NEW ROLE SCHOOL

DATA SCIENTIST

M1 | NLP: Natural Language Processing

YOUR WAY TO THE FUTURE

GENERALI | We LEARN

About me

- Researcher at DI-UNIMI
  - mauricioabel.soto@polimi.it
- Previous
  - Researcher at DIG-POLIMI
  - Researcher - U. of Orléans, U. of Chile, UNIMIB
  - PhD. Computer Science - U. of Paris
  - Mathematical Engineering - U. of Chile
- Research interests
  - Graph theory
  - Evolutionary networks
  - Data representation (NLP)

Agenda

- Text representation
- Data Preparation for Textual Data
- Sentiment Analysis
- Topic Modeling
- Word embedding: Word2vec and Glove
- RNN - Long Short Term Memory networks (LSTM)

Slides and code: https://github.com/mauriciosotogomez/2023_GENERALI_M1_NLP

Sentiment Analysis

@VirginAmerica it was amazing, and arrived an hour early. You're too good to me.

Positive

@USAirways they finally answered. They were rude and didn't help me at all.

Negative

@united what do you have in mind?

Neutral

Topic Modeling

@united Cancelled Flightled my flight for some unknown reason and haven't really given me anything to make my overnight layover tolerable

Cancelled flight

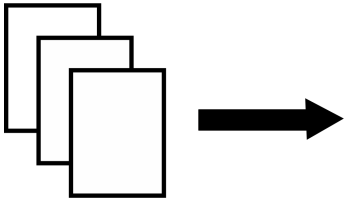
@USAirways they finally answered. They were rude and didn't help me at all.

Customer assistance

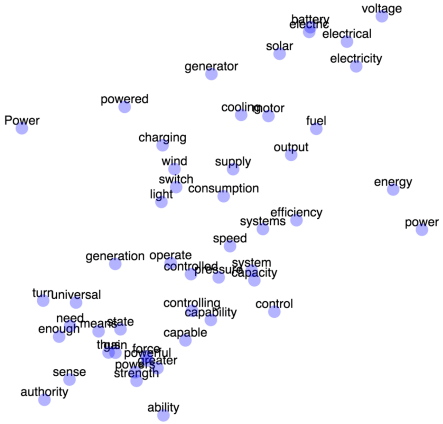
@united my friend lost luggage on flight 1547. What to do?

Luggage problems

Word embedding

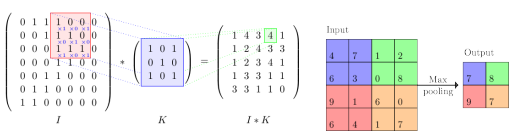


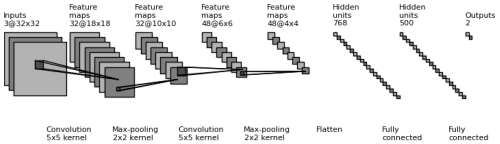
Corpus



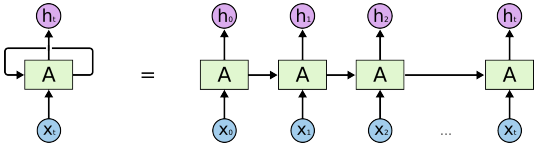
RNN

Convolutional Neural Networks



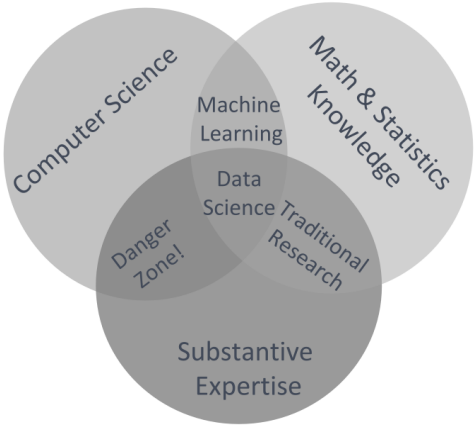


Recurrent Neural Networks



Data Science

- Drew Conway, 2010



@manudellavalle - http://emanueledellavalle.org

Text representation

Some important questions

- What is the meaning of a word?
- How we can represent it?
- Which are the limits/problems of a representation?

