Лекция 6
Word2Vec

Бручес Елена

15.07.2025

Word Embedding

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers.

In linguistics word embeddings were discussed in the research area of distributional semantics. The underlying idea that "*a word is characterized by the company it keeps*" was popularized by Firth.


Обучение
ML моделей
Векторизация
слов

Word Vectors

One-hot vector: Represent every word as an $R^{|V| \times 1}$ vector with all 0s and one 1 at the index of that word in the sorted english language.

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots \; w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

There is no natural notion of similarity in a set of one-hot vectors.

Word-Document Matrix

Loop over billions of documents and for each time word *i* appears in document *j*, we add one to entry X_{ij}. This is obviously a very large matrix ($R^{|V|\times M}$) and it scales with the number of documents (M).

	Word A	Word B	Word C	Word D	Word E
Doc 1	1	0	1	1	1
Doc 2	1	1	0	0	0
Doc 3	0	0	0	1	1

Window based Co-occurrence Matrix

Let our corpus contain just three sentences and the window size be 1:

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

The resulting counts matrix will then be:

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Word2Vec

https://code.google.com/archive/p/word2vec/ - Code

https://arxiv.org/pdf/1301.3781.pdf - Mikolov, Tomas; et al. "Efficient Estimation of Word Representations in Vector Space"

http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/ - Word2Vec Tutorial

Word2Vec

Word2vec is a software package that actually includes:

- 2 algorithms: continuous bag-of-words (CBOW) and skip-gram. CBOW aims to predict a center word from the surrounding context in terms of word vectors. Skip-gram does the opposite, and predicts the distribution (probability) of context words from a center word.
- 2 training methods: negative sampling and hierarchical softmax. Negative sampling defines an objective by sampling negative examples, while hierarchical softmax defines an objective using an efficient tree structure to compute probabilities for all the vocabulary.

Language Models

Unigram model:

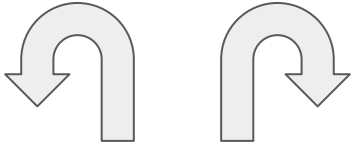
$$P(w_1, w_2, \cdots, w_n) = \prod_{i=1}^{n} P(w_i)$$

Bigram model:

$$P(w_1, w_2, \cdots, w_n) = \prod_{i=2}^{n} P(w_i | w_{i-1})$$

Skip-Gram Model

Predicting surrounding context words given a center word:



The cat jumped over the puddle

Skip-Gram Model

Source Text		Training Samples
The quick brown fox jumps over the lazy dog.	⟹	(the, quick) (the, brown)
The quick brown fox jumps over the lazy dog.	⟹	(quick, the) (quick, brown) (quick, fox)
The quick brown fox jumps over the lazy dog.	⟹	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown fox jumps over the lazy dog.	⟹	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

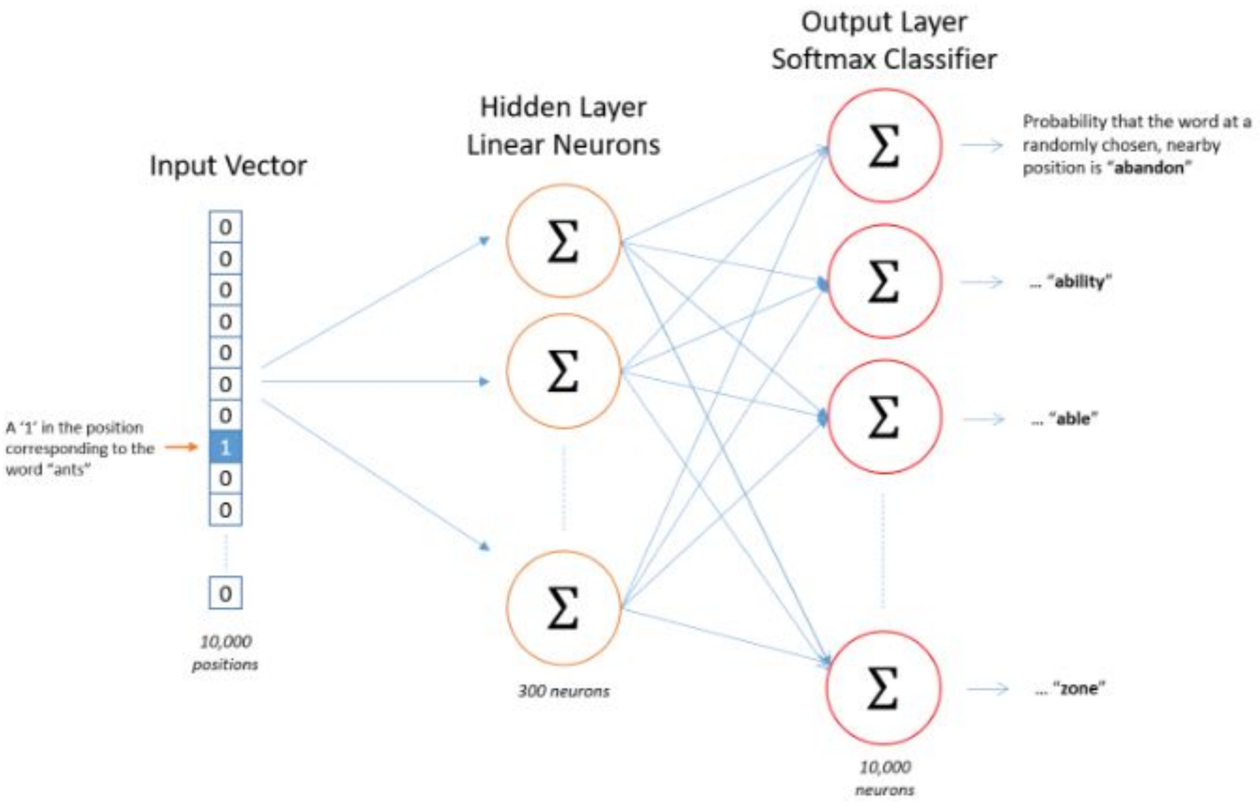
Skip-Gram Model

First build a vocabulary of words from our training documents: let's say we have a vocabulary of 10,000 unique words.

We're going to represent an input word like "*ants*" as a one-hot vector. This vector will have 10,000 components (one for every word in our vocabulary) and we'll place a "1" in the position corresponding to the word "*ants*", and 0s in all of the other positions.

The output of the network is a single vector (also with 10,000 components) containing, for every word in our vocabulary, the probability that a randomly selected nearby word is that vocabulary word.

