Sentiment analysis

WORD EMBEDDINGS

Nguyen Ngoc Thao
nnthao@fit.hcmus.edu.vn

Content outline

- Word embeddings: An introduction
- Conventional text encoding
- Modern word embeddings

Word embeddings

- A word embedding is a learned text representation where terms of the same meaning have an equal representation.
  - It can capture the context of a word in a document, semantic and syntactic similarity, relation with other words, etc.
- It is typically in the form of a real-valued vector that encodes the meaning of the word.

Word embeddings

- **Key idea:** The embeddings of similar words are closer in the vector space than those of different ones.

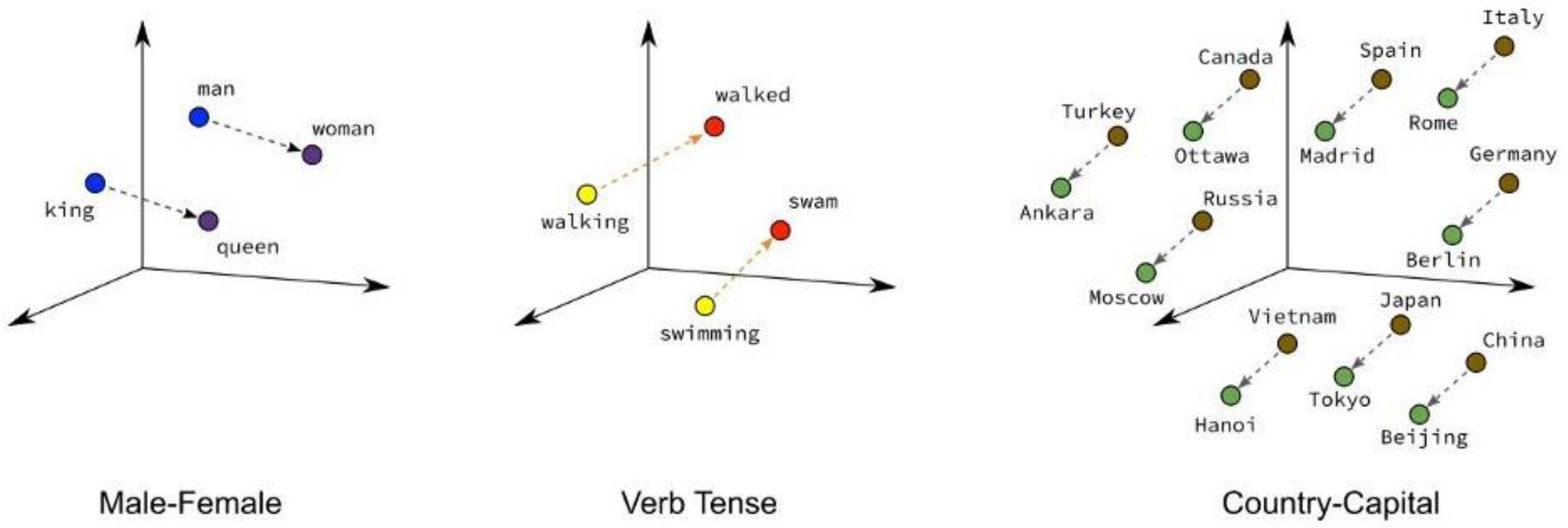


Image credit: Google Developer

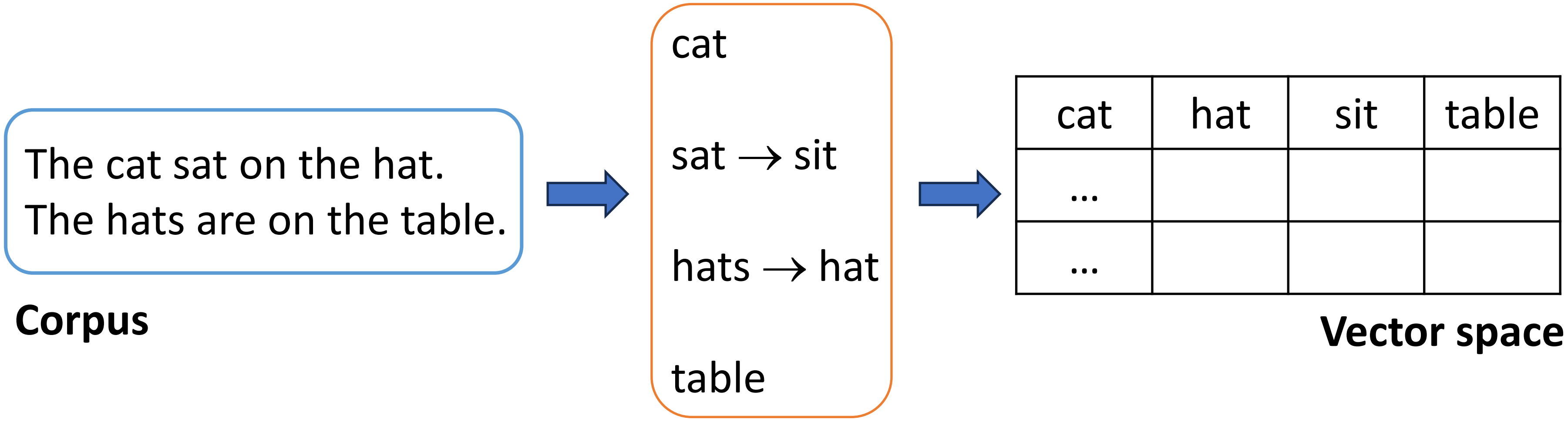
Word embeddings: Applications

- Similarity analysis
- Word clustering
- Ambiguity resolution & Paraphrasing
- Information retrieval & Data mining
- Topic classification
- Sentiment analysis

Conventional text encodings

Document vectorization

- A vocabulary is built from all words available in the corpus.
- Each word is a column in the vector space.



The cat sat on the hat.
The hats are on the table.

Corpus

cat

sat → sit

hats → hat

table

cat	hat	sit	table
…			
…			

Vector space

Text processing
- Tokenize
- Remove stop words
- Lemmatize

One-hot encoding for text

- Each term is encoded by a one-hot vector, in which
  - The vector's length is the number of distinct terms in the dataset
  - Only the element at the corresponding index is set to one.
- A tensor of multiple one-hot vectors describes a document.

