NLP BASICS

Introduction to Word Vectors

How do we represent the meaning of a word in a machine?

Three eras of NLP

A brief history of NLP

Symbolic
1940 - 2000

Rule-based systems
Formal grammars

Lexicon, ontologies, grammars

Statistical Learning
1990 - 2010

Statistical learning theory, Graphical probabilistic models (e.g. LSA, HMM)

Annotated datasets

Deep Learning
2010 - now

Deep learning based methods,
Transfer learning in NLP (BERT, GPT)

Larger datasets, open source libraries (hugging face)

Units in NLP

Corpus: A collection of *m* documents

$$C = (d_1, d_2, ..., d_m)$$

(Book(s), wikipedia, all articles of the NYT)

Document: A sequence of *k* words

$$D = (w_1, w_2, ..., w_k)$$

(Sentence, paragraph, sequence of paragraphs)

Token: A basic unit of a sequence of characters grouped together for processing

(Word, sub-word, character(s))

Preprocessing

Preprocessing: Tokenization

- Chunk a character sequence into smaller discrete element(s) (sentence, word, sub-word)

Input: All I know is that I know nothing

Output: ['All', 'I', 'know', 'is', 'that', 'I', 'know', 'nothing']

- The first step in any NLP pipeline
- Challenge: Mostly language-agnostic, but different language systems require other algorithms (e.g. Arabic, Chinese, Korean, Tamil, Urdu, and others)

Preprocessing: Stemming

- Used to “normalise” word into base form or root form.

  Input: celebrates, celebrated, celebrating
  Output: celebrate

- Challenge: can produce a root word which may not have any meaning

  Input: intelligence, intelligent, intelligently
  Output: intelligen

Preprocessing: Lemmatization

- Reduce inflectional/variant forms to base form

  Input: am/are/is Output: be

  Input: car/cars/car’s/cars’ Output: car

  Input: the boy's cars are different colors

  Output: the boy car be different color

- Lemmatization produces the root word which has a meaning.

Information Extraction

Information extraction: Part of Speech Tagging

- Annotate each word in a sentence with a part-of-speech marker.

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun

Tag	Description
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Whdeterminer
WP	Whpronoun
WP$	Possessive whpronoun
WRB	Whadverb

Information extraction: Part of Speech Tagging

- Annotate each word in a sentence with a part-of-speech marker.

  Input: Barack Obama was born in Hawaii and served as the 44th President of the United States.

  Output: $[\text{Barack}]_{NNP}$ $[\text{Obama}]_{NNP}$ $[\text{was}]_{VBD}$ $[\text{born}]_{VBN}$ $[\text{in}]_{IN}$ $[\text{Hawaii}]_{NNP}$ $[\text{and}]_{CC}$ $[\text{served}]_{VBD}$ $[\text{as}]_{IN}$ $[\text{the}]_{DT}$ $[\text{44th}]_{JJ}$ $[\text{President}]_{NN}$ $[\text{of}]_{IN}$ $[\text{the}]_{DT}$ $[\text{United}]_{NNP}$ $[\text{States}]_{NNPS}$ $[.]_{.}$

- Lowest level of syntactic analysis.

Information extraction: Named Entity Recognition

- Identify and categorize the text into specific entities
- Entities: who did what, when, where, why

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	The **Mt. Sanitas** loop is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states, provinces	**Palo Alto** is raising the fees for parking.
Facility	FAC	bridges, buildings, airports	Consider the **Tappan Zee Bridge**.
Vehicles	VEH	planes, trains, automobiles	It was a classic **Ford Falcon**.

Figure 21.1 A list of generic named entity types with the kinds of entities they refer to.

Information extraction: Named Entity Recognition

- Identify and categorize the text into specific entities
- Entities: who did what, when, where, why

Input: Barack Obama was born in Hawaii and served as the 44th President of the United States.

Output: [Barack Obama]$_{PER}$ was born in [Hawaii]$_{GPE}$ and served as the [44th]$_{ORD}$ President of the [United States]$_{GPE}$.