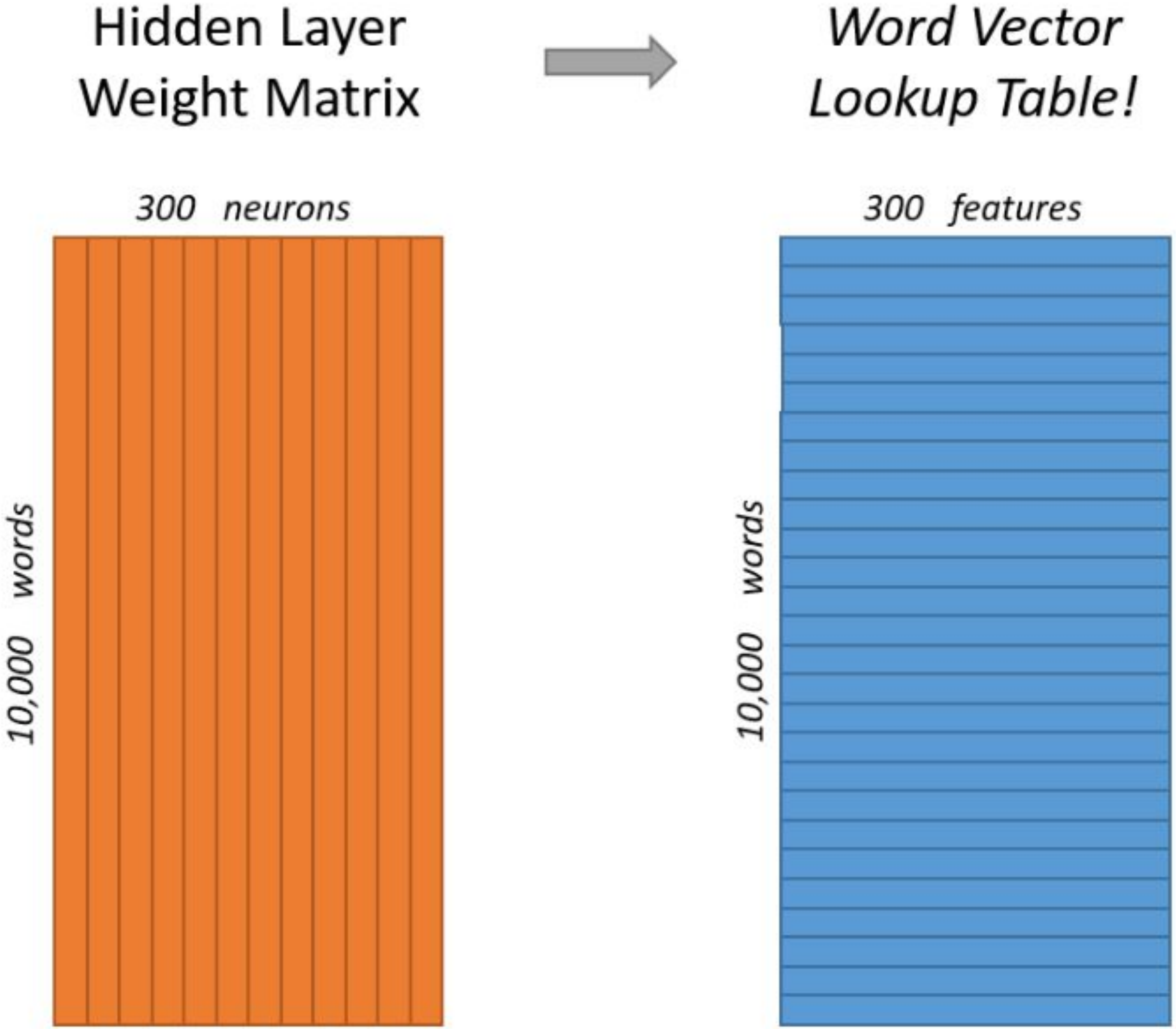
Skip-Gram Model



Skip-Gram Model

For our example, we’re going to say that we’re learning word vectors with 300 features. So the hidden layer is going to be represented by a weight matrix with 10,000 rows (one for every word in our vocabulary) and 300 columns (one for every hidden neuron).

If you look at the rows of this weight matrix, these are actually what will be our word vectors!

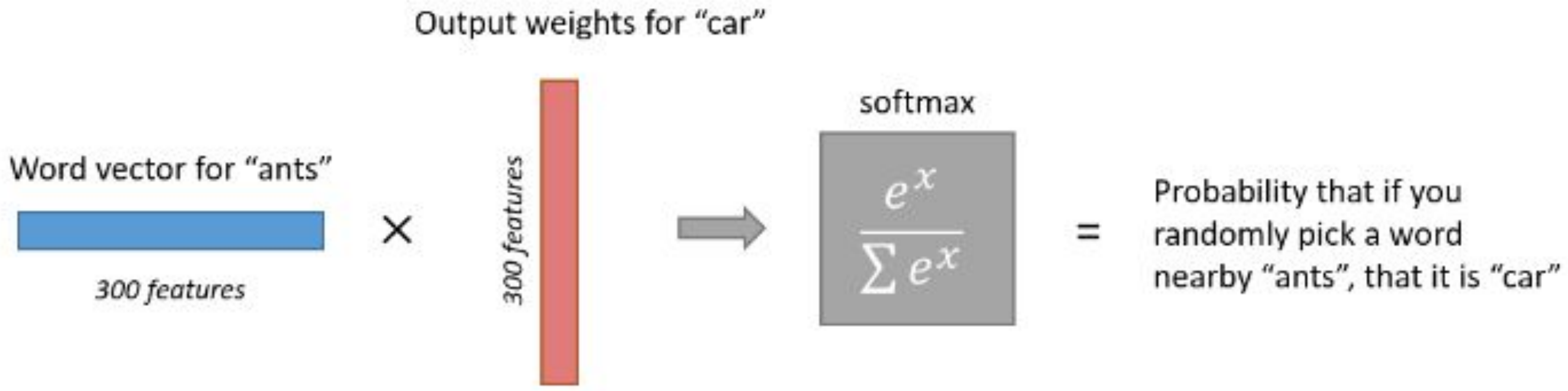

Hidden Layer
Weight Matrix
Word Vector
Lookup Table!
300 neurons
10,000 words
300 features
10,000 words

Skip-Gram Model

The hidden layer of this model is really just operating as a lookup table. The output of the hidden layer is just the “word vector” for the input word.

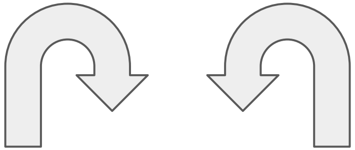
$$[0 \quad 0 \quad 0 \quad \boxed{1} \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ \boxed{10} & \boxed{12} & \boxed{19} \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Skip-Gram Model



Continuous Bag of Words Model (CBOW)

Predicting a center word from the surrounding context:



The cat jumped over the puddle

For each word, we want to learn 2 vectors

- *v*: (input vector) when the word is in the context;
- *u*: (output vector) when the word is in the center

Continuous Bag of Words Model (CBOW)

Notation for CBOW Model:

- w_i : Word i from vocabulary V;
- $V \in \mathbb{R}^{n \times |V|}$: Input word matrix;
- v_i : i-th column of V, the input vector representation of word w_i
- $U \in \mathbb{R}^{|V| \times n}$: Output word matrix;
- u_i : i-th row of U, the output vector representation of word w_i

Softmax Function

$$softmax(\boldsymbol{x})_i = \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}}$$

Continuous Bag of Words Model (CBOW)

1. We generate our one hot word vectors for the input context of size $m : (x^{(c-m)}, \ldots, x^{(c-1)}, x^{(c+1)}, \ldots, x^{(c+m)} \in \mathbb{R}^{|V|})$.

2. We get our embedded word vectors for the context ($v_{c-m} = \mathcal{V}x^{(c-m)}$, $v_{c-m+1} = \mathcal{V}x^{(c-m+1)}$, …, $v_{c+m} = \mathcal{V}x^{(c+m)} \in \mathbb{R}^n$)

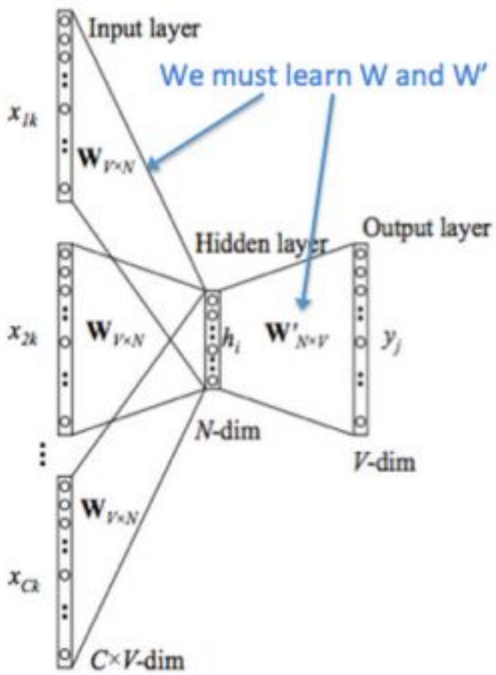
3. Average these vectors to get $\hat{v} = \frac{v_{c-m}+v_{c-m+1}+\ldots+v_{c+m}}{2m} \in \mathbb{R}^n$

4. Generate a score vector $z = \mathcal{U}\hat{v} \in \mathbb{R}^{|V|}$. As the dot product of similar vectors is higher, it will push similar words close to each other in order to achieve a high score.