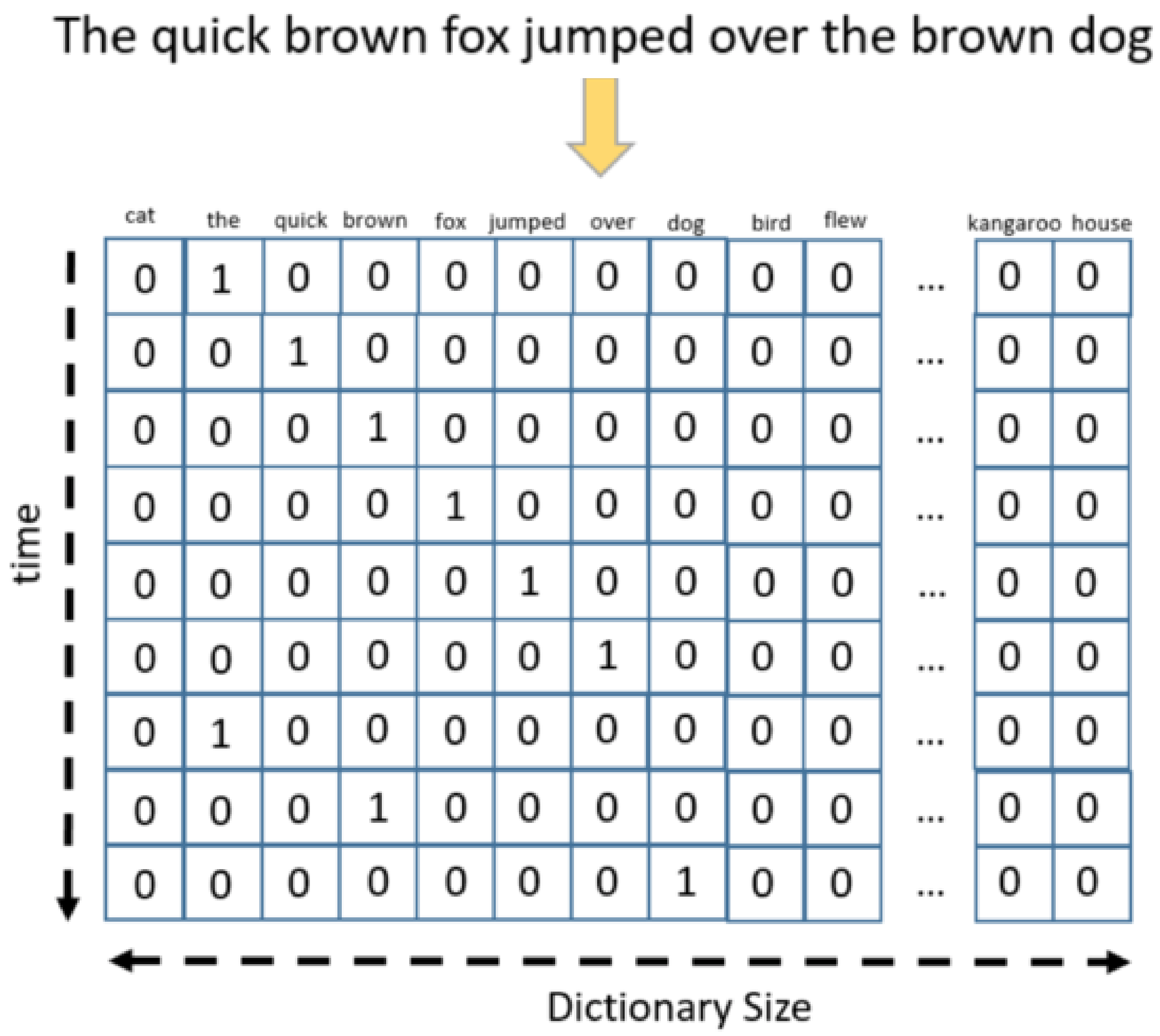
Gain

- Terms are ordered $\rightarrow$ suitable for models like RNN,…

Loss

- The vector is huge due to large real-world vocabulary.
- Binary representation $\rightarrow$ the frequency of words is lost.

One-hot encoding: An example


The quick brown fox jumped over the brown dog

cat	the	quick	brown	fox	jumped	over	dog	bird	flew		kangaroo	house
0	1	0	0	0	0	0	0	0	0	...	0	0
0	0	1	0	0	0	0	0	0	0	...	0	0
0	0	0	1	0	0	0	0	0	0	...	0	0
0	0	0	0	1	0	0	0	0	0	...	0	0
0	0	0	0	0	1	0	0	0	0	...	0	0
0	0	0	0	0	0	1	0	0	0	...	0	0
0	1	0	0	0	0	0	0	0	0	...	0	0
0	0	0	1	0	0	0	0	0	0	...	0	0
0	0	0	0	0	0	0	1	0	0	...	0	0

time
Dictionary Size

Image credit: Data Science Central

Binary encoding for text

- Each document is encoded by a binary vector, in which
  - The vector's length is the number of distinct terms in the dataset
  - If a term is present, the element at the corresponding index is set to one. Otherwise, set it to zero.

Gain

- Tensor is unnecessary, yet the vector is **still** huge.

Loss

- Binary representation $\rightarrow$ the frequency of words is lost.
- Terms are unordered $\rightarrow$ the context of words are lost.

Binary encoding: An example

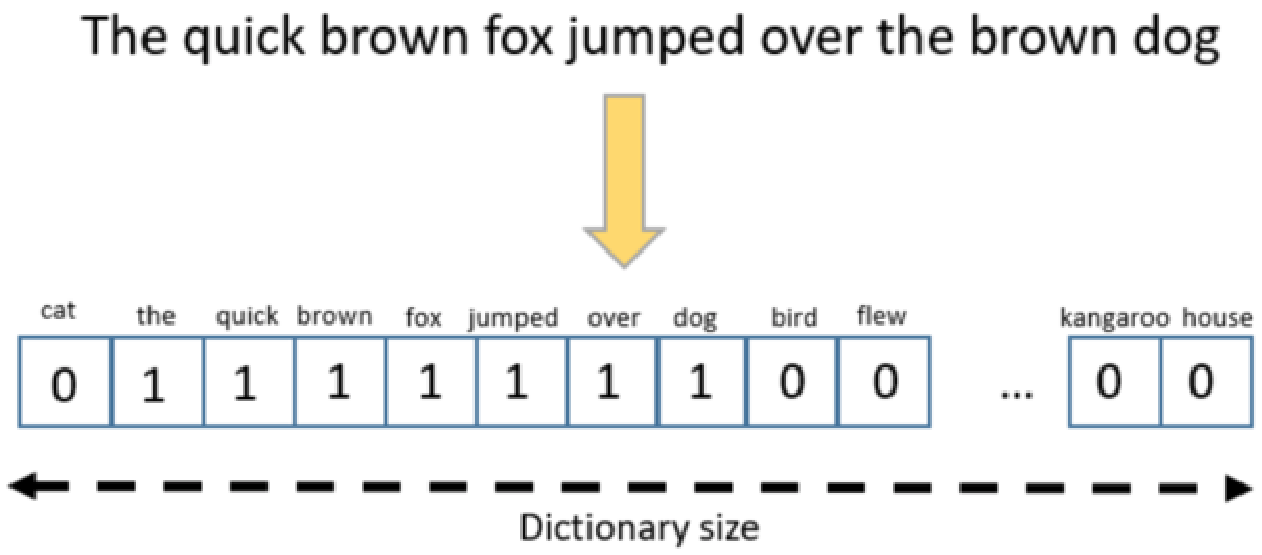


Image credit: Data Science Central

Index-based encoding for text

- A dictionary is required to map words to (numeral) indexes.
- We represent each document via a sequence of indices, each encodes one word.

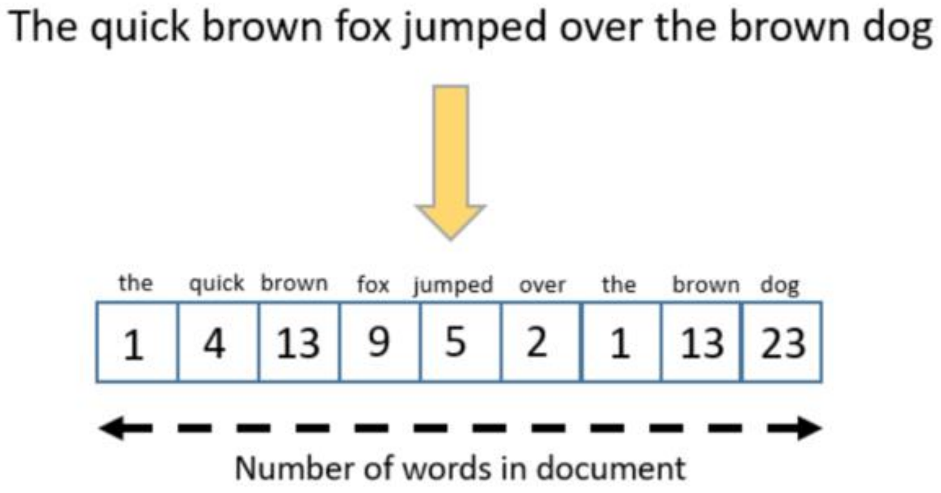

The quick brown fox jumped over the brown dog
the
quick
brown
fox
jumped
over
the
brown
dog
1
4
13
9
5
2
1
13
23
Number of words in document

Image credit:
Data Science Central

Loss

- It introduces a numerical distance between texts that does not really exist.

Term frequency (TF)

- Let the raw count $f_{t,d}$ of a term t in document d be the number of times that t appears in d.
- The term frequency $\boldsymbol{tf}(\boldsymbol{t},\boldsymbol{d})$ of a term t in doc d is given by

$$\boldsymbol{tf}(\boldsymbol{t},\boldsymbol{d}) = \frac{\boldsymbol{f}_{\boldsymbol{t},\boldsymbol{d}}}{\sum_{\boldsymbol{t}' \in \boldsymbol{d}} \boldsymbol{f}_{\boldsymbol{t}',\boldsymbol{d}}}$$

  - t' represents every distinct term in the document d.

Term frequency (TF)

Gain

- The frequency of words can be captured.

Loss

- A term that appears in many documents of the corpus may not be discriminative.
- Terms are unordered $\rightarrow$ the context of words are lost.

Term frequency (TF): Example

- Consider a data corpus of four documents
  - Doc 1: fast car highway road car
  - Doc 2: car car bike fast fast
  - Doc 3: road road highway fast wheel
  - Doc 4: bike wheel car wheel
- The following table shows the term frequencies.

	bike	car	fast	highway	road	wheel
Doc 1	0	2/5	1/5	1/5	1/5	0
Doc 2	1/5	2/5	2/5	0	0	0
Doc 3	0	0	1/5	1/5	2/5	1/5
Doc 4	1/4	1/4	0	0	0	2/4

Term frequency (TF): Variants

- There are various other ways to define the term frequency.