Let’s code it! 👨‍💻👩‍💻

NLP modeling framework

- Modeling an NLP task involves estimating the conditional probability

$$p(Y|X)$$

Tokens

- **Sequence classification** task: Y is a single label
- **Sequence labelling** task: Y has one label per token
- **Sequence prediction** task: Y is a sequence of tokens
- **Structure prediction** task: Y is a graph or a tree

Representing text with vectors

- Let’s assume “token = word"
- We can represent words in vectors by applying different techniques
  - One-hot encoding
  - Hand-crafted representations
  - Count-based representations
  - Learning-based representations
- Word vectors and word embeddings are used interchangeably

One-hot encoding

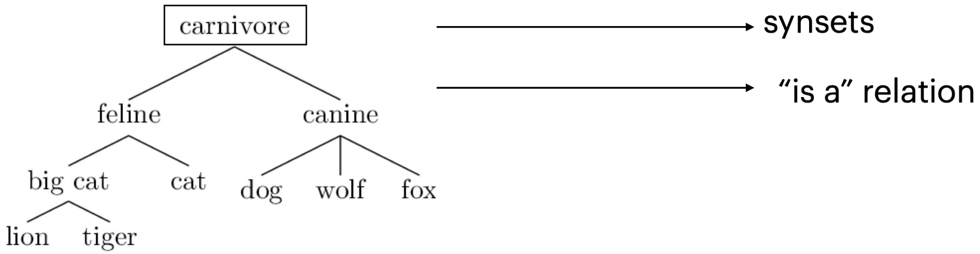
- Every word is represented in an element of a vector
- A word vector consists of 0s everywhere except a single 1 for the word

human	1	0	0	0	…	0
machine	0	1	0	0	…	0
system	0	0	1	0	…	0
for	0	0	0	1	…	0
…	…	…	…	…	…	…
user	0	0	0	0	…	1

- <u>Challenge</u>: Polysemy (river bank, financial institution bank, bank for sitting)

Hand-crafted representations: WordNet

- WordNet (see also VerbNet, FrameNet)
- Manually annotated lexical database of words and their semantic relationships



is_similar(dog, lion) = ?

- Challenge: Requires a lot of human labor, subjectivity of annotators, does not scale

Word representations using data

- Representing words by their context: **Distributional semantics**

"You shall know a word by the company it keeps." (Firth, 1957)

I deposited my pay check at the **bank** this morning.

I need to visit the **bank** to withdraw some cash for my trip.

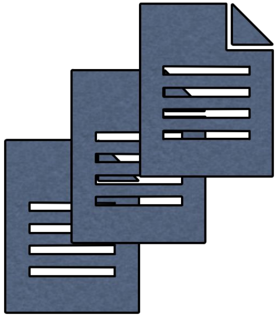
I need to transfer some funds at the **bank**.

nearby context words
will represent
bank

Idea: Consider the context of a word to build its vector representation

Co-occurrence matrix

Text corpus → Distributions of words in the text



	human	machine	system	for	…	user
human	0	2	1	0	…	0
machine	2	0	0	0	…	0
system	1	0	0	0	…	1
for	0	0	0	0	…	0
…	…	…	…	…	…	…
user	0	0	1	0	…	0

1. Define the context of a word
2. Count how many times a word co-occurs in this context

Co-occurrence matrix

The **human** operated the **machine** to complete the task efficiently.

The **user** relied on the **machine**'s accuracy to obtain the desired result.

Humans and **machine** collaborate to achieve optimal **system** performance.

The **system** was tailored to meet the needs of different **users**.

word embedding

	human	machine	system	for	…	user
human	0	2	1	0	…	0
machine	2	0	0	0	…	0
system	1	0	0	0	…	1
for	0	0	0	0	…	0
…	…	…	…	…	…	…
user	0	0	1	0	…	0

Co-occurrence matrix
Represents the frequency of word co-occurrences within a specified window of text

Co-occurrence matrix

The **human** operated the **machine** to complete the task efficiently.

The **user** relied on the **machine**'s accuracy to obtain the desired result.

Humans and **machine** collaborate to achieve optimal **system** performance.

