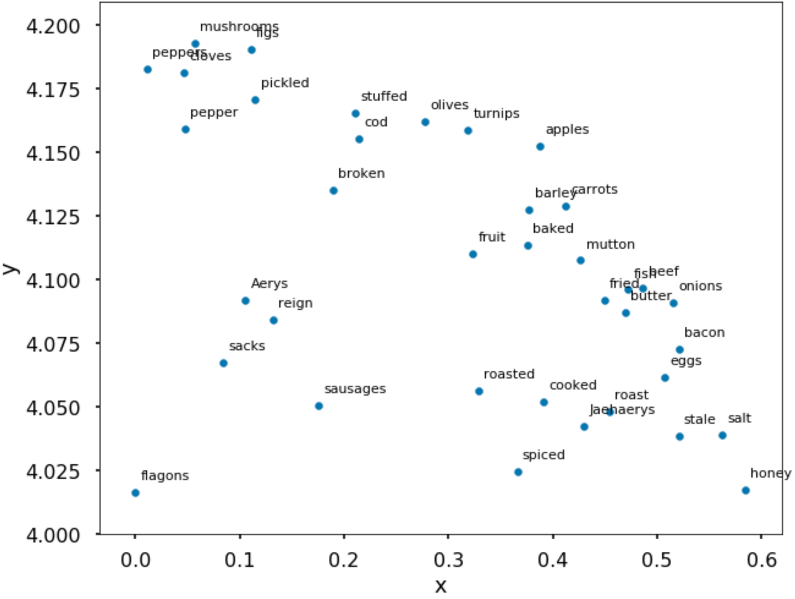
Embedding of the entire vocabulary space onto 2-D

Word Mappings

	word	x	y
0	fawn	-4.470860	-0.406855
1	raining	2.432409	-1.825349
2	writings	-3.212095	1.967637
3	Ysilla	1.436866	-2.421560
4	Rory	-1.090941	-2.569549
5	hordes	-2.204853	2.614524
6	mustachio	-1.086925	-3.887781
7	Greyjoy	1.585396	3.667034
8	yellow	-0.813293	-5.425221
9	four	1.871287	2.557694

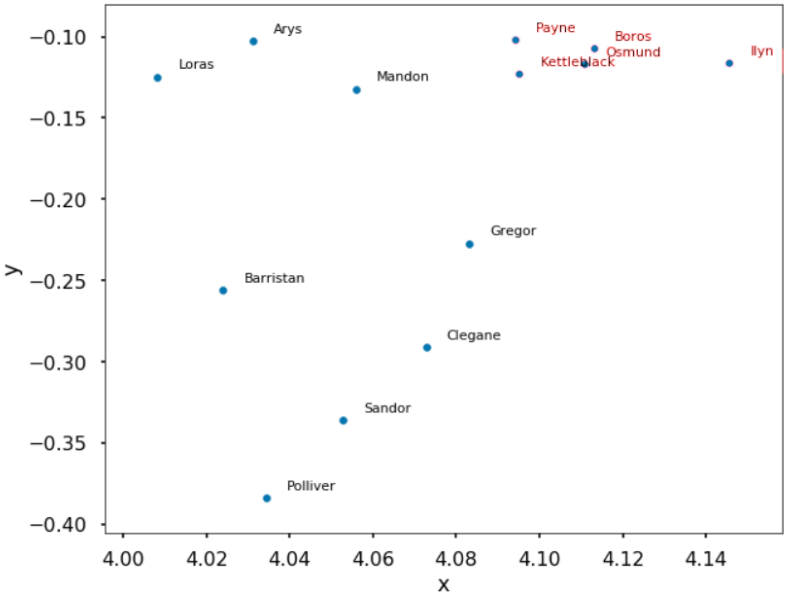
Mapping of words on x,y axis from t-SNE

Similar objects cluster together



Food Items group together

Similar objects cluster together



People related to Kingsguard ended up together

Most Similar To

- `thrones2vec.most_similar("Stark")`

Most Similar To

- `thrones2vec.most_similar("Stark")`

  ('Eddard', 0.7424380779266357),
  ('Winterfell', 0.6484879851341248),
  ('Brandon', 0.643855094909668),
  ('Lyanna', 0.6438395977020264),
  ('Robb', 0.6242259740829468),
  ('executed', 0.6220564842224121),
  ('Arryn', 0.6189972162246704),
  ('Benjen', 0.6188897490501404),
  ('direwolf', 0.6143664121627808),
  ('beheaded', 0.6046537756919861)

Most Similar to

- `thrones2vec.most_similar("Dragons")`

Most Similar to

- `thrones2vec.most_similar("Dragons")`

  (‘Unburnt’, 0.8507828712463379),
  (‘Stormborn’, 0.815880537033081),
  (‘Khaleesi’, 0.7907167673110962),
  (‘Mother’, 0.7906662225723267),
  (‘khaleesi’, 0.7895367741584778),
  (‘Shackles’, 0.7814539074897766),
  (‘Breaker’, 0.7562315464019775),
  (‘warlocks’, 0.7459860444068909),
  (‘fairest’, 0.7372589111328125),
  (‘Grass’, 0.7342460751533508)

Linear Relationships

- ("Stark", "Winterfell", "Martell") #Leader

Linear Relationships

- `("Stark", "Winterfell", "Martell") #Leader`

  `Stark is related to Winterfell, as Doran is related to Martell`

Linear Relationships

- `("Stark", "Winterfell", "Martell") #Leader`

  `Stark is related to Winterfell, as Doran is related to Martell`
- `("Stark", "Winterfell", "Bolton") #Leader`