Weighting scheme	tf weight
Binary	0, 1
Raw count	$f_{t,d}$
Log normalization	$\log\left(1 + f_{t,d}\right)$
Double normalization 0.5	$0.5 + 0.5 \frac{f_{t,d}}{\max_{t' \in d} f_{t',d}}$
Double normalization K $K \in [0,1]$	$K + (1 - K) \frac{f_{t,d}}{\max_{t' \in d} f_{t',d}}$

Inverse document frequency (IDF)

- The inverse document frequency $\boldsymbol{idf(t,D)}$ of a term t in the document collection D is given by

$$\boldsymbol{idf(t,D) = \log \frac{N}{|\{d \in D: t \in d\}|}}$$

  - N is the number of document in the corpus D
  - The term is not in the corpus $\rightarrow$ division-by-zero $\rightarrow$ adjust the denominator to $1 + |\{\boldsymbol{d} \in \boldsymbol{D}: \boldsymbol{t} \in \boldsymbol{d}\}|$.

- This solves the issue of terms that are too frequent across the documents.

IDF: Example

- Consider a data corpus of four documents
  - Doc 1: fast car highway road car
  - Doc 2: car car bike fast fast
  - Doc 3: road road highway fast wheel
  - Doc 4: bike wheel car wheel
- The following table shows the inverse document frequencies.

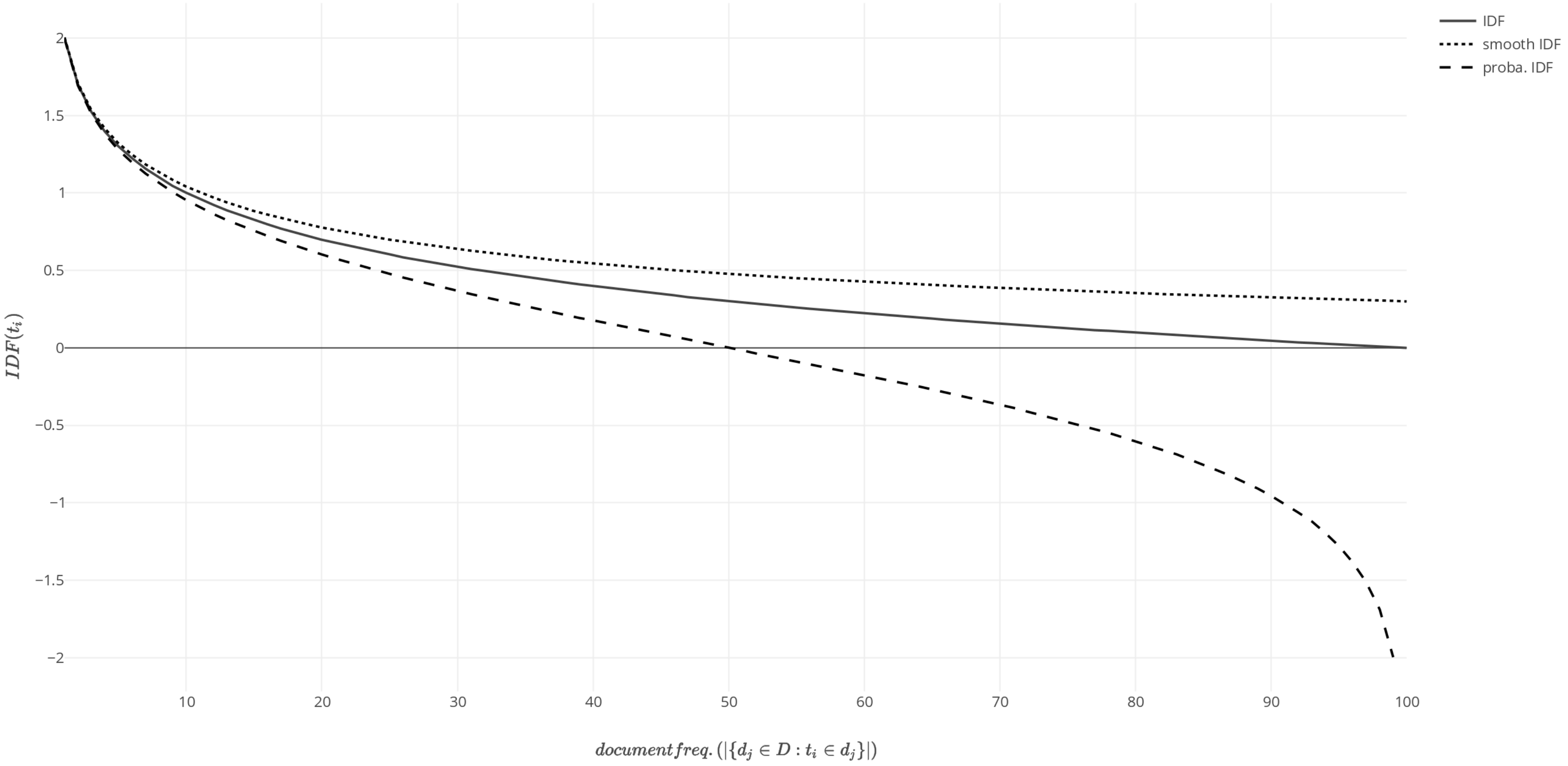
	bike	car	fast	highway	road	wheel
idf(t, D)	log(4/2)	log(4/3)	log(4/3)	log(4/2)	log(4/2)	log(4/2)

IDF: Variants

- There are also many ways to define the inverse document frequency.

Weighting scheme	idf weight ($n_t = \|\{d \in D: t \in d\}\|$)
Unary	1
Inverse document frequency smooth	$\log\left(\frac{N}{1+n_t}\right)+1$
Inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1+n_t}\right)$
Probabilistic inverse document frequency	$\log\left(\frac{N-n_t}{n_t}\right)$

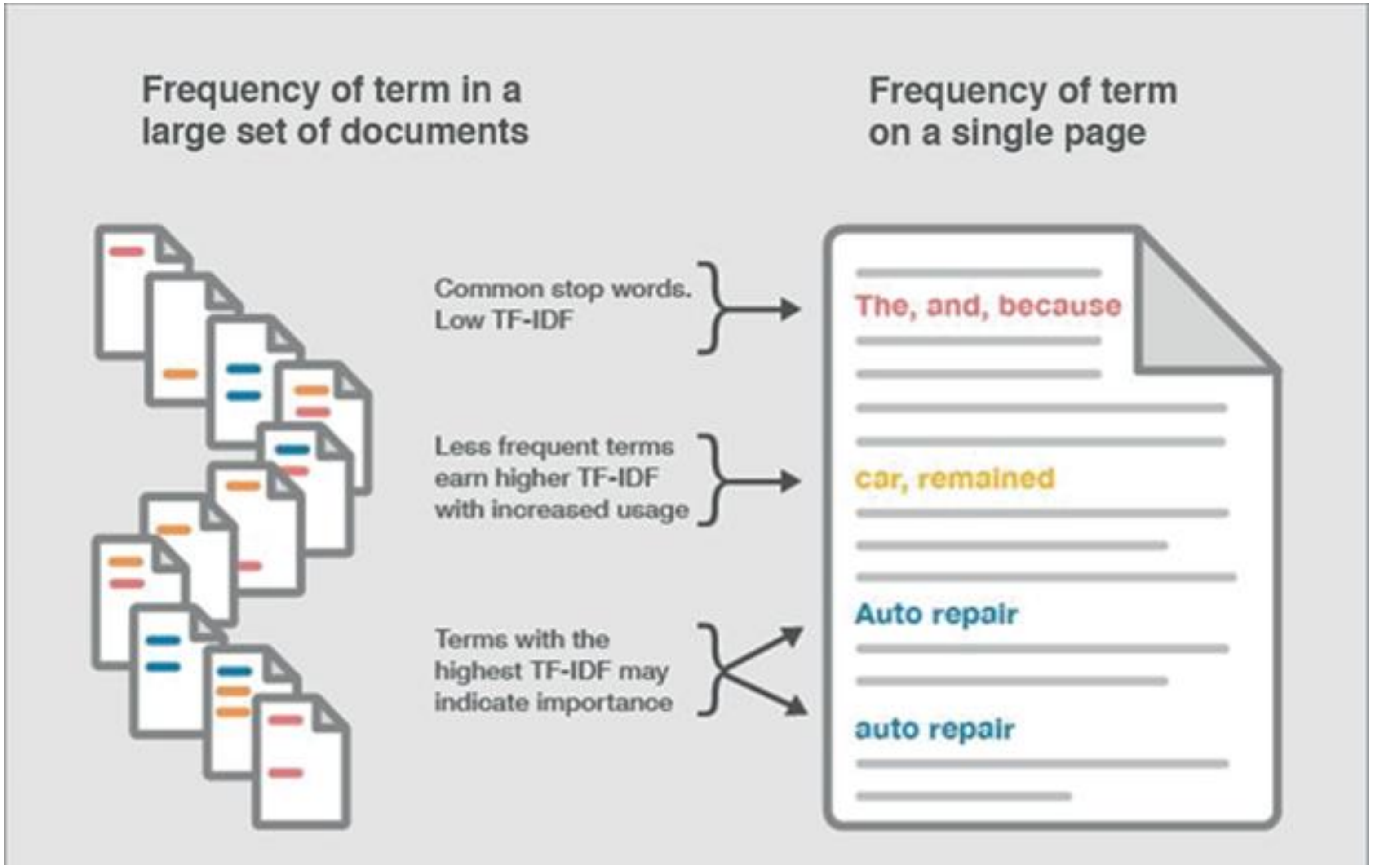
IDF: Variants



Plot of different inverse document frequency functions: standard, smooth, probabilistic (Source: Wikipedia)