Question time

Go to https://pollev.com/mauriciosoto

Similarity vs Relatedness

Similarity

- Not necessary synonyms
- Sharing some element of meaning
- Examples
  - dog, cat
  - coffee, tea
  - short, long (antonyms)

Relatedness

- Also called “word association”
- Share semantic field
- Examples
  - car, gas
  - monitor, RAM
  - complexity, algorithm

A first (too simple) approach: One-Hot encoding

	1	2	3	4	5	6	7	8	9
The	1	0	0	0	0	0	0	0	0
quick	0	1	0	0	0	0	0	0	0
brown	0	0	1	0	0	0	0	0	0
fox	0	0	0	1	0	0	0	0	0
jumps	0	0	0	0	1	0	0	0	0
over	0	0	0	0	0	1	0	0	0
lazy	0	0	0	0	0	0	0	1	0
dog	0	0	0	0	0	0	0	0	1

A first (too simple) approach: One-Hot encoding

	1	2	3	4	5	6	7	8	9
The	1	0	0	0	0	0	0	0	0
quick	0	1	0	0	0	0	0	0	0
brown	0	0	1	0	0	0	0	0	0
fox	0	0	0	1	0	0	0	0	0
jumps	0	0	0	0	1	0	0	0	0
over	0	0	0	0	0	1	0	0	0
lazy	0	0	0	0	0	0	0	1	0
dog	0	0	0	0	0	0	0	0	1

Problem

- Does not capture relatedness/similarity
- Sparse representation

A second approach: frequency counting

Document-Term Matrix: word counts in matrix format

doc1: "...cell is the basic structure and functional unit of life forms..."

doc2: "...cancer is a group of diseases involving abnormal cell growth..."

doc3: ".. graph, which is a mathematical structure used to model connections between units..."

doc4: "...neural network is based on the connections of units called artificial neurons..."

	cell	is	neurons	disease	···	mathematical	connection	node
doc1	30	15	4	7	···	0	2	0
doc2	15	25	3	20	···	0	3	0
doc3	1	18	0	5	···	5	20	15
doc2	2	22	10	0	···	3	10	10

A second approach: frequency counting

Document-Term Matrix: word counts in matrix format

doc1: "...cell is the basic structure and functional unit of life forms..."

doc2: "...cancer is a group of diseases involving abnormal cell growth..."

doc3: ".. graph, which is a mathematical structure used to model connections between units..."

doc4: "...neural network is based on the connections of units called artificial neurons..."

	cell	is	neurons	disease	$\cdots$	mathematical	connection	node
doc1	30	15	4	7	$\cdots$	0	2	0
doc2	15	25	3	20	$\cdots$	0	3	0
doc3	1	18	0	5	$\cdots$	5	20	15
doc2	2	22	10	0	$\cdots$	3	10	10

Problem

- Frequent words do not characterize the documents.

A second approach: frequency counting

TF-IDF: Term Frequency - Inverse Document Frequency

$$\text{TF-IDF} = f_{t,d} \times \text{idf}(t, D)$$

where

- $f_{t,d}$: the number of times that term t occurs in document d.
- $\text{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}| + 1}$

A second approach: frequency counting

TF-IDF: Term Frequency - Inverse Document Frequency

$$\text{TF-IDF} = f_{t,d} \times \text{idf}(t, D)$$

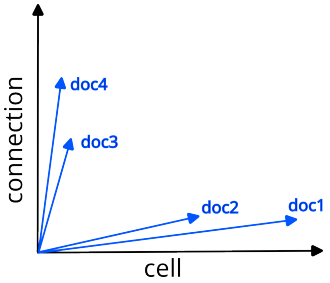
where

- $f_{t,d}$: the number of times that term t occurs in document d.
- $\text{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}| + 1}$

We reduce the importance of common words.