The **system** was tailored to meet the needs of different **users**.

word embedding

	human	machine	system	for	…	user
human	0	2	1	0	…	0
machine	2	0	0	0	…	0
system	1	0	0	0	…	1
for	0	0	0	0	…	0
…	…	…	…	…	…	…
user	0	0	1	0	…	0

Limitations

Word frequency is very skewed —> biases the representations

Good embeddings depends on the size of the corpus

Solution: Pointwise Mutual Information

Use the probability of co-occurrences instead of absolute counts

$$PMI(w_1, w_2) = log \frac{p(w_1, w_2)}{p(w_1)p(w_2)}$$

Idea: Do words w_1,w_2 co-occur more than if they were independent?

	human	machine	system	for	…	user
human	0	2	1	0	…	0
machine	2	0	0	0	…	0
system	1	0	0	0	…	1
for	0	0	0	0	…	0
…	…	…	…	…	…	…
user	0	0	1	0	…	0

→

	human	machine	system	for	…	user
human	0	1.54	2.94	0	…	0
machine	1.54	0	0	0	…	0
system	2.94	0	0	0	…	2.54
for	0	0	0	0	…	0
…	…	…	…	…	…	…
user	0	0	2.54	0	…	0

Solution: Pointwise Mutual Information

Use the probability of co-occurrences instead of absolute counts

$$PMI(w_1, w_2) = log \frac{p(w_1, w_2)}{p(w_1)p(w_2)}$$

Less sensitive to change in frequent words

Idea: Do words w_1,w_2 co-occur more than if they were independent?

	human	machine	system	for	…	user
human	0	2	1	0	…	0
machine	2	0	0	0	…	0
system	1	0	0	0	…	1
for	0	0	0	0	…	0
…	…	…	…	…	…	…
user	0	0	1	0	…	0

	human	machine	system	for	…	user
human	0	1.54	2.94	0	…	0
machine	1.54	0	0	0	…	0
system	2.94	0	0	0	…	2.54
for	0	0	0	0	…	0
…	…	…	…	…	…	…
user	0	0	2.54	0	…	0

Solution: Pointwise Mutual Information

Use the probability of co-occurrences instead of absolute counts

$$PMI(w_1, w_2) = log\frac{p(w_1, w_2)}{p(w_1)p(w_2)}$$

Less sensitive to change in frequent words

Idea: Do words w_1,w_2 co-occur more than if they were independent?

Limitations:

- Very large matrix and word vectors
- Corpus dependency: Words and columns represent words in the corpus!

Solution: Pointwise Mutual Information

Use the probability of co-occurrences instead of absolute counts

$$PMI(w_1, w_2) = log \frac{p(w_1, w_2)}{p(w_1)p(w_2)}$$

Less sensitive to change in frequent words

Idea: Do words w_1,w_2 co-occur more than if they were independent?

Limitations:

- Very large matrix and word vectors
- Corpus dependency: Words and columns represent words in the corpus!

—> A solution: Dimensionality reduction (Singular Value Decomposition)

Let’s code it! 👨‍💻👩‍💻

Learning based representations

Co-occurrence vectors are

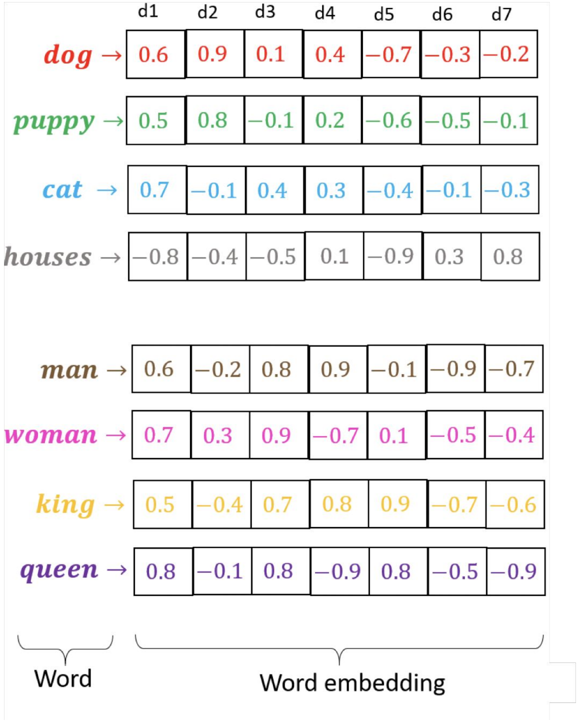
- Long (#unique words in corpus)
- Sparse (most elements are 0)