5. Turn the scores into probabilities $\hat{y} = \text{softmax}(z) \in \mathbb{R}^{|V|}$.

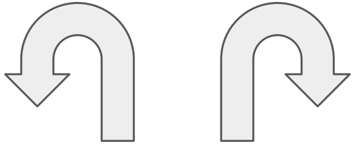
6. We desire our probabilities generated, $\hat{y} \in \mathbb{R}^{|V|}$, to match the true probabilities, $y \in \mathbb{R}^{|V|}$, which also happens to be the one hot vector of the actual word.

Continuous Bag of Words Model (CBOW)



Skip-Gram Model

Predicting surrounding context words given a center word:



The cat jumped over the puddle

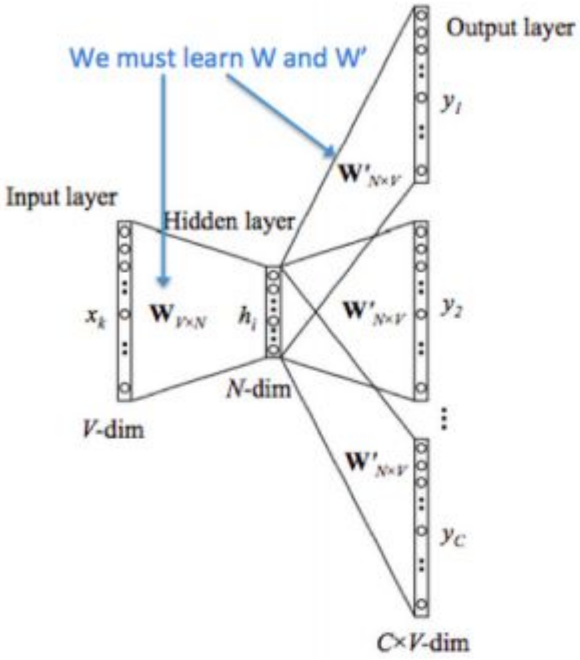
Skip-Gram Model

- w_i : Word i from vocabulary V;
- $V \in R^{n \times |V|}$: Input word matrix;
- v_i : i-th column of V, the input vector representation of word w_i ;
- $U \in R^{n \times |V|}$: Output word matrix;
- u_i : i-th row of U, the output vector representation of word w_i

Skip-Gram Model

1. We generate our one hot input vector $x \in \mathbb{R}^{|V|}$ of the center word.
2. We get our embedded word vector for the center word $v_c = \mathcal{V}x \in \mathbb{R}^n$
3. Generate a score vector $z = \mathcal{U}v_c$.
4. Turn the score vector into probabilities, $\hat{y} = \text{softmax}(z)$. Note that $\hat{y}_{c-m}, \ldots, \hat{y}_{c-1}, \hat{y}_{c+1}, \ldots, \hat{y}_{c+m}$ are the probabilities of observing each context word.
5. We desire our probability vector generated to match the true probabilities which is $y^{(c-m)}, \ldots, y^{(c-1)}, y^{(c+1)}, \ldots, y^{(c+m)}$, the one hot vectors of the actual output.

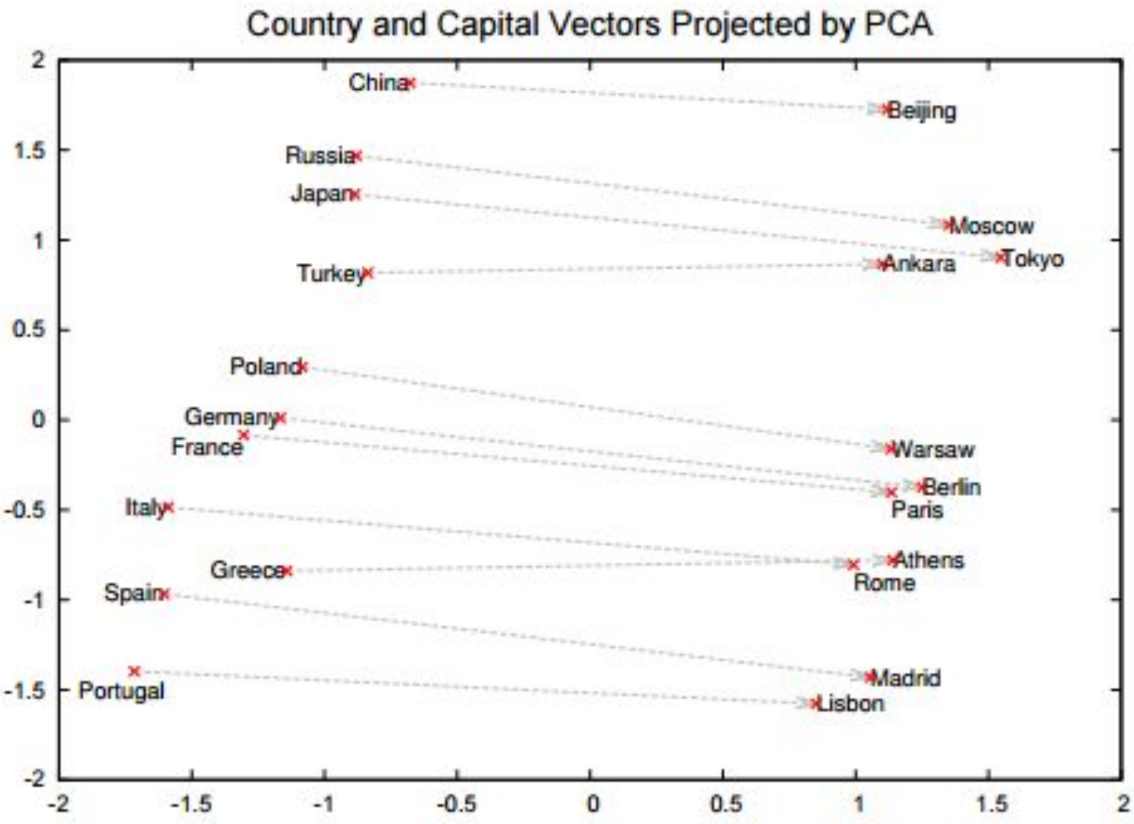
Skip-Gram Model



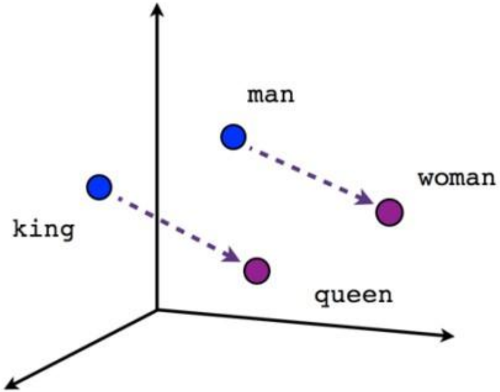
Word2Vec

Hyper-parameters choices:

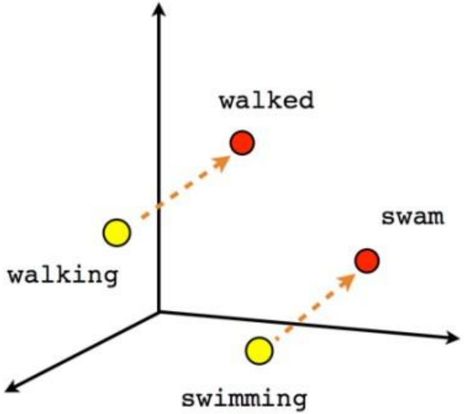
- architecture: skip-gram (slower, better for infrequent words) vs CBOW (fast)
- the training algorithm: hierarchical softmax (better for infrequent words) vs negative sampling (better for frequent words, better with low dimensional vectors)
- dimensionality of the word vectors: usually more is better, but not always
- context (window) size: for skip-gram usually around 10, for CBOW around 5