Cosine similarity

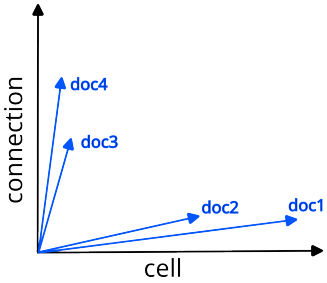
	cell	is	neurons	disease	···	mathematical	connection	node
doc1	30	15	4	7	···	0	2	0
doc2	15	25	3	20	···	0	3	0
doc3	1	18	0	5	···	5	20	15
doc2	2	22	10	0	···	3	10	10

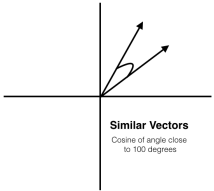


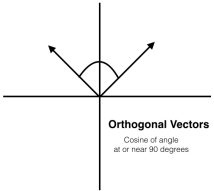
Cosine similarity

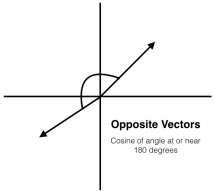
	cell	is	neurons	disease	⋯	mathematical	connection	node
doc1	30	15	4	7	⋯	0	2	0
doc2	15	25	3	20	⋯	0	3	0
doc3	1	18	0	5	⋯	5	20	15
doc2	2	22	10	0	⋯	3	10	10

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{\|\vec{v}\|\|\vec{w}\|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \in [-1,1]$$









A third approach: frequency counting

Term-Context Matrix: co-occurrence word counts

doc1: "I like deep learning."

doc2: "I like NLP."

doc3: "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

Data Preparation for Textual Data

Text Cleaning

- Convert to lower case
- Remove punctuation
- Remove numerical values
- Typos
- Remove special characters ([?@)
- Remove stop words (the, it, etc)
- Remove special description words ([chorus], [fade], [applause])
- Tokenize text (O'Neill → [o] [neill], [o'neill]?; aren't → [arent], [are][nt]?)
- Create bi-grams or tri-grams ([United Kingdom] vs [United][Kingdom])
- Normalization:
  - Stemming (car, cars, car's, cars' → car;)
  - Lemmatization (am, are, is → be)

Sentiment Analysis

Sentiment Analysis

@VirginAmerica it was amazing, and arrived an hour early. You're too good to me.

Positive

@USAirways they finally answered. They were rude and didn't help me at all.

Negative

@united what do you have in mind?

Neutral

Sentiment Analysis

The simple strategy: We assign an score to phrases based on predefined rules on the words.

Sentiment Analysis

The simple strategy: We assign an score to phrases based on predefined rules on the words.

• VADER: Valence Aware Dictionary and sEntiment Reasoner
Phrases are labeled according to proportions of text in categories **pos,neg, neu**

• TextBlob: : Linguistic labeled the sentiment of words.
Sentiment Labels: Each word is labeled in terms of

- Polarity: negative(-1) or positive(+1)
- Subjectivity: subjective(0) or fact(+1)

The customize strategy: Create a classification model base on (annotated) historical data.

Topic Modelling

Topic Modeling

@united Cancelled Flightled my flight for some unknown reason and haven't really given me anything to make my overnight layover tolerable

Cancelled flight

@USAirways they finally answered. They were rude and didn't help me at all.

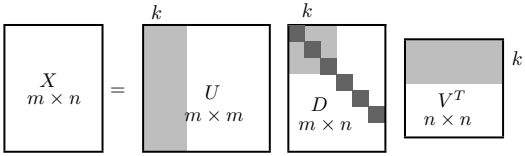
Customer assistance

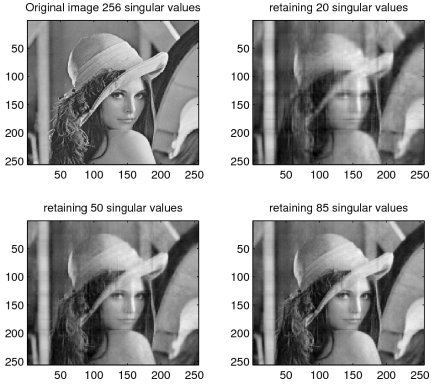
@united my friend lost luggage on flight 1547. What to do?