Alternative: learn vectors which are

- Short (~300 dimensions)
- Dense (most elements are non-zero)

Learning based representations

- Building a vector for each word
- Learning objective:
  - A word vector is similar to vectors of words that appear in similar contexts
- Similarity measurement: Cosine similarity of two vectors



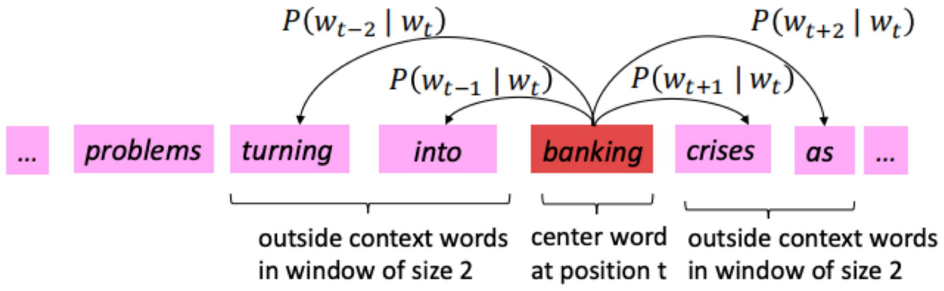
Rozado, PlosOne, 2020

Learning based representations

word2vec: An algorithm for learning word vectors

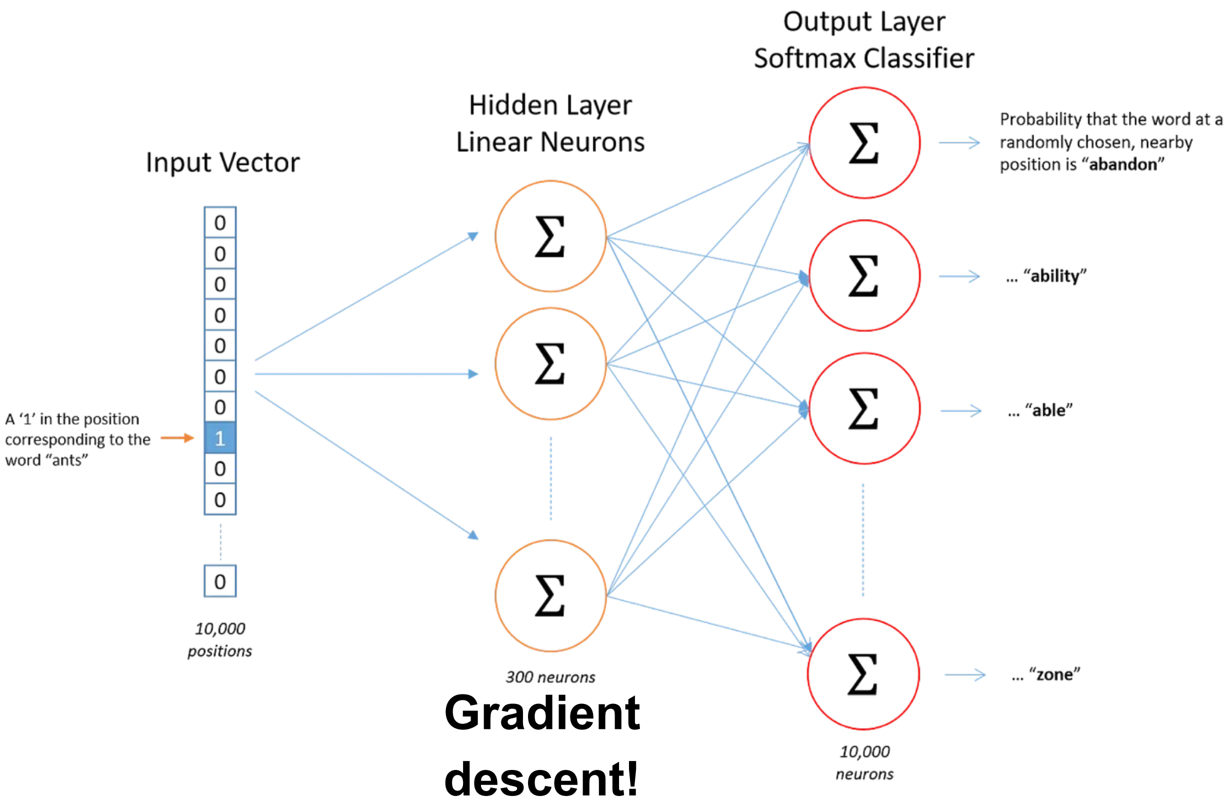
1. Take a large corpus of text
2. Every word is represented by a vector: random initialization
3. Pick a center word w
4. Pick context word c surrounding the word w
5. Use the similarity of word vectors for w and c to compute p(c|w)
6. Repeat to maximize the probability