Country and Capital Vectors Projected by PCA
China
Beijing
Russia
Japan
Moscow
Tokyo
Turkey
Ankara
Poland
Germany
France
Warsaw
Berlin
Paris
Italy
Greece
Spain
Athens
Rome
Madrid
Portugal
Lisbon
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

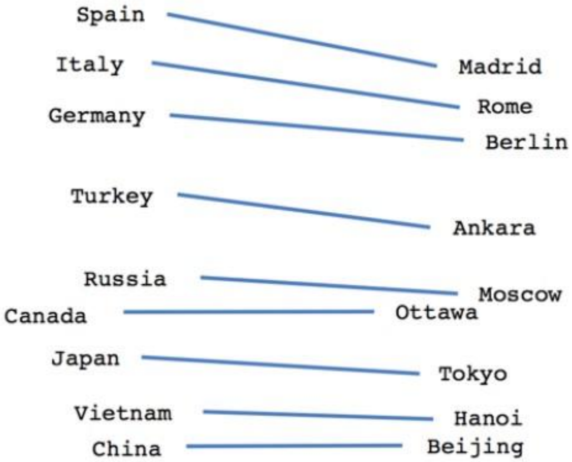
Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - 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



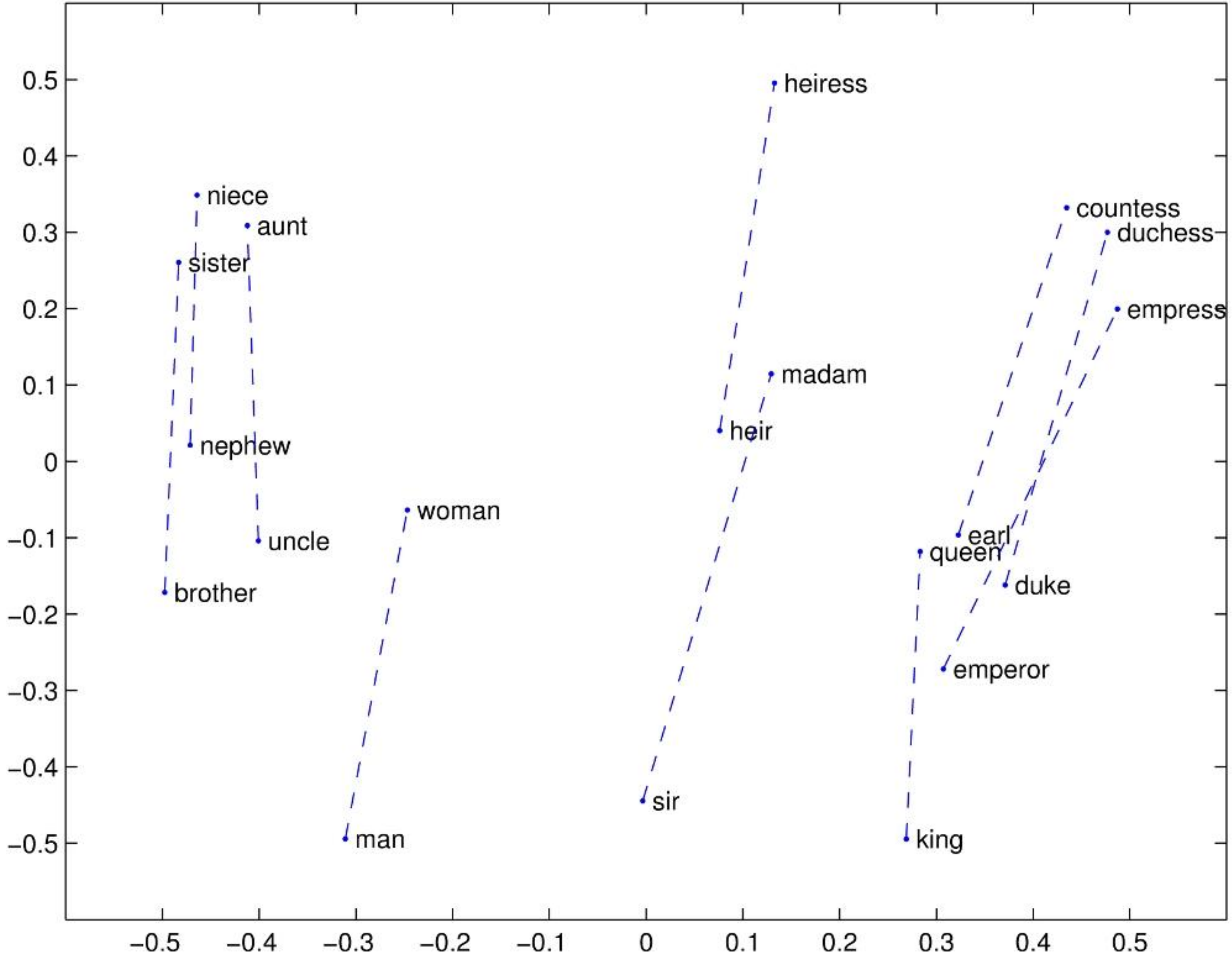
Male-Female



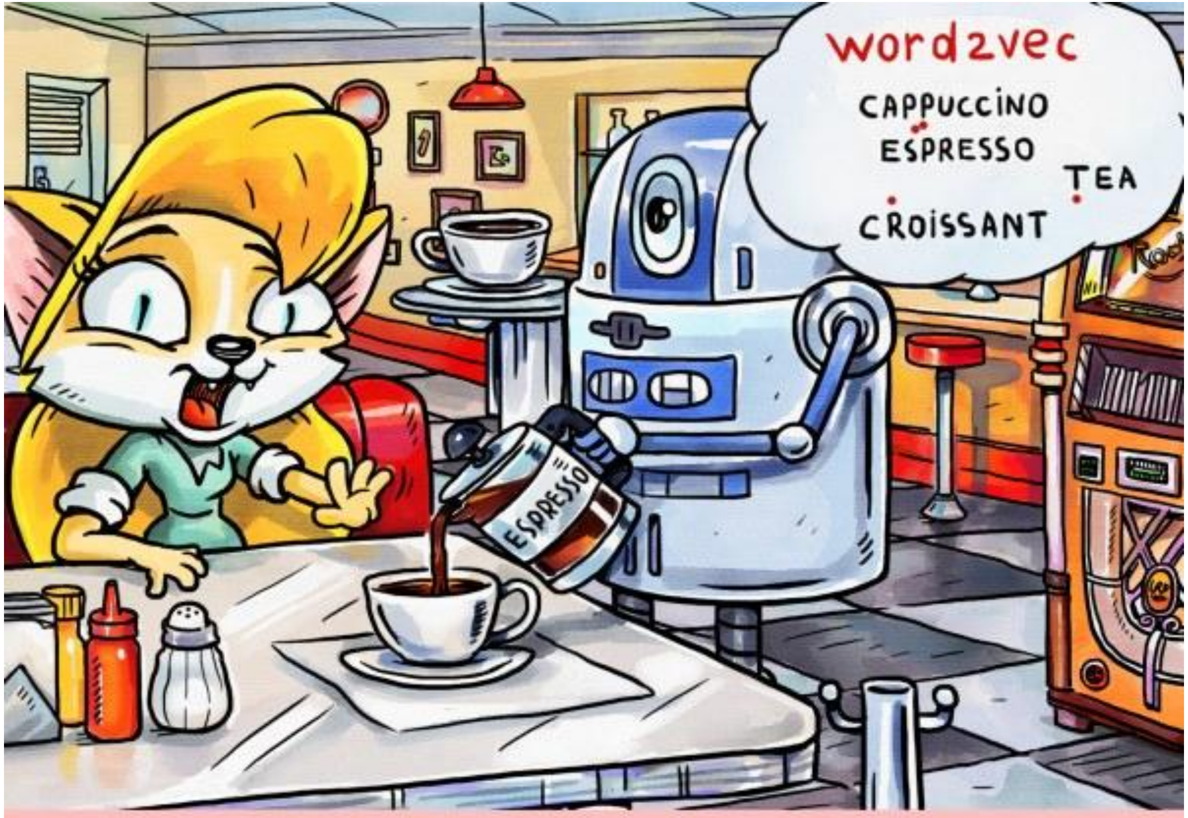
Verb tense



Country-Capital



0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.5
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5
niece
aunt
sister
nephew
uncle
brother
woman
man
heiress
madam
heir
sir
countess
duchess
empress
earl
queen
duke
emperor
king