Luggage problems

Topic Modeling - LSA

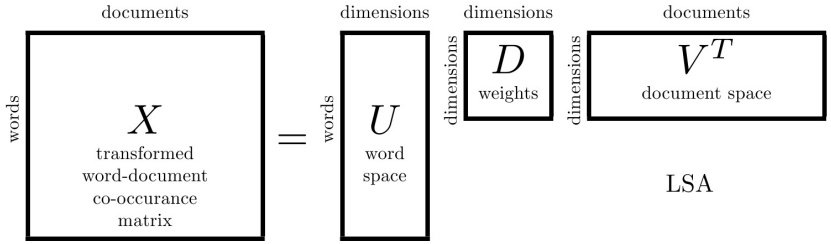
Latent Semantic Analysis: Singular Value Decomposition (SVD)



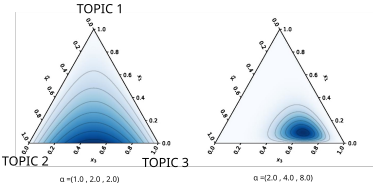


Topic Modeling - LSA

Latent Semantic Analysis: Singular Value Decomposition (SVD) of the Document-Term Matrix

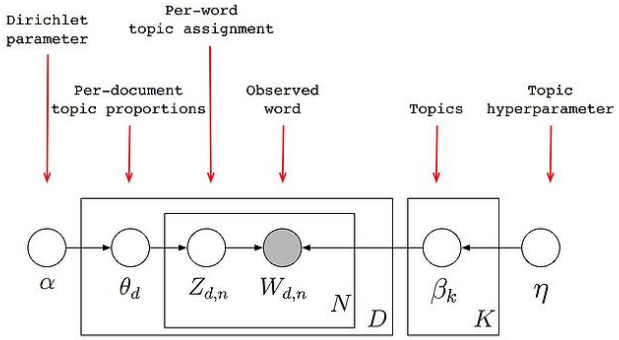


Topic Modeling - LDA



Latent Dirichlet Allocation (LDA)

- **Documents are probabilistic distribution over topics**: let say that a document is $p_i\%$ of topic i.
- **Topics are probabilistic distribution over words**: given a topic chosen according to the distribution of the document, we generate a word according to the topic distribution

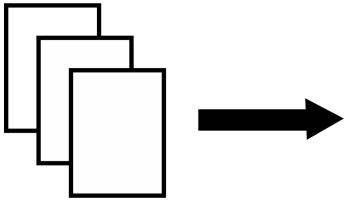


Topic Modeling - LDA

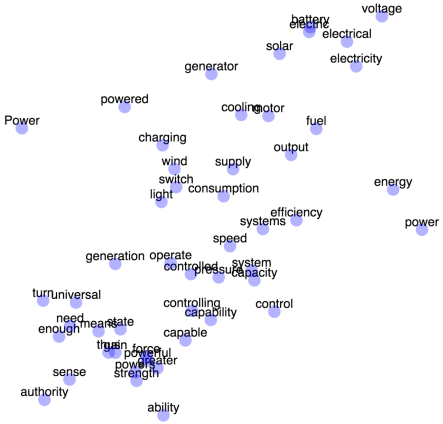
1. Random initialisation: assign each word to a random topic
2. Update each word by considering
   - proportion of words in the document of topic
   - proportion of topics in all documents for the word
3. We sample until a “reasonable” result

Word embedding

Word embedding



Corpus



Word Embedding

Distributed Representations of Words (a.k.a. word embeddings) are geometric representation of words/entities learned from the data/corpus in such a way that semantically related words are often close to each other.