word2vec



Hewitt (CS2224N, Stanford)

The learned weights in the network are the word embeddings



Gradient descent!

word2vec: objective function

For each position t=1,...T, predict context words within a window size m given a center word w_t for all parameters theta:

Likelihood

$$L_\theta = \prod_{t=1}^{T} \prod_{-m \leq j \leq m} p(w_{t+j} | w_t; \theta)$$

word2vec: objective function

For each position t=1,...T, predict context words within a window size m given a center word w_t for all parameters theta:

Likelihood $$argmax_\theta(L_\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m} p(w_{t+j} | w_t; \theta)$$

Maximizing Likelihood

word2vec: objective function

For each position t=1,...T, predict context words within a window size m given a center word w_t for all parameters theta:

Likelihood

$$argmax_\theta(L_\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m} p(w_{t+j} | w_t; \theta)$$

Objective Function

$$argmin_\theta(-\frac{1}{T} log(L_\theta)) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m} log(p(w_{t+j} | w_t; \theta))$$

Maximizing likelihood <=> Minimizing negative log likelihood

word2vec: objective function

For each position t=1,...T, predict context words within a window size m given a center word w_t for all parameters theta:

Likelihood $$argmax_\theta(L_\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m} p(w_{t+j}|w_t;\theta)$$

Objective Function $$argmin_\theta(-\frac{1}{T}log(L_\theta)) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{-m \leq j \leq m} \boxed{log(p(w_{t+j}|w_t;\theta))}$$

Maximizing likelihood <=> Minimizing negative log likelihood

word2vec: objective function

Calculating the conditional probability

How to calculate the conditional probability?

$$p(w_{t+j}|w_t)$$

Intuition: Use two vectors per word

w when the word is a center word

c when the word is a context word

Familiar?

$$p(c|w) = \boxed{\frac{exp\boxed{(c^T w)}}{\boxed{\sum_{v \in V} exp(v^T w)}}}$$

Dot product measures similarity of w and c

Normalize over entire vocabulary V

word2vec: objective function

$len(V) = 10000$

$v \in V$

$$argmin_\theta(J_\theta) = argmin_\theta(-\frac{1}{T}log(L_\theta)) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m} \boxed{log(p(w_{t+j}|w_t;\theta))}$$

$$\boxed{log(p(c|w)) = c^T w - log(\sum_{v\in V} exp(v^T w))}$$

$$p(c|w) = \frac{exp(c^T w)}{\sum_{v\in V} exp(v^T w)}$$

Negative log-likelihood is computed for every (at every iteration)

—> Very expensive to compute (for —> 3 million weights)