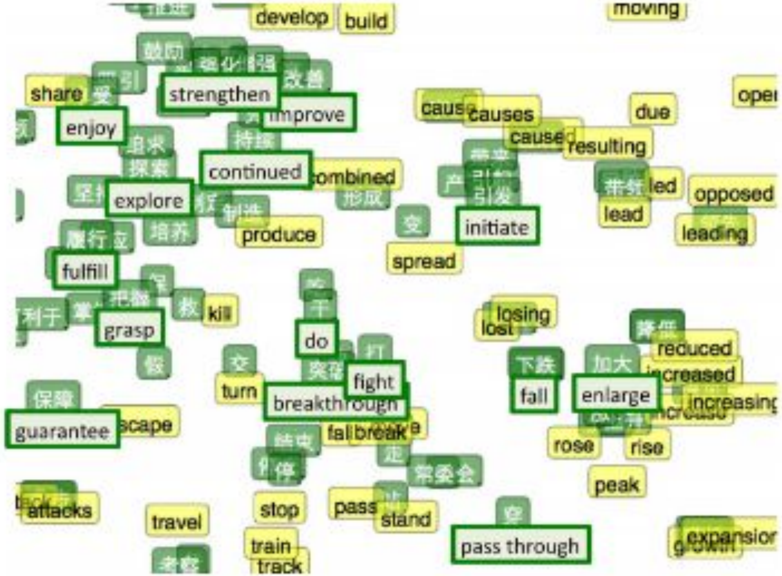
- Espresso? But I ordered a cappuccino!
- Don't worry, the cosine distance between them is so small that they are almost the same thing.

Shared Representations (1)

A bilingual word-embedding, produced in *Socher et al.* : we can learn to embed words from two different languages in a single, shared space. In this case, they learn to embed English and Mandarin Chinese words in the same space.

Of course, the known words had similar meanings end up close together. More interesting is that words they didn't know were translations end up close together.

Shared Representations (1)

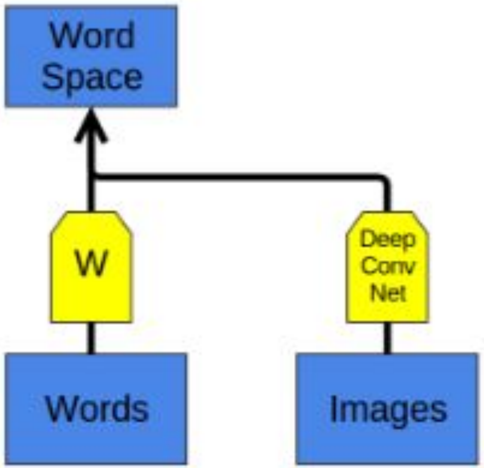


t-SNE visualization of the bilingual word embedding. Green is Chinese, Yellow is English. (Socher *et al.* (2013a))

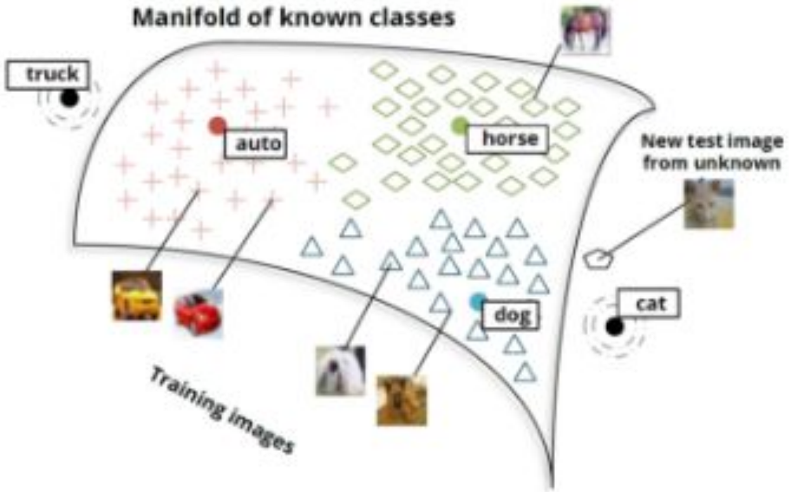
Shared Representation (2)

Recently, deep learning has begun exploring models that embed images and words in a single representation.

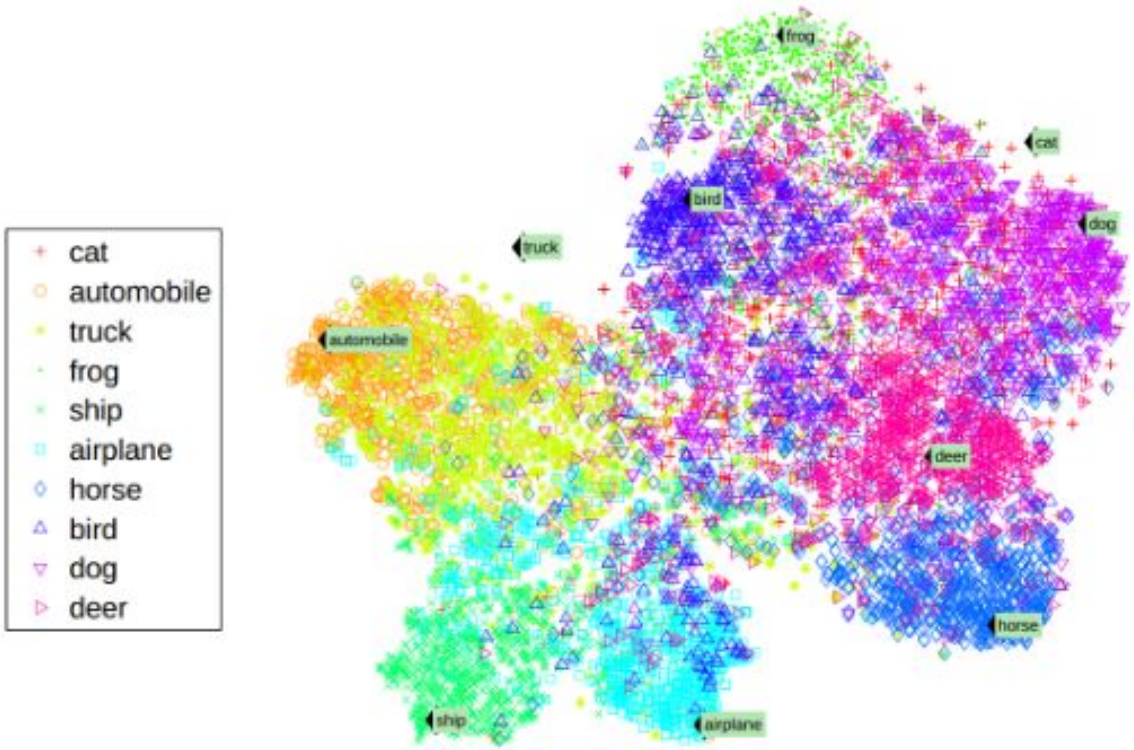
The basic idea is that one classifies images by outputting a vector in a word embedding. E.g., images of dogs are mapped near the “dog” word vector.