In practice we attempt to embed entities onto a low -dimensional metric space in which similar words are placed close

Distributional Hypothesis in action

What's the meaning of '**bardiwac**'?[1]

- He handed her 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 on bread and cheese and this excellent bardiwac
- The drinks were delicious: blood-red bardiwac as well as light, sweet Rhenish

[1] Lenci & Evert 2010

Distributional Hypothesis in action

What's the meaning of '**bardiwac**'?[1]

- He handed her 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 on bread and cheese and this excellent bardiwac
- The drinks were delicious: blood-red bardiwac as well as light, sweet Rhenish

Bardiwac is a heavy red alcoholic beverage made from grape

[1] Lenci & Evert 2010

Distributional hypothesis

“The meaning of a word is its use in the language”

(Wittgenstein, 1953)

“You shall know a word by the company it keeps”

(Firth, 1957)

Distributional Hypothesis: similar words tend to appear in similar contexts

(Harris, 1954)

Distributional hypothesis

"The meaning of a word is its use in the language"

(Wittgenstein, 1953)

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

(Firth, 1957)

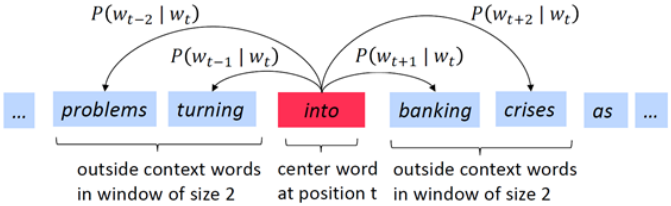
Distributional Hypothesis: similar words tend to appear in similar contexts

(Harris, 1954)

...therefore a word can be represented based on the co-occurrence across the data in the same context.

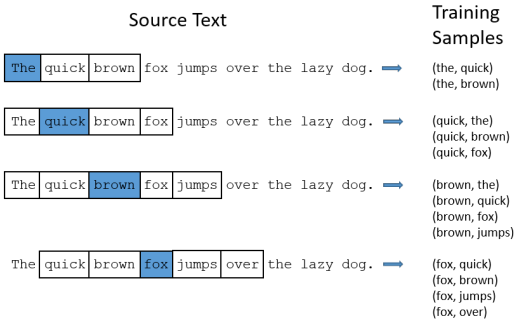
Word2vec (Skip-gram)

- Mikolov et al. (2013). *Efficient Estimation of Word Representations in Vector Space*
  - Construct a vectorial representation of words from a neural network for a classification task .
  - Task: Given a word, we aim to predict the if another word is in the same context.
  - Context is given by a window around each word (self-supervision)

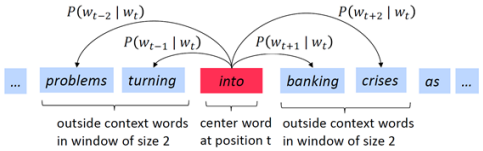


http://web.stanford.edu/class/cs224n/

Word2Vec: Training examples



2

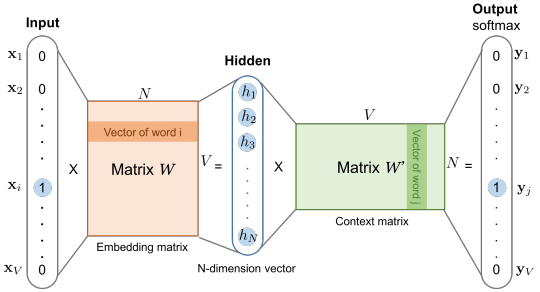


3

[2] http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

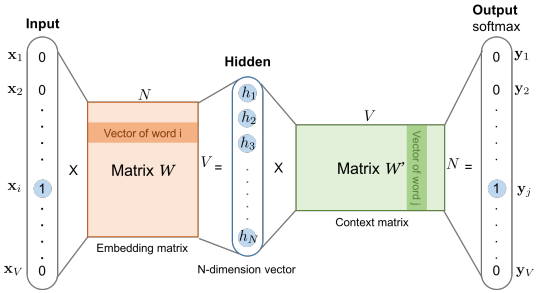
[3] http://web.stanford.edu/class/cs224n/