Solution: Negative sampling

word2vec: objective function

$$p(c|w) = \frac{exp(c^T w)}{\sum_{v \in V} exp(v^T w)}$$

$$softmax(x_i) = p_i = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)}$$

- This is a softmax of dot products between context and word vectors
- A softmax transforms a vector of numbers x_i into a probability distribution p_i
- Frequently used as the activation function in neural network

word2vec: objective function

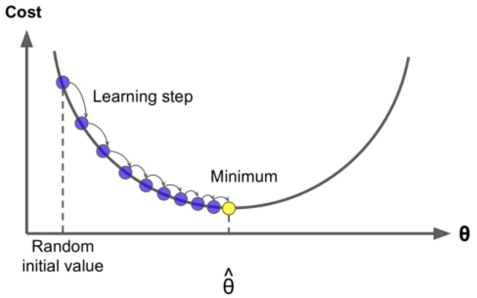
- Negative sampling

At each iteration update only a small percentage of the model’s weights

Negative log-likelihood is computed over K<<V words that are not in the context of w

Optimization through gradient descent

We have a objective function J_theta that we want to minimize



Use gradient descent to minimize J_theta

Idea:

For the current value of thetae, calculate the gradient of J_theta

Take the small step in the direction of the genitive gradient

Repeat

Hewitt (CS2224N, Stanford)

Algorithm 1 Skip-Gram Word2vec Training

Given a corpus C, made of a set of unique tokens V. Hyperparameters: number of negative samples K, a window size l, dimension of word vectors d, learning rate (α_t)

Initalize Randomly: $\mathbf{W} \in \mathbb{R}^{(V,d)}$ and $\mathbf{C} \in \mathbb{R}^{(V,d)}$

for `step` t `in` $0..T$ **do**

    ### Step 1: Sampling

    Sample $s = (w_1, .., w_n) \in C$ # a sequence in your corpus (e.g. sentence)

    Sample a pair $(i, j) \in [1, .., n]$ with $|i - j| \leq l$

    we note $w = w_i$, $c = w_j$ represented by vectors $\mathbf{w}$ in $\mathbf{W}$ and $\mathbf{c}$ in $\mathbf{C}$

    Sample $N_K = \{v_1, .., v_K\} \subset V$ represented by $\{\mathbf{v}_1, .., \mathbf{v}_K\}$ in $\mathbf{C}$ # Negative samples

    ### Step 2: Compute loss

    $l(\mathbf{W}, \mathbf{C}) = -\sigma(\mathbf{w}, \mathbf{c}) - \frac{1}{K} \sum_{v \in N_K} \log \sigma(\text{-}\mathbf{w}, \mathbf{v})$

    ### Step 3: Parameter update with SGD

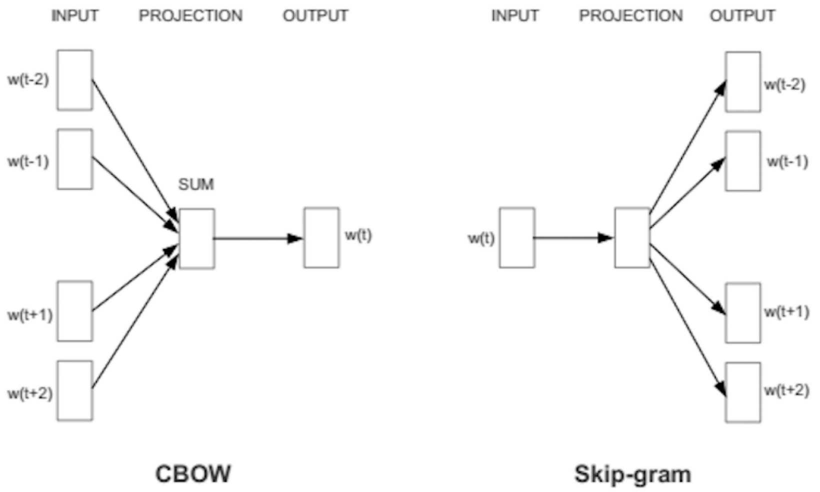
    $\mathbf{W}_t = \mathbf{W}_{t-1} - \alpha_t . \nabla\, l(\mathbf{W}_{t-1}, \mathbf{C}_{t-1})$