Shared Representation (2)



Shared Representation (2)



Word Embeddings

Types

Word Embedding Types (1)

- **Substitute-based Word Embeddings** (*Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. 2012. Learning syntactic categories using paradigmatic representations of word context. In Proceedings of EMNLP*)
- **Window-based Word Embeddings** (Word2Vec) (*Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In Proceedings of ICLR.*)
- **Dependency-based Word Embeddings** (*Levy, O., & Goldberg, Y. (2014). Dependency-Based Word Embeddings. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), 302–308.*)
- **Glove** (*Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014), volume 12.*)
- **Structured Word2Vec** (position-aware contexts) (*Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. Two/too simple adaptations of word2vec for syntax problems. In HLT-NAACL, pp. 1299–1304, 2015.*)

Word Embedding Types (2)

- **Attention-Based Continuous Bag-of-words** (*Wang Ling, Lin Chu-Cheng, Yulia Tsvetkov, et al. Not All Contexts Are Created Equal: Better Word Representations with Variable Attention, 2015. In Proceedings of EMNLP.*)
- **Meta-Embeddings** (*Wenpeng Yin and Hinrich Schutze. 2016. Learning word meta-embeddings. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1351–1360, Berlin, Germany. Association for Computational Linguistics.*)
- **Subword-level embeddings** (*Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics.*)

Qualitative effect of context type (1) [1]

- In embeddings learned with **large window contexts** there are both functionally similar words and topically similar words, sometimes with a different part-of-speech;
- With **small windows** and **dependency contexts**, we see much fewer topically similar words;
- With **substitute-based contexts**, there appears to be even a stronger preference for functional similarity, with a tendency to also strictly preserve verb tense.

$W10^{300}$	DEP^{300}	SUB^{300}
played	play	singing
play	played	rehearsing
plays	understudying	performing
professionally	caddying	composing
player	plays	running

Table 1: The top five words closest to target word *playing* in different embedding spaces.

Qualitative effect of context type (2) [2]

- Bound context representation plays an important role, especially for CBOW.
- Bound context representation is suitable for sequence labeling task, especially when it is used along with dependency-based context.
- On text classification task, different contexts do not affect the final performance much. Nonetheless, the use of pre-trained word embeddings is crucial and linear context type with unbound representation (Skip-Gram) is still the best choice.
- The overall tendency of models with different contexts is similar, especially for Skip-Gram and GloVe. GloVe is more sensitive to different contexts than Skip-Gram and CBOW. CBOW benefits most from linear context type.

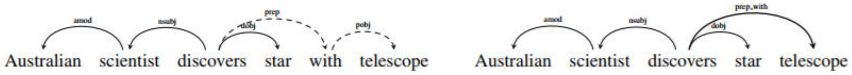


Figure 1: Illustration of dependency parse tree for sentence "Australian scientist discovers star with telescope". Note that preposition relation is collapsed in the right sub-figure, where *telescope* is considered as a direct modifier of *discovers*.

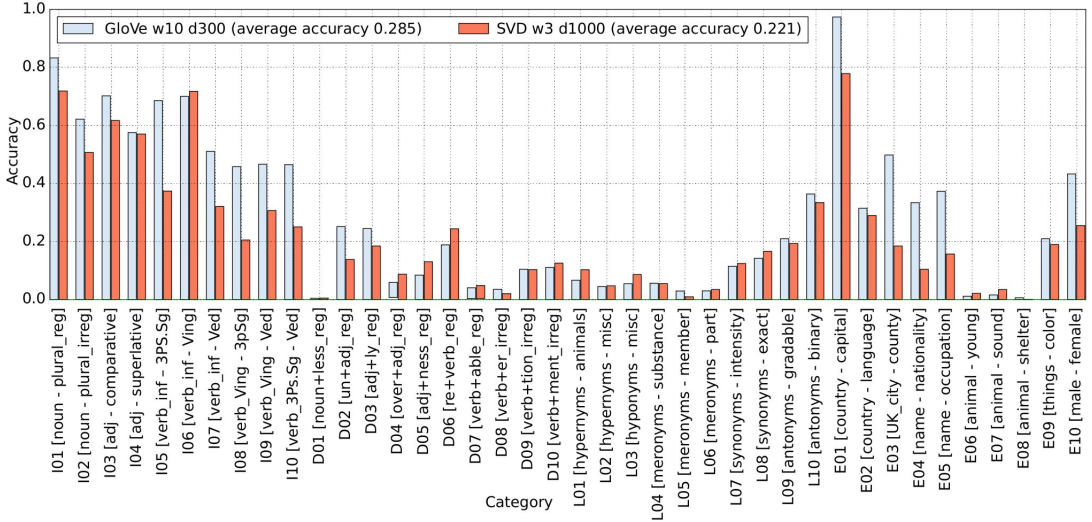
Table 2: Illustration of bound and unbound context representations under linear and dependency-based context types. This example is based on Figure 1 and the target word is "discovers".

context type / context representation	linear	dependency-based
unbound	australian, scientist, star, with	scientist, star, telescope
bound	australian/-2, scientist/-1, star/+1, with/+2	scientist/nsubj, star/dobj, telescope/prep_with

Qualitative effect of context type (3) [3]

- There does not seem to be a correlation between type of linguistic relation and preference for higher or low dimensionality;
- Obtained data does not confirm the intuition about larger windows being more beneficial for semantic relations, and smaller windows - for morphological

*Release The **B**igger **A**nalogy **T**est **S**et!*



Qualitative effect of context type (4) [4]

- The results are heavily dependent on the actual language: the claims made for English do not extend to other languages;
- Different context types yield semantic spaces with different properties, and that the optimal context type depends on the actual application and language;
- DEPS perform dramatically worse than BOW contexts on analogy tasks;
- UDEPS-ARC in German outperforms all other context types on all syntactic analogies, except for the nationality-adjective relation;
- POSIT displays a strong performance in detecting functional similarity across all three languages (English, Italian, German) in both tasks;
- The usefulness of universal dependency-based contexts is evident with a simple post-parsing context extraction scheme in tasks oriented towards syntactic/functional similarity

Tricks: the order does matter

Tsvetkov, Y., Faruqui, M., Ling, W., and Dyer, C. (2016b). Learning the curriculum with Bayesian optimization for task-specific word representation learning. In Proc. ACL.
https://arxiv.org/pdf/1605.03852.pdf

		Senti	**NER**	**POS**	**Parse**
Shuffled	median	66.01	85.88	96.35	75.08
	best	66.61	85.50	96.38	76.40
Sorted	long→short	66.78	85.22	96.47	75.85
	short→long	66.12	85.49	96.20	75.31
Coherent	original order	66.23	85.99	96.47	76.08
Optimized curriculum	diversity	66.06	86.09	96.59	**76.63**
	prototypicality	**67.44**	85.96	96.53	75.81
	simplicity	67.11	**86.42**	**96.62**	76.54

Table 2: Evaluation of the impact of the curriculum of word embeddings on the downstream tasks.

Limitations of Word Embeddings [5]

- Word embeddings do not capture semantic relations such as hyponymy and entailment (*the principle that under certain conditions the truth of one statement ensures the truth of a second statement*);
- While state-of-the-art embeddings are successful at capturing taxonomic information (e.g., *cow* is an animal), they are much less successful in capturing attributive properties (*bananas* are yellow);
- Word embeddings are unable to distinguish between pairs of words with opposite meanings (antonyms, e.g., *good*/*bad*).

References

1. Melamud, O., McClosky, D., Patwardhan, S., & Bansal, M. (2016). The Role of Context Types and Dimensionality in Learning Word Embeddings. In Proceedings of NAACL-HLT 2016 (pp. 1030–1040). https://arxiv.org/abs/1601.00893
2. B Li, T Liu, Z Zhao, B Tang, A Drozd, A Rogers, X Du. Investigating Different Syntactic Context Types and Context Representations for Learning Word Embeddings. 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, September 7–11, 2017 (pp. 2411–2421) https://openreview.net/pdf?id=Bkfwyw5xg
3. Gladkova, A., Drozd, A., & Matsuoka, S. (2016). Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In Proceedings of NAACL-HLT 2016 (SRW). http://www.aclweb.org/anthology/N16-2002
4. Ivan Vulic and Anna Korhonen. 2016. Is "universal ´ syntax" universally useful for learning distributed word representations? In ACL. http://anthology.aclweb.org/P16-2084
5. Schwartz R., Reichart R., Rappoport A. Symmetric Patterns and Coordinations: Fast and Enhanced Representations of Verbs and Adjectives //HLT-NAACL. – 2016. – C. 499-505. https://pdfs.semanticscholar.org/62c4/fc68d1dcaf99a6c0b4e7f657861ae573c060.pdf

Word Embeddings

Measures, trends etc.