TF-IDF

- The tf-idf is defined as: $\boldsymbol{tfidf(t,d,D) = tf(t,d) \cdot idf(t,D)}$



Frequency of term in a large set of documents
Frequency of term on a single page
Common stop words. Low TF-IDF
Less frequent terms earn higher TF-IDF with increased usage
Terms with the highest TF-IDF may indicate importance
The, and, because
car, remained
Auto repair
auto repair

Image credit: Springer

TF-IDF: Represent the query

- We describe a query $\boldsymbol{q}$ in the same way as a document d in the corpus, or slightly differently.
- Recommended tf-idf weighting schemes

Document term weight	Query term weight
$f_{t,d} \cdot \log\left(\frac{N}{n_t}\right)$	$\left(0.5 + 0.5 \frac{f_{t,q}}{\max_{t\prime \in q} f_{t\prime,q}} f_{t,d}\right) \cdot \log\left(\frac{N}{n_t}\right)$
$\log(1 + f_{t,d})$	$\log\left(1 + \frac{N}{n_t}\right)$
$(1 + f_{t,d}) \cdot \log\left(\frac{N}{n_t}\right)$	$(1 + f_{t,d}) \cdot \log\left(\frac{N}{n_t}\right)$

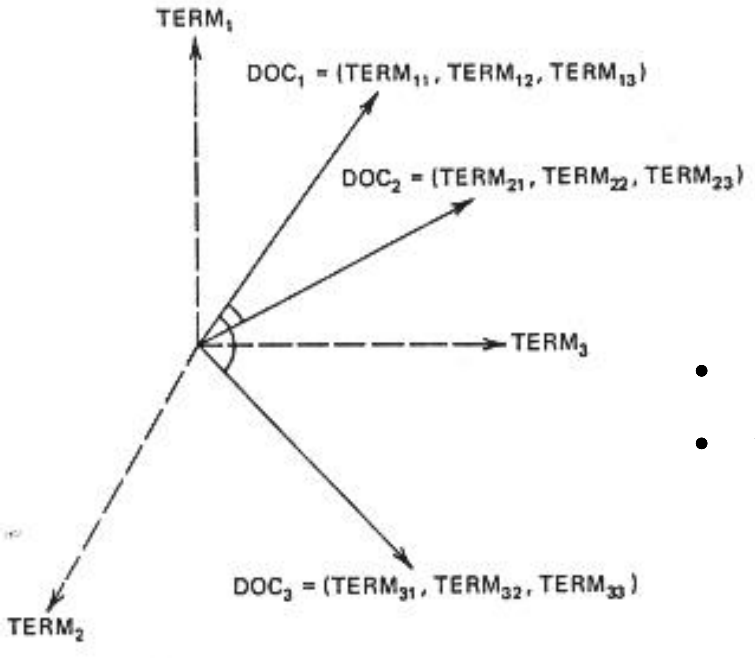
TF-IDF: Calculate the similarity

- The similarity between tf-idf vectors can be estimated by using common metrics.

- Cosine similarity:

$$cosine(\mathbf{d}, \mathbf{q}) = \frac{\langle \overset{\text{document}}{\mathbf{d}} \cdot \overset{\text{query}}{\mathbf{q}} \rangle}{\|\mathbf{d}\| \times \|\mathbf{q}\|}$$

$$= \frac{\sum_{i=1}^{|V|} w_{id} \times w_{iq}}{\sqrt{\sum_{i=1}^{|V|} w_{id}^2} \times \sqrt{\sum_{i=1}^{|V|} w_{iq}^2}}$$



TERM$_1$
DOC$_1$ = (TERM$_{11}$, TERM$_{12}$, TERM$_{13}$)
DOC$_2$ = (TERM$_{21}$, TERM$_{22}$, TERM$_{23}$)
TERM$_3$
DOC$_3$ = (TERM$_{31}$, TERM$_{32}$, TERM$_{33}$)
TERM$_2$

- $|V|$ is the number of terms being considered.
- w_{id} and w_{iq} are the weights of term i in document d and query q, respectively.

TF-IDF: Cosine similarity example

- The following table shows the tfidf weights for the four documents.

	bike	car	fast	highway	road	wheel
Doc 1	0	2/5 · log(4/3)	1/5 · log(4/3)	1/5 · log(4/2)	1/5 · log(4/2)	0
Doc 2	1/5 · log(4/2)	2/5 · log(4/3)	2/5 · log(4/3)	0	0	0
Doc 3	0	0	1/5 · log(4/3)	1/5 · log(4/2)	2/5 · log(4/2)	1/5 · log(4/2)
Doc 4	1/5 · log(4/2)	1/5 · log(4/3)	0	0	0	2/4 · log(4/2)

- Consider the query Q: “fast fast car bike”. For simplicity, we apply the same tfidf scheme for the query.

Query	1/4 · log(4/2)	1/4 · log(4/3)	2/4 · log(4/3)	0	0	0

- Calculate the similarity between the query and the documents

sim(Doc 2, Q) = 0.967 > sim(Doc 4, Q) = 0.315
> sim(Doc 1, Q) = 0.299 > sim(Doc 3, Q) = 0.102

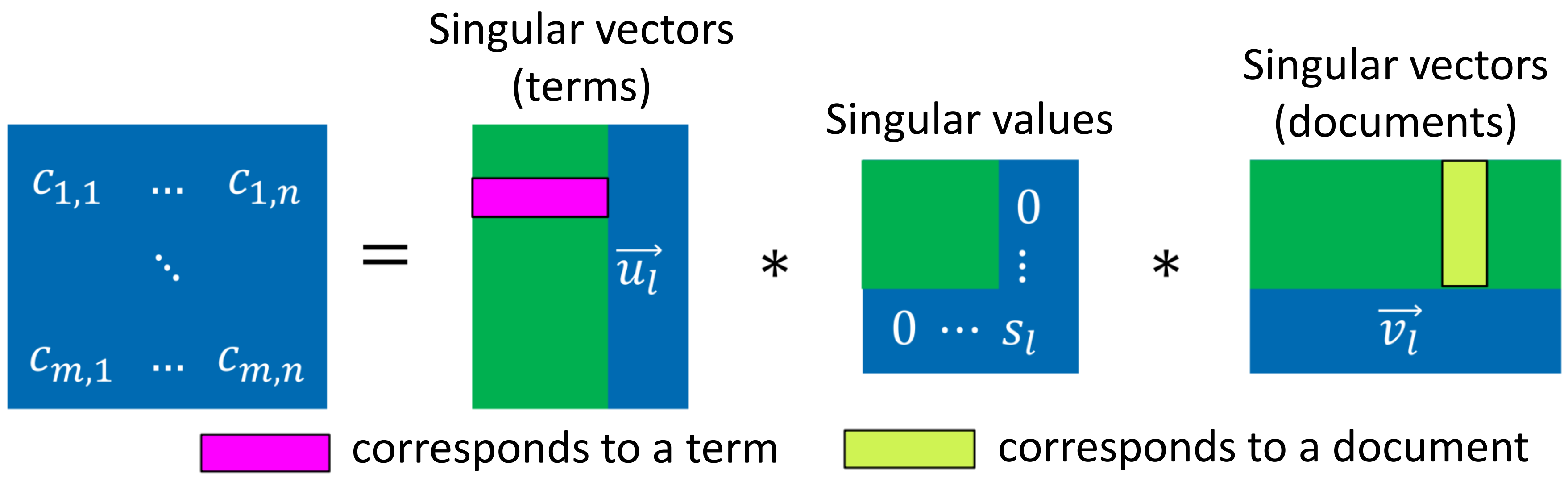
Modern word embeddings

Michael Heck, 2019. The World of Vectors. Dialog Systems and Machine Learning. Heinrich-Heine-Universität Düsseldorf.

Latent semantic analysis (LSA)

- LSA computes a term-document matrix A by using SVD.

$$\boldsymbol{A = U * S * V^T}$$



- Dimensional reduction is done by omitting singular values.
- Term and document vectors are treated as semantic spaces.