Word2Vec (Skip-gram) architecture



$$\underbrace{e_i^\top \times W_{N\times d}}_{v_i^{(c)}} \times W'_{d\times N} \xrightarrow{\text{softmax}} \frac{exp(\overbrace{v_w^{(o)} \cdot v_i^{(c)}}^{\text{similarity between i and j}})}{\sum_{w=1}^{V} exp(v_w^{(o)} v_i^{(c)})} = \mathbb{P}(\text{word}_j^{(o)} | \text{word}_i^{(c)})$$

Word2Vec (Skip-gram) architecture

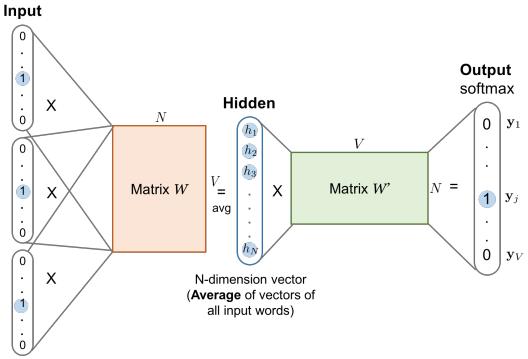


$$\underbrace{e_i^\top \times W_{N\times d}}_{v_i^{(c)}} \times W'_{d\times N} \rightarrow^{\text{softmax}} \frac{exp(\overbrace{v_w^{(o)} \cdot v_i^{(c)}}^{\text{similarity between i and j}})}{\sum_{w=1}^{V} exp(v_w^{(o)} v_i^{(c)})} = \mathbb{P}(\text{word}_j^{(o)} | \text{word}_i^{(c)})$$

Notice that each have two vector representations:
- as a **center** word $v^{(c)}$ (row of embedding matrix W), and
- as a **context** word $v^{(o)}$ (column of context matrix W').

Final vector of v is usually defined as $(v_c + v_o)$

Word2Vec CBOW



$$v = \frac{1}{|\text{window}|} \sum_{i=1}^{|\text{window}|} e_i^\top \times W_{N \times d}$$

https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

Word2Vec Loss function

For each word $w_t, t \in \{1, \ldots, T\}$ in the text we maximize the probability of any context word in a m window given the center word along all the text (by assuming that context word are independent)

$$\prod_{t=1}^{T} \prod_{\substack{-m \leqslant j \leqslant m \\ j \neq 0}} p\left(w_{t+j} \mid w_t; \theta\right)$$

or equivalently we minimize its **Negative Log Likelihood**

$$-\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \leqslant j \leqslant m \\ j \neq 0}} \log(p\left(w_{t+j} \mid w_t\right))$$

where $p(o|c) = exp(v_o^\top v_c) / \sum_{w=1}^{V} exp(u_w^\top v_C)$

Computational expensive!

Word2Vec - Negative sampling

- Mikolov et al. (2013). *Distributed representations of words and phrases and their compositionality*. Advances in Neural Information Processing Systems.

1. Subsampling frequent words according to its frequency to reduce training examples.
2. Modify the objective function using "Negative Sampling", which reduce weights update process.

Word2Vec - Negative sampling

- Mikolov et al. (2013). *Distributed representations of words and phrases and their compositionality.* Advances in Neural Information Processing Systems.

1. Subsampling frequent words according to its frequency to reduce training examples.
2. Modify the objective function using "Negative Sampling", which reduce weights update process.