Linear Relationships

- `("Stark", "Winterfell", "Martell") #Leader`

  `Stark is related to Winterfell, as Doran is related to Martell`
- `("Stark", "Winterfell", "Bolton") #Leader`

  `Stark is related to Winterfell, as Roose is related to Bolton`

Linear Relationships

- ("Stark", "Winterfell", "Martell") #Leader

  Stark is related to Winterfell, as Doran is related to Martell
- ("Stark", "Winterfell", "Bolton") #Leader

  Stark is related to Winterfell, as Roose is related to Bolton
- ("Arya", "Horseface", "Daenerys") #Nicknames

Linear Relationships

- `("Stark", "Winterfell", "Martell") #Leader`

  `Stark is related to Winterfell, as Doran is related to Martell`
- `("Stark", "Winterfell", "Bolton") #Leader`

  `Stark is related to Winterfell, as Roose is related to Bolton`
- `("Arya", "Horseface", "Daenerys") #Nicknames`

  `Arya is related to Horseface, as Dany is related to Daenerys`

Linear Relationships

- ("Stark", "Winterfell", "Martell") #Leader

  Stark is related to Winterfell, as Doran is related to Martell
- ("Stark", "Winterfell", "Bolton") #Leader

  Stark is related to Winterfell, as Roose is related to Bolton
- ("Arya", "Horseface", "Daenerys") #Nicknames

  Arya is related to Horseface, as Dany is related to Daenerys
- ("Arya", "Nymeria", "dragons") # Mystic creatures

Linear Relationships

- ("Stark", "Winterfell", "Martell") #Leader
  Stark is related to Winterfell, as Doran is related to Martell
- ("Stark", "Winterfell", "Bolton") #Leader
  Stark is related to Winterfell, as Roose is related to Bolton
- ("Arya", "Horseface", "Daenerys") #Nicknames
  Arya is related to Horseface, as Dany is related to Daenerys
- ("Arya", "Nymeria", "dragons") # Mystic creatures
  Arya is related to Nymeria, as Dany is related to dragons

Linear Relationships

- ("Stark", "Winterfell", "Martell") #Leader
  Stark is related to Winterfell, as Doran is related to Martell
- ("Stark", "Winterfell", "Bolton") #Leader
  Stark is related to Winterfell, as Roose is related to Bolton
- ("Arya", "Horseface", "Daenerys") #Nicknames
  Arya is related to Horseface, as Dany is related to Daenerys
- ("Arya", "Nymeria", "dragons") # Mystic creatures
  Arya is related to Nymeria, as Dany is related to dragons
- ("Jon", "Snow", "Ellaria") # Bastards by area

Linear Relationships

- ("Stark", "Winterfell", "Martell") #Leader

  Stark is related to Winterfell, as Doran is related to Martell
- ("Stark", "Winterfell", "Bolton") #Leader

  Stark is related to Winterfell, as Roose is related to Bolton
- ("Arya", "Horseface", "Daenerys") #Nicknames

  Arya is related to Horseface, as Dany is related to Daenerys
- ("Arya", "Nymeria", "dragons") # Mystic creatures

  Arya is related to Nymeria, as Dany is related to dragons
- ("Jon", "Snow", "Ellaria") # Bastards by area

  Jon is related to Snow, as Sand is related to Ellaria

Who Doesn't Belong

- Even an algorithm can tell, who doesn't belong

Who Doesn't Belong

- Even an algorithm can tell, who doesn't belong

  `("Jaime, Cersei, Robert")`

Who Doesn't Belong

- Even an algorithm can tell, who doesn't belong
  `("Jaime, Cersei, Robert")`
  - `'Robert'`

Who Doesn't Belong

- Even an algorithm can tell, who doesn't belong `("Jaime, Cersei, Robert")`
  - `'Robert'`
- The Night is Dark and full of spoilers!

Who Doesn't Belong

- Even an algorithm can tell, who doesn't belong
  `("Jaime, Cersei, Robert")`
  - `'Robert'`
- The Night is Dark and full of spoilers!
  `("Robb, Jon, Arya, Sansa, Rickon, Brandon")`

Who Doesn't Belong

- Even an algorithm can tell, who doesn't belong
  `("Jaime, Cersei, Robert")`
  - `'Robert'`
- The Night is Dark and full of spoilers!
  `("Robb, Jon, Arya, Sansa, Rickon, Brandon")`
  - `'Jon'`

Who Doesn't Belong

- Even an algorithm can tell, who doesn't belong
  `("Jaime, Cersei, Robert")`
  - `'Robert'`
- The Night is Dark and full of spoilers!
  `("Robb, Jon, Arya, Sansa, Rickon, Brandon")`
  - `'Jon'`
- Season 8 predictions!

Who Doesn't Belong

- Even an algorithm can tell, who doesn't belong
  `("Jaime, Cersei, Robert")`
  - `'Robert'`
- The Night is Dark and full of spoilers!
  `("Robb, Jon, Arya, Sansa, Rickon, Brandon")`
  - `'Jon'`
- Season 8 predictions!
  `("Tyrion, Daenerys, Gendry, Bran, Jon")`

Who Doesn't Belong

- Even an algorithm can tell, who doesn't belong
  `("Jaime, Cersei, Robert")`
  - `'Robert'`
- The Night is Dark and full of spoilers!
  `("Robb, Jon, Arya, Sansa, Rickon, Brandon")`
  - `'Jon'`
- Season 8 predictions!
  `("Tyrion, Daenerys, Gendry, Bran, Jon")`
  - `'Daenerys'`

Conclusions

- Word2Vec can efficiently learn word-embeddings in a lower-dimension space such that similar words cluster together

Questions?