    $\mathbf{C}_t = \mathbf{C}_{t-1} - \alpha_t . \nabla\, l(\mathbf{W}_{t-1}, \mathbf{C}_{t-1})$

end

Word2vec: other variations

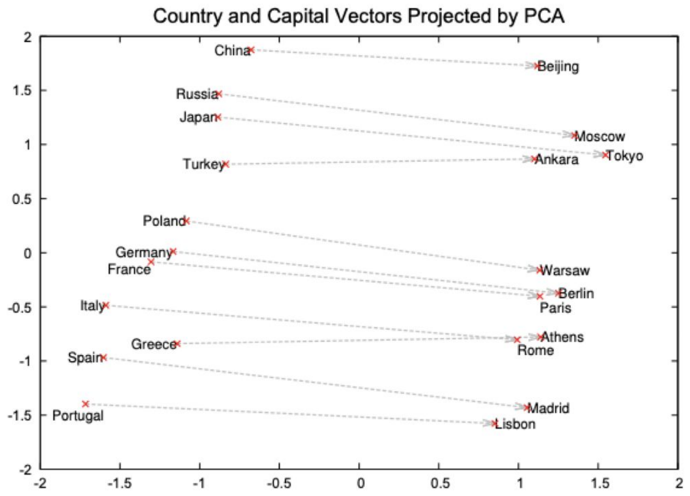
Continuous Bag of Words(CBOW) vs. skip-gram



Center word prediction

Context word prediction

Evaluation of word embeddings



How can we evaluate the quality of word embeddings?

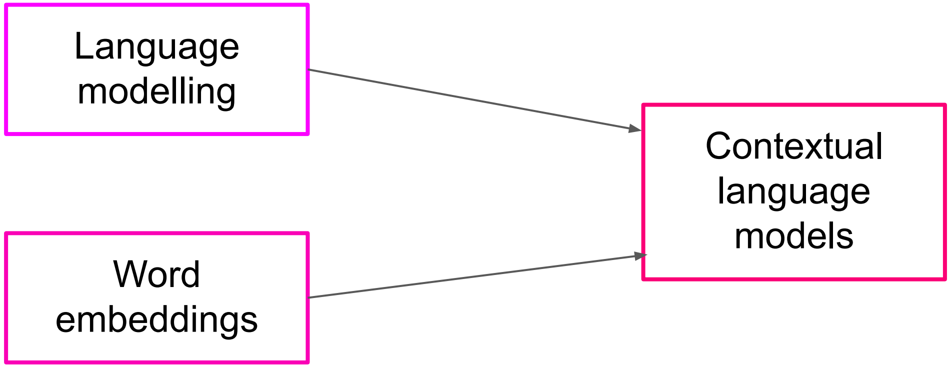
- Idea: Similar words should have similar vectors
- Visualize word embeddings
  - High dimensional! —> PCA or T-SNE
- Similarity measurements
  - Cosine similarity : $similar(w_1, w_2) = \frac{w_1^T w_2}{||w_1|| ||w_2||}$
  - L2 distance : $similar(w_1, w_2) = ||w_1 - w_2||$
- Similarity to human judgment (e.g.WordSim353 dataset)

Word2vec

- word2vec is still widely used
- but “contextualised” language modelling took over in the last years (e.g., BERT)
- Different variations and extensions exist:
  - Continuous Bag of Words (CBOW),
  - GloVe
- Multilingual version by building shared word embeddings across languages
  - FastText
- Limitations:
  - Trained on fixed vocabulary
  - Each token has a unique representation (e.g. the word “bank”)

Let’s code it! 👨‍💻👩‍💻

Tomorrow: Language modelling



Bibliography and Acknowledgement

Benoit and Sagot 2022, “Algorithms for speech and natural language processing”, MVA course material

Muller, “The Basics of Natural Language Processing” course material

Manning, “Natural Language Processing with Deep Learning” course material

Chris McCormick’s blog: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Mikolov et al., Distributed Representations of Words and Phrases and their Compositionality, 2013, https://arxiv.org/pdf/1310.4546.pdf

Mikolov et al. Efficient estimation of word representations in vector space. ICLR Workshop, 2013, https://arxiv.org/pdf/1301.3781.pdf

Aguirre et al., A study on similarity and relatedness using distributional and word-net based approaches, In Proceedings of NAACL-HLT, 2009