Latent semantic analysis (LSA)	
Capture the meaning	Difficult to interpret

word2vec (2013)

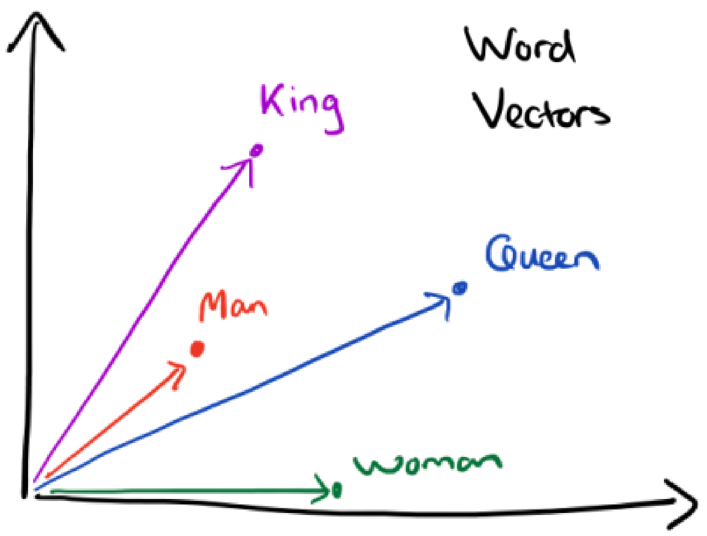
- word2vec uses a neural network to learn word associations from a large corpus.
  - Word embeddings are a by-product of solving a prediction task.
- Each distinct term in the corpus is assigned a corresponding vector in the space, typically of several hundred dimensions.

word2vec (2013)

- It performs operations with vectors to answer questions

$$A - B + C = ?$$

  - What is to C in the same sense as B is to A?
- Word closest by cosine distance is taken as answer.



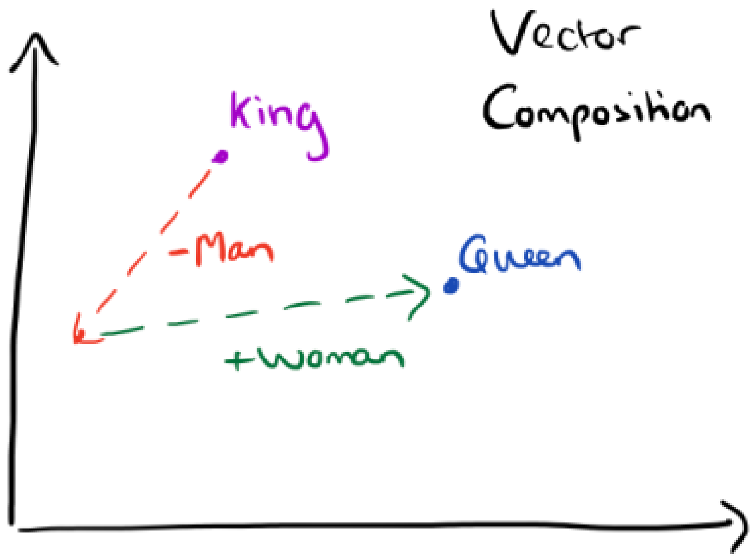
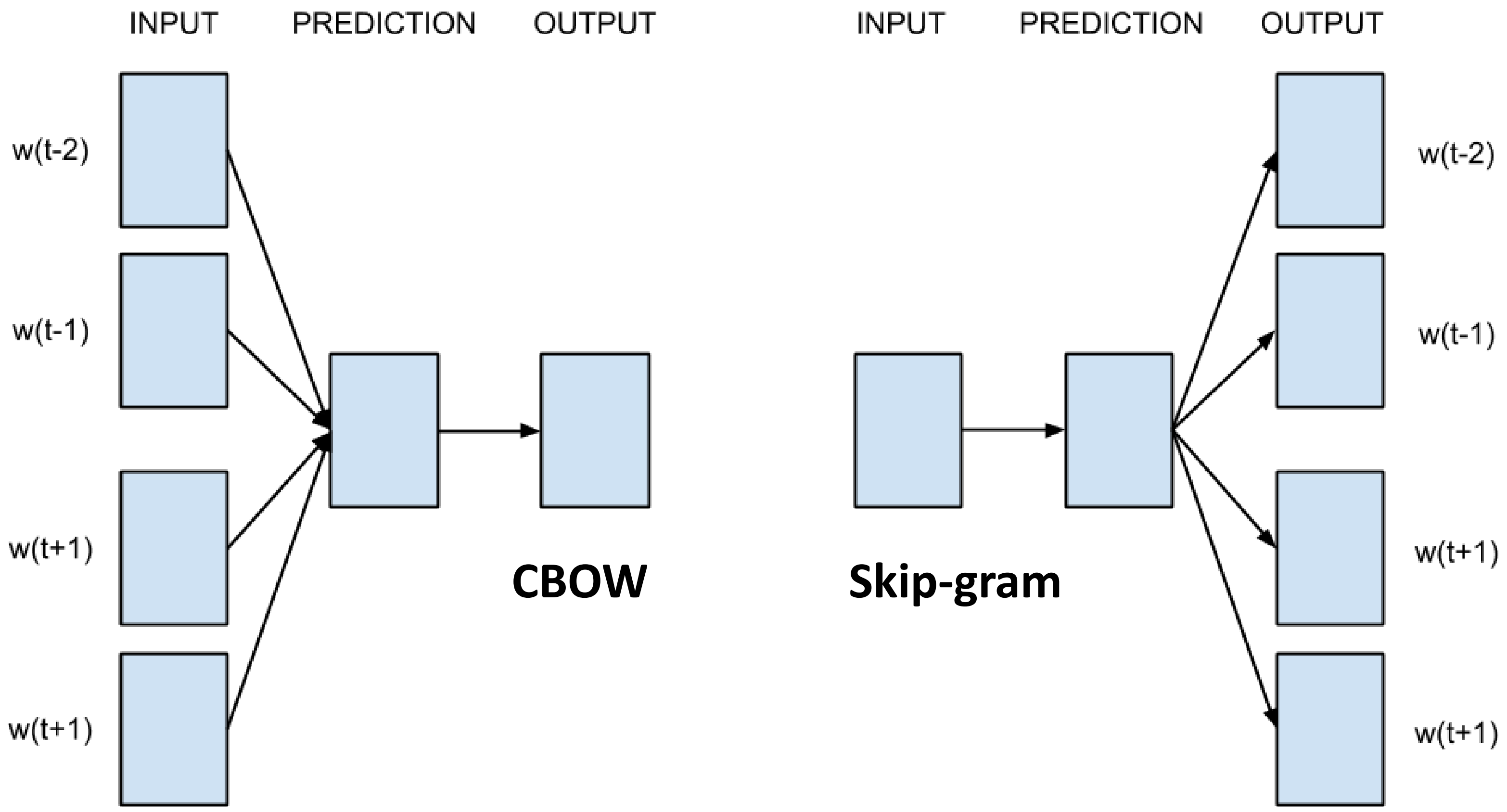


Image credit: KDnuggets, The Amazing Power of Word Vectors

word2vec (2013)

- It utilizes either of the two architectures: continuously sliding bag-of-words (CBOW) or continuously sliding skip-gram.