Negative sampling For each positive couple

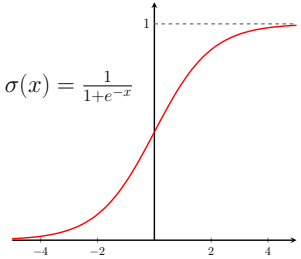
- We take k negative samples according to the probability $\mathrm{P}(w) \propto \mathrm{freq}(w)^{3/4}$ (the power makes less frequent words be sampled more often)
- **Maximize** probability that real outside word appears a and **minimize** the probability that random words appear around center.

Word2Vec Loss function with negative sampling

For a positive example (v_c, u_o), the new loss function with negative examples $\{(v_c, u_1), \ldots, (v_c, u_k)\}$ will be :

$$Loss = -\log\left(\sigma(u_o^T v_c)\right) - \sum_{i=1}^{k} \log\left(\sigma(-u_j^T v_c)\right)$$

where σ denotes the sigmoid function

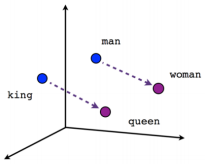


Analogical Reasoning

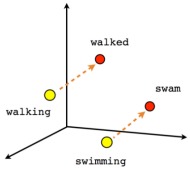
The city of Rome is in relation with the country Italy in the same way as the city of Paris is in relation with the country France.

The propositional analogy task: find an ?x such that

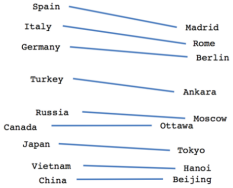
$$\text{Rome} : \text{Italy} = ?x : \text{France}$$



Male-Female



Verb tense



Country-Capital

$$v(Italy) - v(Rome) \sim v(France) - v(Paris)$$

Glove

- Keffrey Pennington, Richard Socher, and Christopher D. Manning. (2014). *GloVe: Global Vectors for Word Representation*.

- Use the co-occurrence matrix for the entire corpus.
- As in SVD we try to encode a matrix into some "principal componets"
- As in word2vec the probability o the context word given the central word is proportional to the dot product of vector representations.

doc1: "I like deep learning."

doc2: "I like NLP."

doc3: "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

Glove

- Keffrey Pennington, Richard Socher, and Christopher D. Manning. (2014). *GloVe: Global Vectors for Word Representation.*

- Use the co-occurrence matrix for the entire corpus.
- As in SVD we try to encode a matrix into some "principal componets"
- As in word2vec the probability o the context word given the central word is proportional to the dot product of vector representations.

doc1: "I like deep learning."

doc2: "I like NLP."

doc3: "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

Minimize the function $J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f\left(P_{ij}\right) \left(u_i^T v_j - \log P_{ij}\right)^2$

Glove

- Keffrey Pennington, Richard Socher, and Christopher D. Manning. (2014). *GloVe: Global Vectors for Word Representation.*

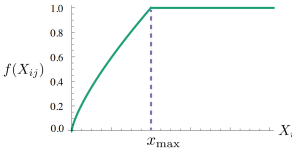
- Use the co-occurrence matrix for the entire corpus.
- As in SVD we try to encode a matrix into some "principal componets"
- As in word2vec the probability o the context word given the central word is proportional to the dot product of vector representations.

doc1: "I like deep learning."
doc2: "I like NLP."
doc3: "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

Minimize the function $J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f\left(P_{ij}\right) \left(u_i^T v_j - \log P_{ij}\right)^2$

where $f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$



Word embedding issues

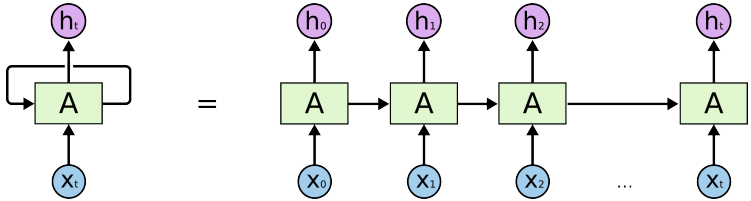
- Just one sense per word (apple vs Apple)
- human biases from the data they are trained on
- fail to capture higher-level information