Word embeddings in 2017: Trends and future directions
http://ruder.io/word-embeddings-2017/

- Subword-level embeddings
- OOV handling
- Evaluation
- Multi-sense embeddings
- Beyond words as points
- Phrases and multi-word expressions
- Bias
- Temporal dimension
- Lack of theoretical understanding
- Task and domain-specific embeddings
- Embeddings for multiple languages
- Embeddings based on other contexts

Schnabel T., Labutov I., Mimno D., Joachims Th. *Evaluation methods for unsupervised word embeddings* // Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 298–307, Lisbon, Portugal, 17-21 September 2015. http://www.aclweb.org/anthology/D15-1036

Evaluation

1. **Extrinsic evaluation**: using word embeddings as input features to a downstream task and measuring changes in performance metrics specific to that task (e.g. POS-tagging, NER);
2. **Intrinsic evaluation**: testing directly for syntactic or semantic relationships between words

Intrinsic evaluation

- **Relatedness**: the cosine similarity of the embeddings for two words should have high correlation (Spearman or Pearson) with human relatedness scores;
- **Analogy**: the goal is to find a term x for a given term y so that $x : y$ best resembles a sample relationship $a : b$;
- **Categorization**: the goal is to recover a clustering of words into different categories;
- **Selectional preference**: the goal is to determine how typical a noun is for a verb either as a subject or as an object (e.g., *people eat*, but we rarely *eat people*).

	relatedness						categorization			sel. prefs		analogy			
	rg	ws	wss	wsr	men	toefl	ap	esslli	batt.	up	mcrae	an	ansyn	ansem	average
CBOW	**74.0**	**64.0**	**71.5**	**56.5**	**70.7**	66.7	**65.9**	**70.5**	**85.2**	24.1	13.9	**52.2**	**47.8**	**57.6**	**58.6**
GloVe	63.7	54.8	65.8	49.6	64.6	**69.4**	64.1	65.9	77.8	27.0	**18.4**	42.2	44.2	39.7	53.4
TSCCA	57.8	54.4	64.7	43.3	56.7	58.3	57.5	**70.5**	64.2	**31.0**	14.4	15.5	19.0	11.1	44.2
C&W	48.1	49.8	60.7	40.1	57.5	66.7	60.6	61.4	80.2	28.3	16.0	10.9	12.2	9.3	43.0
H-PCA	19.8	32.9	43.6	15.1	21.3	54.2	34.1	50.0	42.0	-2.5	3.2	3.0	2.4	3.7	23.1
Rand. Proj.	17.1	19.5	24.9	16.1	11.3	51.4	21.9	38.6	29.6	-8.5	1.2	1.0	0.3	1.9	16.2

Table 1: Results on absolute intrinsic evaluation. The best result for each dataset is highlighted in bold. The second row contains the names of the corresponding datasets.

Results on extrinsic evaluation

	dev	test	p-value
Baseline	94.18	93.78	0.000
Rand. Proj.	94.33	93.90	0.006
GloVe	94.28	93.93	0.015
H-PCA	94.48	93.96	0.029
C&W	**94.53**	**94.12**	
CBOW	94.32	93.93	0.012
TSCCA	**94.53**	94.09	0.357

Table 4: F1 chunking results using different word embeddings as features. The p-values are with respect to the best performing method.

	test	p-value
BOW (baseline)	88.90	$7.45 \cdot 10^{-14}$
Rand. Proj.	62.95	$7.47 \cdot 10^{-12}$
GloVe	74.87	$5.00 \cdot 10^{-2}$
H-PCA	69.45	$6.06 \cdot 10^{-11}$
C&W	72.37	$1.29 \cdot 10^{-7}$
CBOW	**75.78**	
TSCCA	75.02	$7.28 \cdot 10^{-4}$

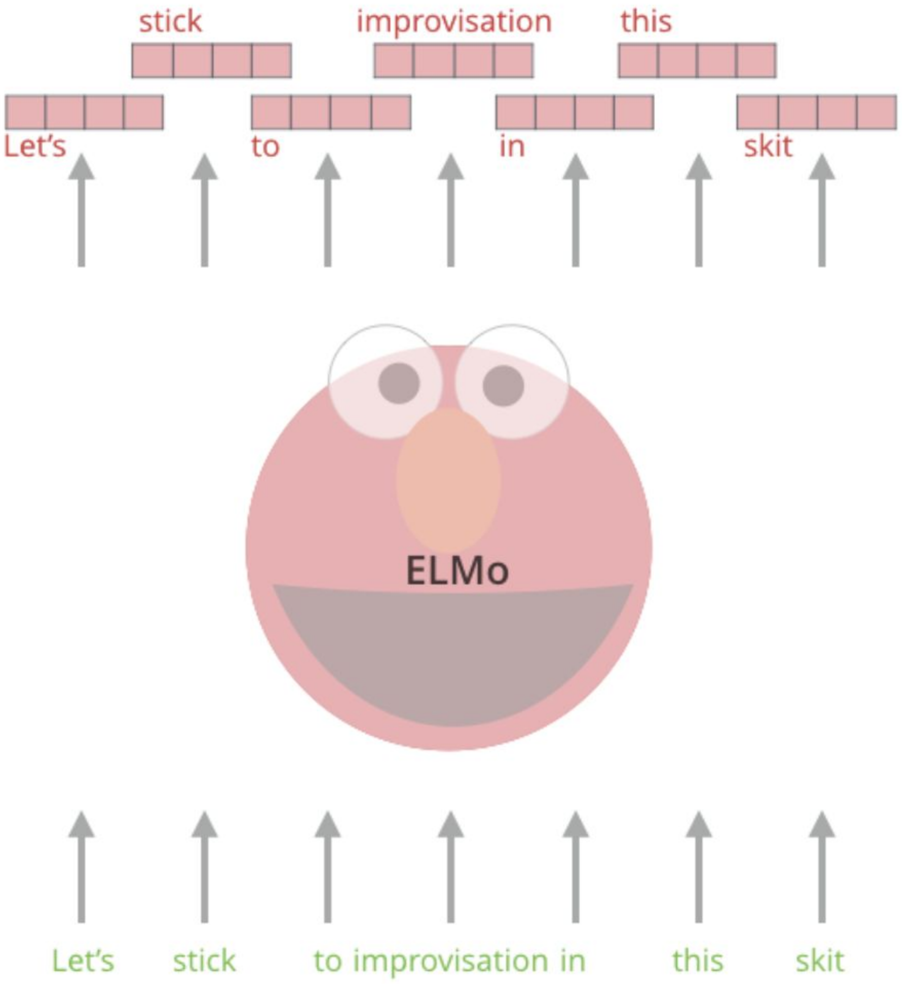
Table 5: F1 sentiment analysis results using different word embeddings as features. The p-values are with respect to the best performing embedding.