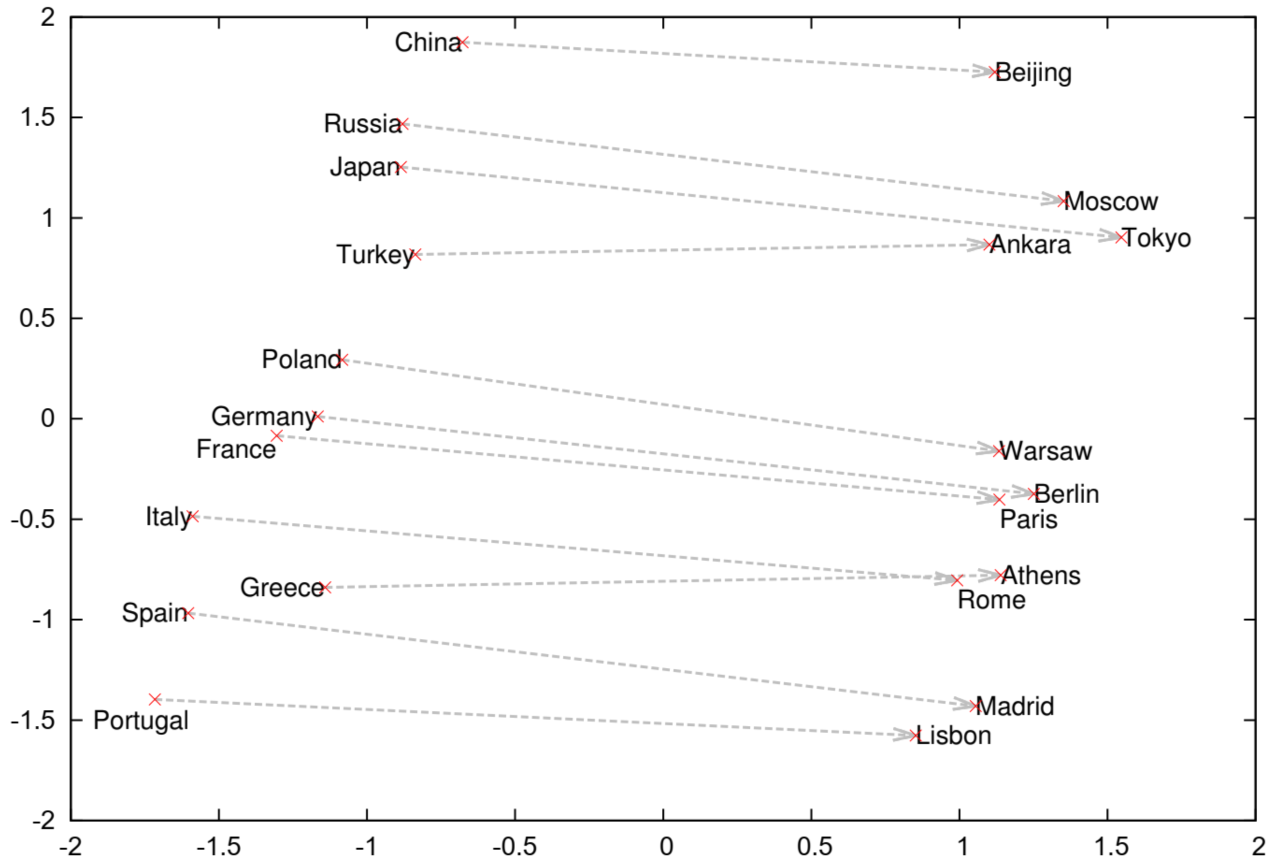
word2vec: Word pair relationships

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

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

Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

word2vec (2013)



Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.

word2vec: Vector compositionality

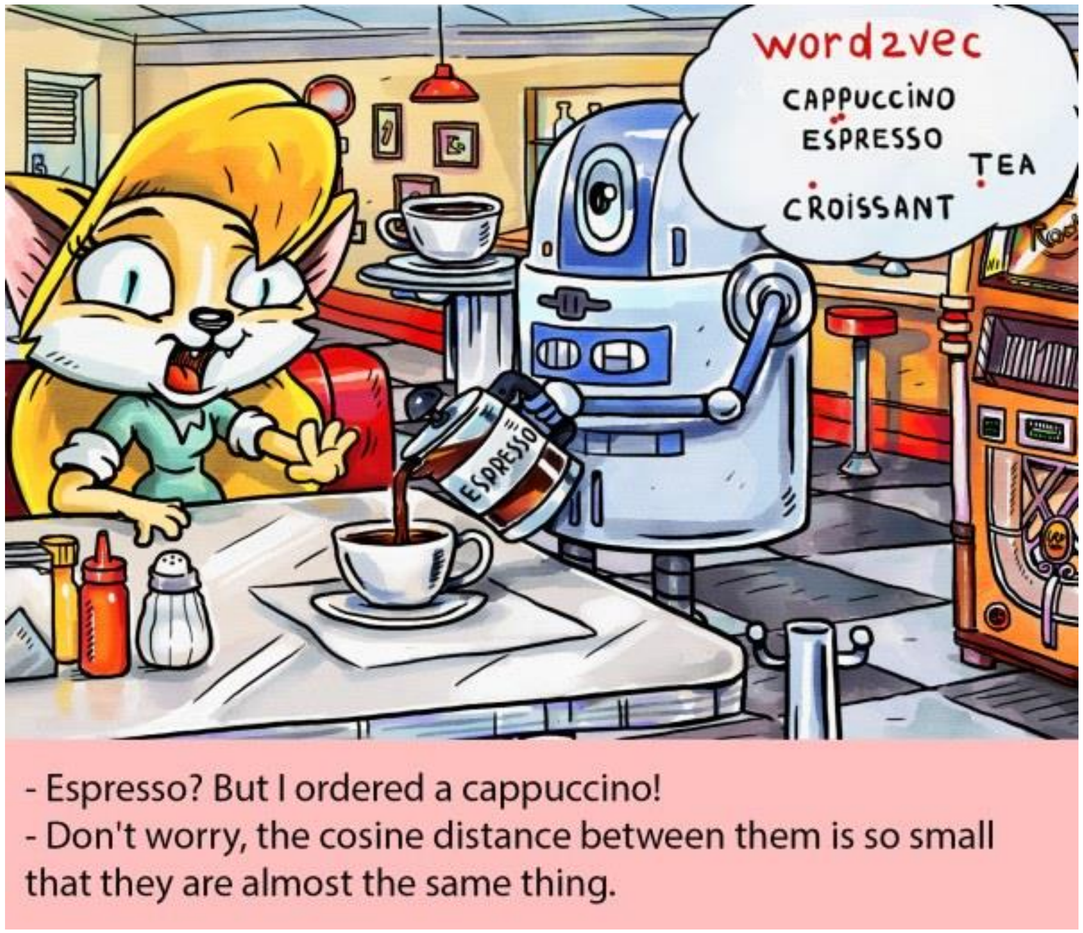
Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
Koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zloty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.

Latent semantic analysis (LSA)	
Capture the meaning	Difficult to interpret

word2vec	
Intuitive	Local context

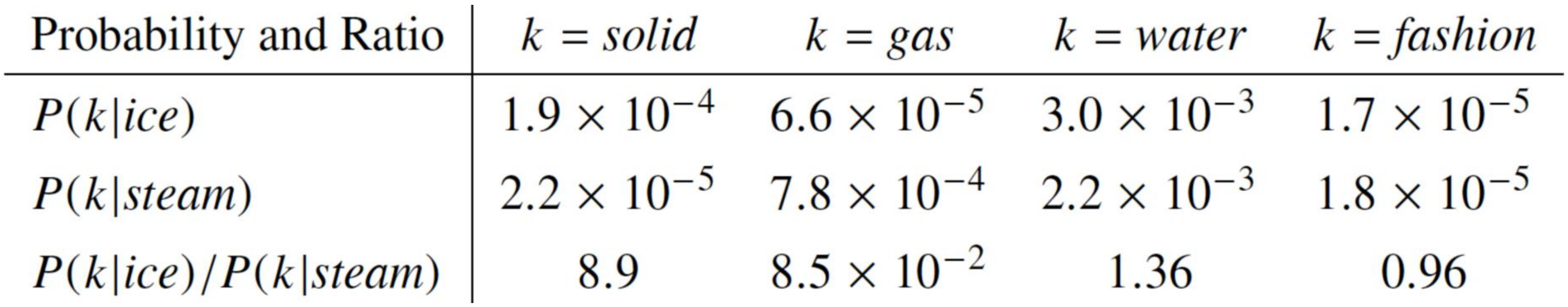


- Espresso? But I ordered a cappuccino!
- Don't worry, the cosine distance between them is so small that they are almost the same thing.

GloVe – Global Vectors (2014)

- GloVe considers the global context by using a co-occurrence matrix to capture overall statistics.
- **Intuition:** The ratio of word-word co-occurrence probabilities has the potential for encoding some form of meaning.

Co-occurrence probabilities for target words **ice** and **steam** with selected context words from a corpus of 6 billion token.



Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Pennington, J., Socher, R. and Manning, C.D., 2014. Glove: Global vectors for word representation. In Proceedings of EMNLP (pp. 1532-1543).

GloVe (2014)

- It is essentially a log-bilinear model with a weighted least-squares objective.

co-occurrence probabilities

$$J = \sum_{i,j=1}^{V} \left(w_i^T \widetilde{w_j} - log(P_{ij}) \right)^2$$

product of word vector and context word vector

- **Goal:** Learn word vectors such that their product equals the log of their co-occurrence probability
  - J associates co-occurrence probability ratios with vector differences $\rightarrow$ the meaning is also encoded in the vector differences.