Implementation and datasets

- Python code: https://github.com/mauriciosotogomez/NLP/
- Neural Network: Word2Vec [Mikolov+, 2013]
  https://code.google.com/archive/p/word2vec
- GloVe [Pennington+,2014]. Matrix Factorization/Neural Network.
  https://nlp.stanford.edu/projects/glove/

RNN: Recurrent Neural Networks

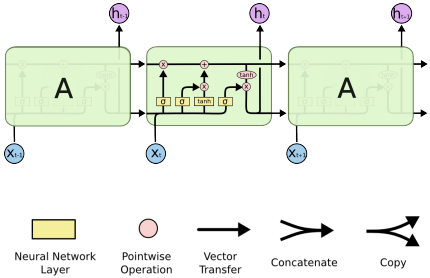
Given a sequence (of words): $x = x_1 x_2 \cdots x_t$



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM: Long Short Term Memory networks

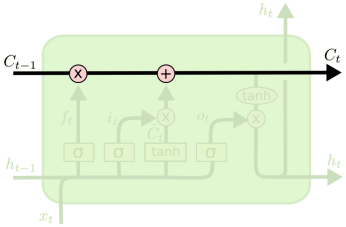
1. We keep a cell state across the sequence C_t
2. After each step t we:
   - forget something: f_t
   - include something : i_t
   - update the cell state: C_t
   - output something to the next step: h_t



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM: Keep Global state

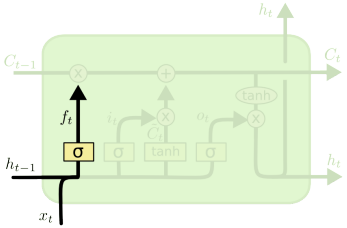
1. **We keep a cell state across the sequence** C_t
2. After each step t we:
   - forget something: f_t
   - include something : i_t
   - update the cell state: C_t
   - output something to the next step: h_t



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM: forget gate state

1. We keep a cell state across the sequence C_t
2. After each step t we:
   - **forget something**: f_t
   - include something : i_t
   - update the cell state: C_t
   - output something to the next step: h_t

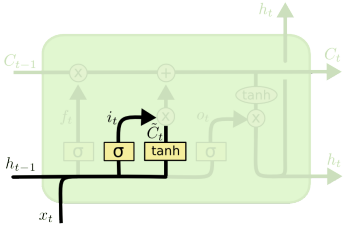


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM: input gate state

1. We keep a cell state across the sequence C_t
2. After each step t we:
   - forget something: f_t
   - **include something** : i_t
   - update the cell state: C_t
   - output something to the next step: h_t



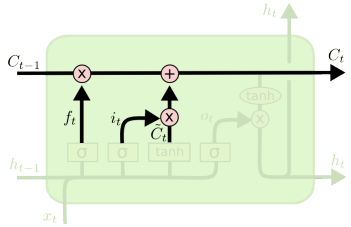
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM: update cell state

1. We keep a cell state across the sequence C_t
2. After each step t we:
   - forget something: f_t
   - include something : i_t
   - **update the cell state**: C_t
   - output something to the next step: h_t

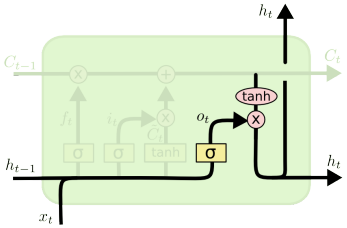


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM: cell output

1. We keep a cell state across the sequence C_t
2. After each step t we:
   - forget something: f_t
   - include something : i_t
   - update the cell state: C_t
   - **output something to the next step**: h_t



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Generating text

1. From the text, we create a training set form by couples $([x_1, \ldots, x_t], y_t)$ where:
   - $[x_1, \ldots, x_t]$ is a sequence of t elements (letters, words)
   - y_t is the element to be predicted
2. From a seed sequence we sequentially generate the text consider as input the last sequence.