Nayak, Neha & Angeli, Gabor & Manning, Christoper. (2016). *Evaluating Word Embeddings Using a Representative Suite of Practical Tasks*. 19-23. http://www.aclweb.org/anthology/W16-2504

Contextualized word embeddings

ELMo: Context Matters

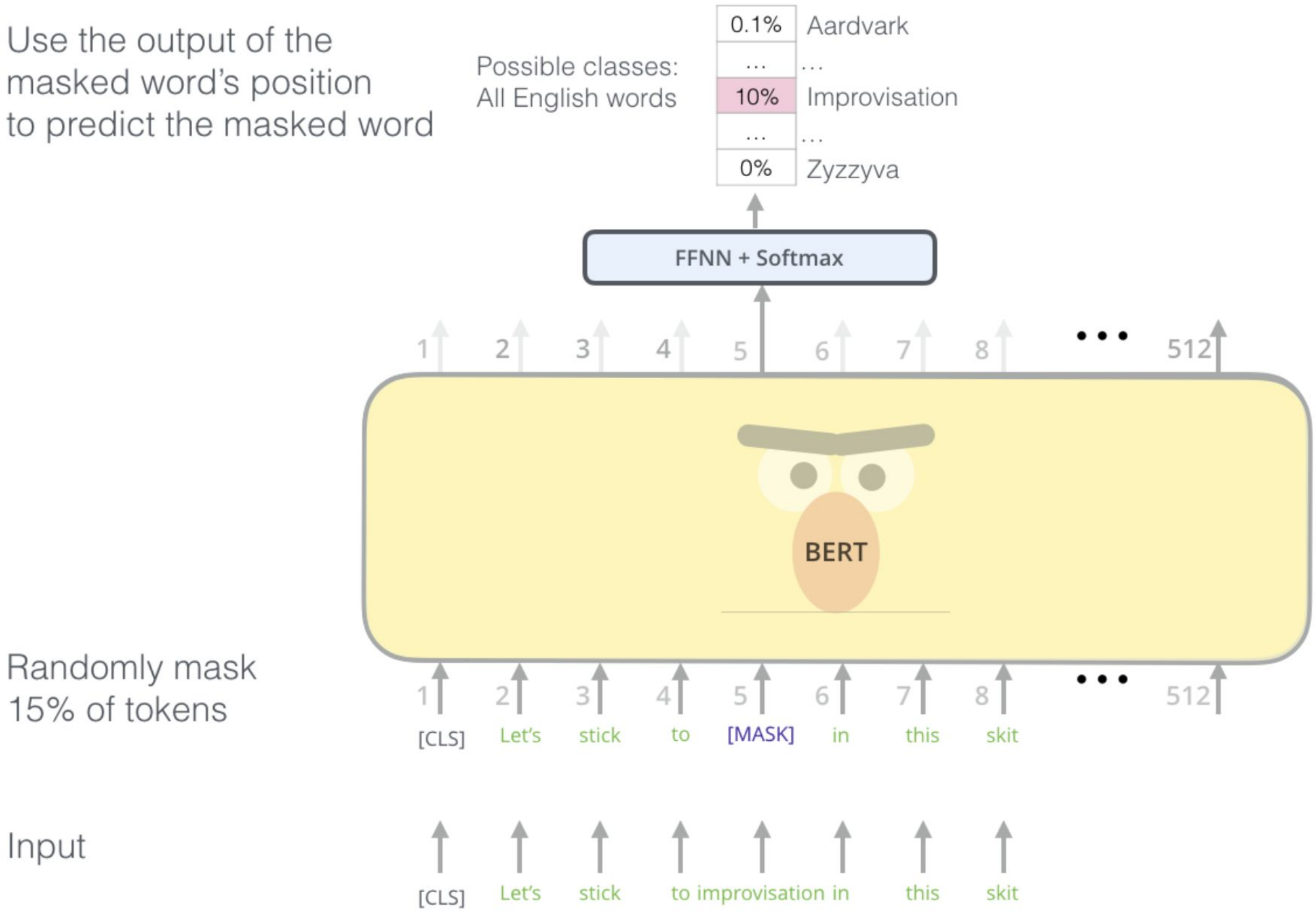
ELMo
Embeddings

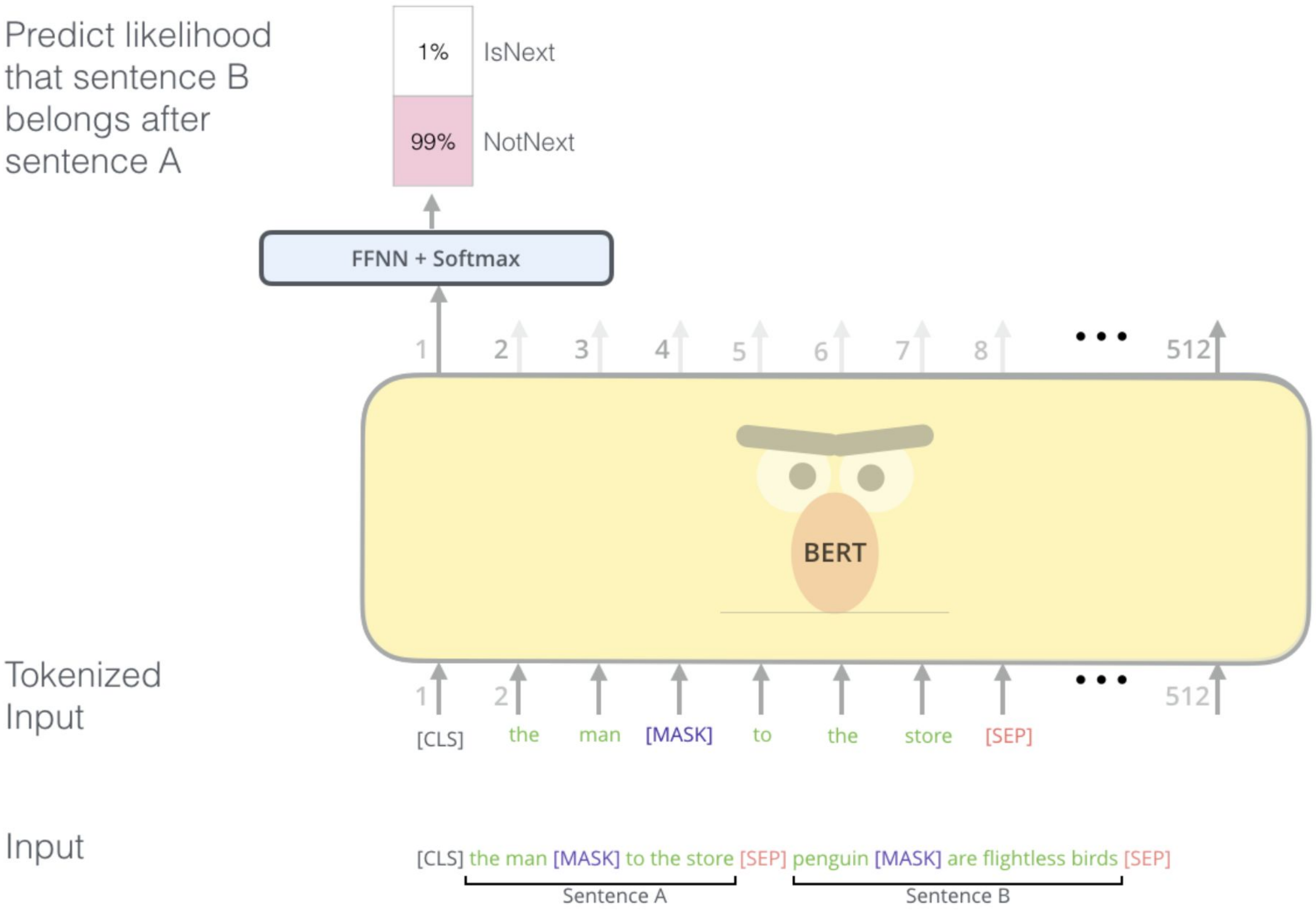


Words to embed

ELMo: Metrics

Task	Previous SOTA		Our baseline	ELMo + Baseline	Increase (Absolute/Relative)
SQuAD	SAN	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al (2017)	88.6	88.0	88.7 +/- 0.17	0.7 / 5.8%
SRL	He et al (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al (2017)	91.93 +/- 0.19	90.15	92.22 +/- 0.10	2.06 / 21%
Sentiment (5-class)	McCann et al (2017)	53.7	51.4	54.7 +/- 0.5	3.3 / 6.8%


Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
...
10% Improvisation
...
0% Zyzzyva
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
Randomly mask 15% of tokens
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit


Predict likelihood that sentence B belongs after sentence A
1%
IsNext
99%
NotNext
FFNN + Softmax
1
2
3
4
5
6
7
8
• • •
512
BERT
Tokenized Input
1
2
• • •
512
[CLS]
the
man
[MASK]
to
the
store
[SEP]
Input
[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flightless birds [SEP]
Sentence A
Sentence B