Latent semantic analysis (LSA)	
Capture the meaning	Difficult to interpret

⇩

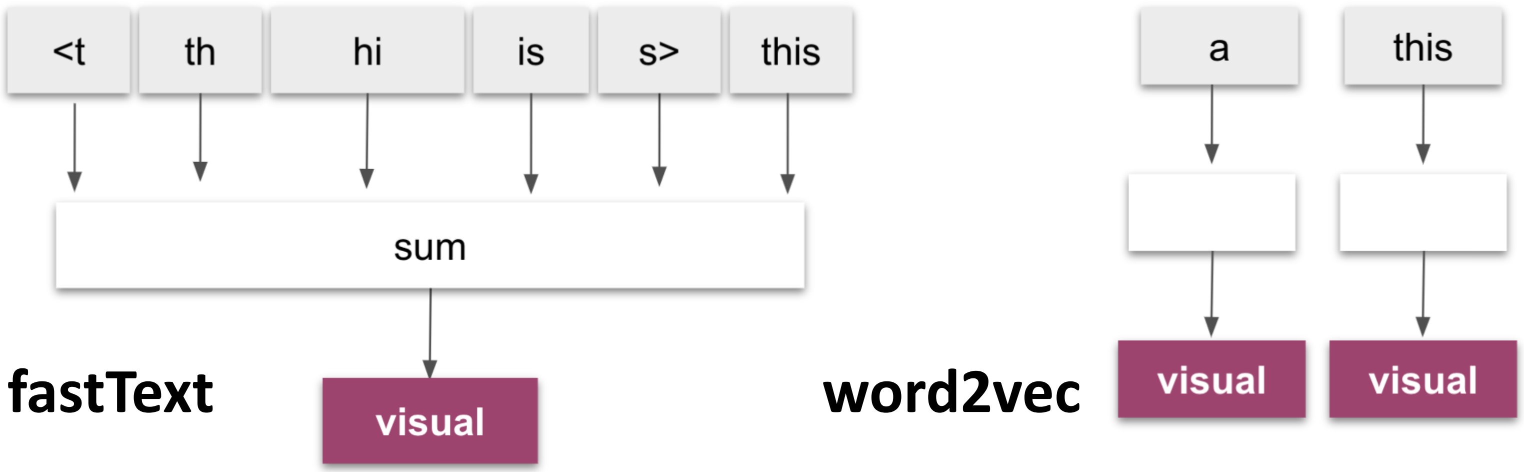
word2vec	
Intuitive	Local context

⇩

GloVe	
Global context	No Out-of-Vocabulary handling

fastText (2017)

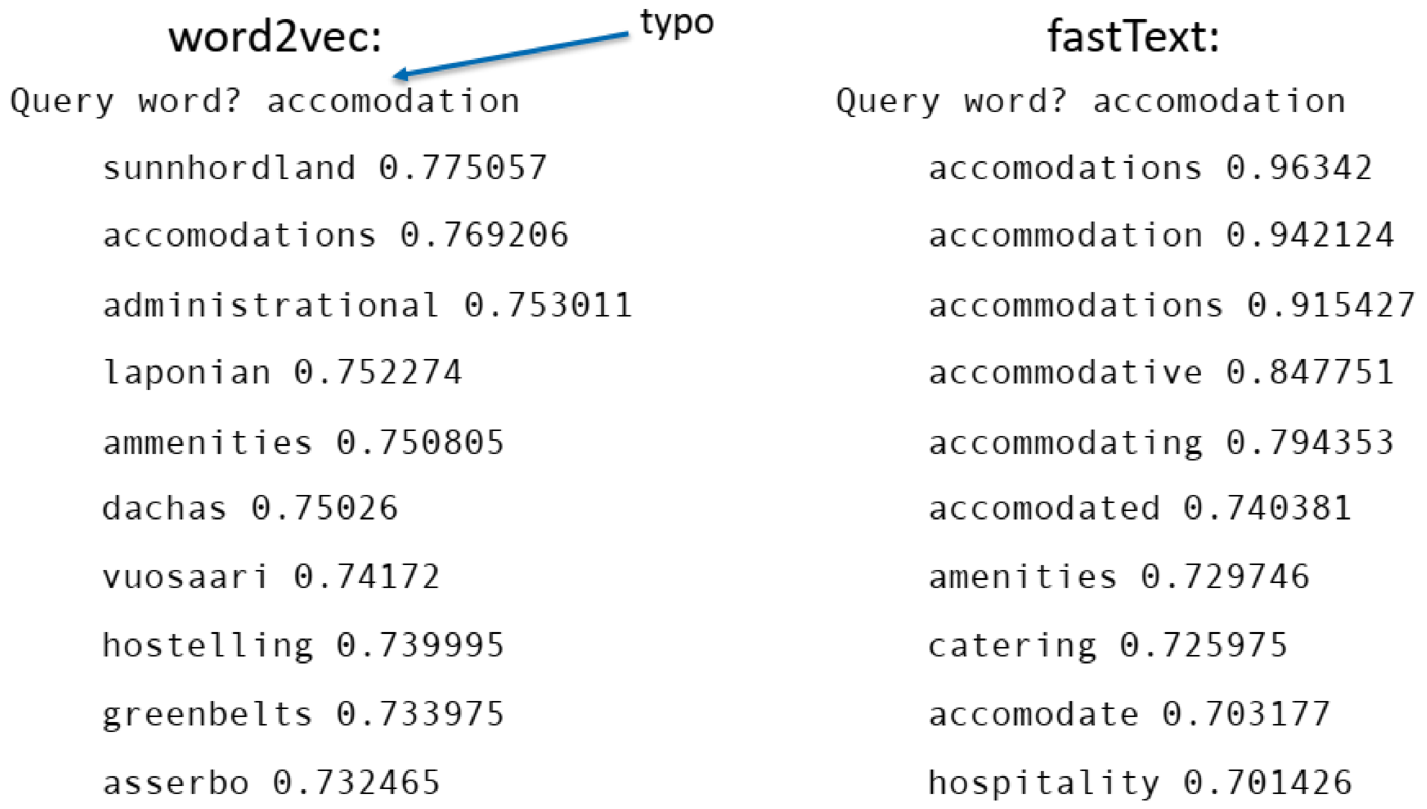
- fastText views words as compositions of character n-grams.
  - word2vec and GloVe consider "words" as smallest units.



- It creates better embeddings for rare words and constructs vectors for unseen (OOV) words.
- Hyperparameter choice is critical for performance

fastText (2017): An example

- An example that demonstrates the importance of sub-word information.



```
Query word? accomodation

    sunnhordland 0.775057
    accomodations 0.769206
    administrational 0.753011
    laponian 0.752274
    ammenities 0.750805
    dachas 0.75026
    vuosaari 0.74172
    hostelling 0.739995
    greenbelts 0.733975
    asserbo 0.732465
```

```
Query word? accomodation

    accomodations 0.96342
    accommodation 0.942124
    accommodations 0.915427
    accommodative 0.847751
    accommodating 0.794353
    accomodated 0.740381
    amenities 0.729746
    catering 0.725975
    accomodate 0.703177
    hospitality 0.701426
```

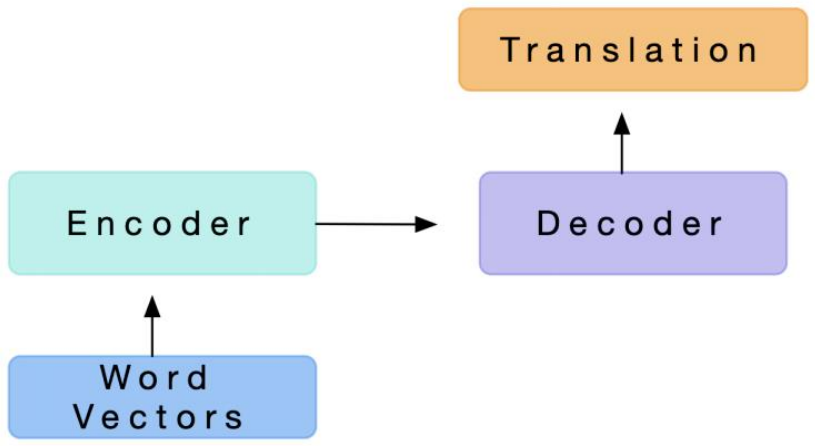
Image credit: https://fasttext.cc

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., 2017. Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5, pp.135-146.

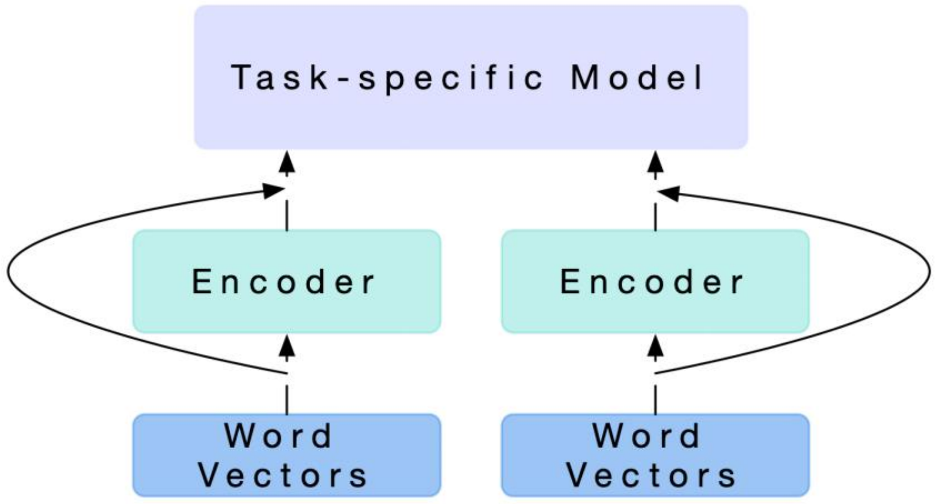
Latent semantic analysis (LSA)	
Capture the meaning	Difficult to interpret
word2vec	
Intuitive	Local context
GloVe	
Global context	No Out-of-Vocabulary handling
fastText	
OOV handling	Not contextual

CoVe (2017)

- **Co**ntextualized word **Ve**ctors are attained by leveraging machine translation (MT).
  - MT is assumed to be general enough to get the words' meanings.
- This enables sense-specific representations for homographs.



Training of encoder-decoder architecture for MT



Encoders are utilized to generate word vectors

CoVe (2017)

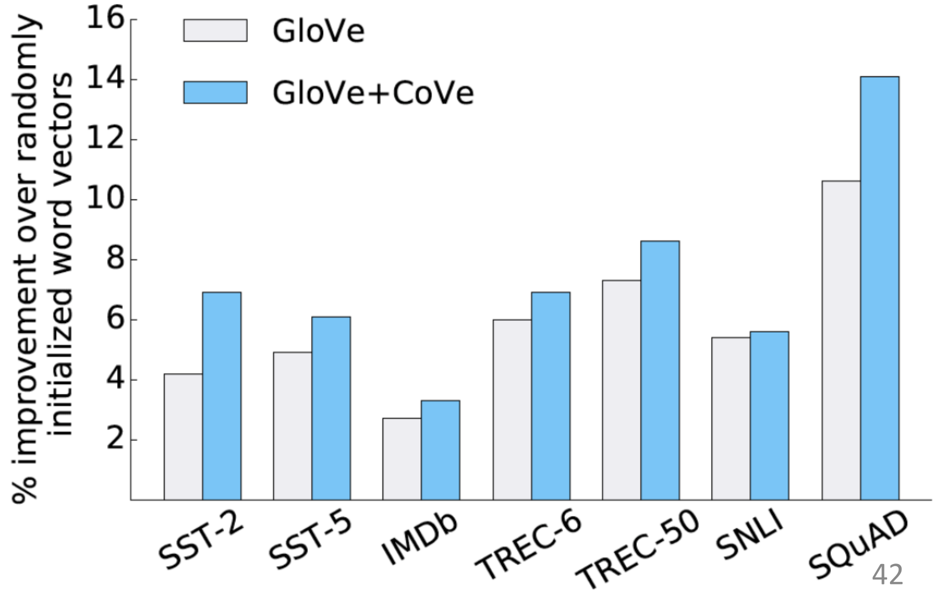
- The sequence of context vectors produced by encoder is

$$\mathbf{CoVe}(w) = \mathbf{MT} - \mathbf{LSTM}\big(\mathbf{GloVe}(w)\big)$$

sequence of words ↗ ↖ sequence of GloVes

- It serves as input for downstream tasks

$$\tilde{w} = [\mathbf{GloVe}(w); \mathbf{CoVe}(w)]$$



McCann, B., Bradbury, J., Xiong, C. and Socher, R., 2017. Learned in translation: Contextualized word vectors. Advances in neural information processing systems, 30.

Latent semantic analysis (LSA)	
Capture the meaning	Difficult to interpret
word2vec	
Intuitive	Local context
GloVe	
Global context	No Out-of-Vocabulary handling
fastText	
OOV handling	Not contextual
CoVe	
Contextual	Supervised

ELMo (2018)

- **E**mbeddings from **L**anguage **Mo**del learns word embeddings through building bidirectional language models (LMs).
- It is a deep contextualized word representation with various levels of knowledge.
  - Each token is described as a function of the entire input sentence.
- No labels needed $\rightarrow$ Unsupervised learning problem
- Out-of-vocabulary words can be accepted as input.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In NAACL-HLT (pp. 2227–2237)

ELMo (2018): Bidirectional LM

- A bidirectional language model (biLM) includes the forward and backward language models.
  - Forward: $p(t_1, t_2, \dots, t_N) = \prod_{k=1}^{N} p(t_1 |\, t_1, t_2, \dots, t_{k-1})$
  - Backward: $p(t_1, t_2, \dots, t_N) = \prod_{k=1}^{N} p(t_k |\, t_{k+1}, t_{k+2}, \dots, t_N)$
    - t_k is the k^{th} term (token) in a sentence of N terms.
- A word t_k is defined as a linear combination of hidden layers.
- biLM parameters are fixed, weighting & scaling is learned
- Higher layers seem to capture semantics, lower layers for syntactics

ELMo (2018): Build bidirectional LMs

- We can build a bidirectional LM with two LSTMs predicting the next words in both directions.

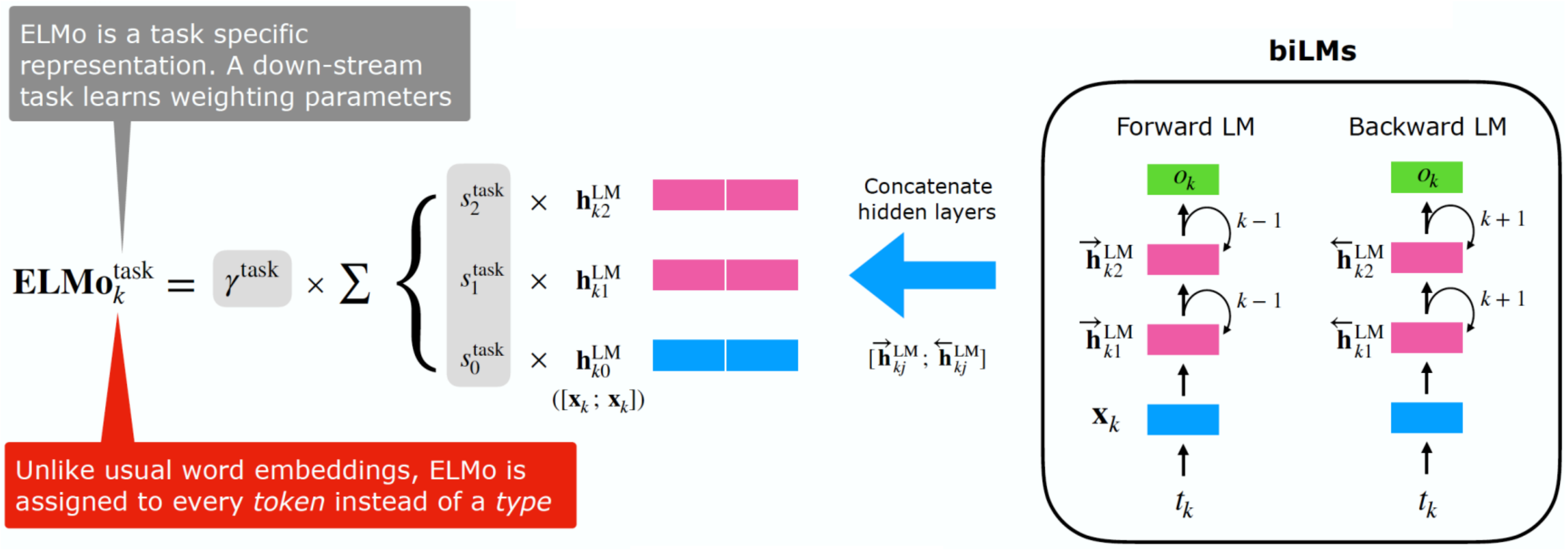

ELMo is a task specific representation. A down-stream task learns weighting parameters
$\mathbf{ELMo}_k^{task} = \gamma^{task} \times \sum \{ s_2^{task} \times \mathbf{h}_{k2}^{LM},\ s_1^{task} \times \mathbf{h}_{k1}^{LM},\ s_0^{task} \times \mathbf{h}_{k0}^{LM}$
$([\mathbf{x}_k ; \mathbf{x}_k])$
Concatenate hidden layers
$[\overrightarrow{\mathbf{h}}_{kj}^{LM} ; \overleftarrow{\mathbf{h}}_{kj}^{LM}]$
biLMs
Forward LM
Backward LM
o_k
$k-1$
$k+1$
$\overrightarrow{\mathbf{h}}_{k2}^{LM}$
$\overrightarrow{\mathbf{h}}_{k1}^{LM}$
$\overleftarrow{\mathbf{h}}_{k2}^{LM}$
$\overleftarrow{\mathbf{h}}_{k1}^{LM}$
$\mathbf{x}_k$
t_k
Unlike usual word embeddings, ELMo is assigned to every *token* instead of a *type*

Image credit: SlideShare, A Review of Deep Contextualized Word Representations. Yada, Shuntaro (2018)

ELMo (2018): Usage

- ELMo vectors are used as additional features in NLP tasks with simple concatenation to the embedding layer.

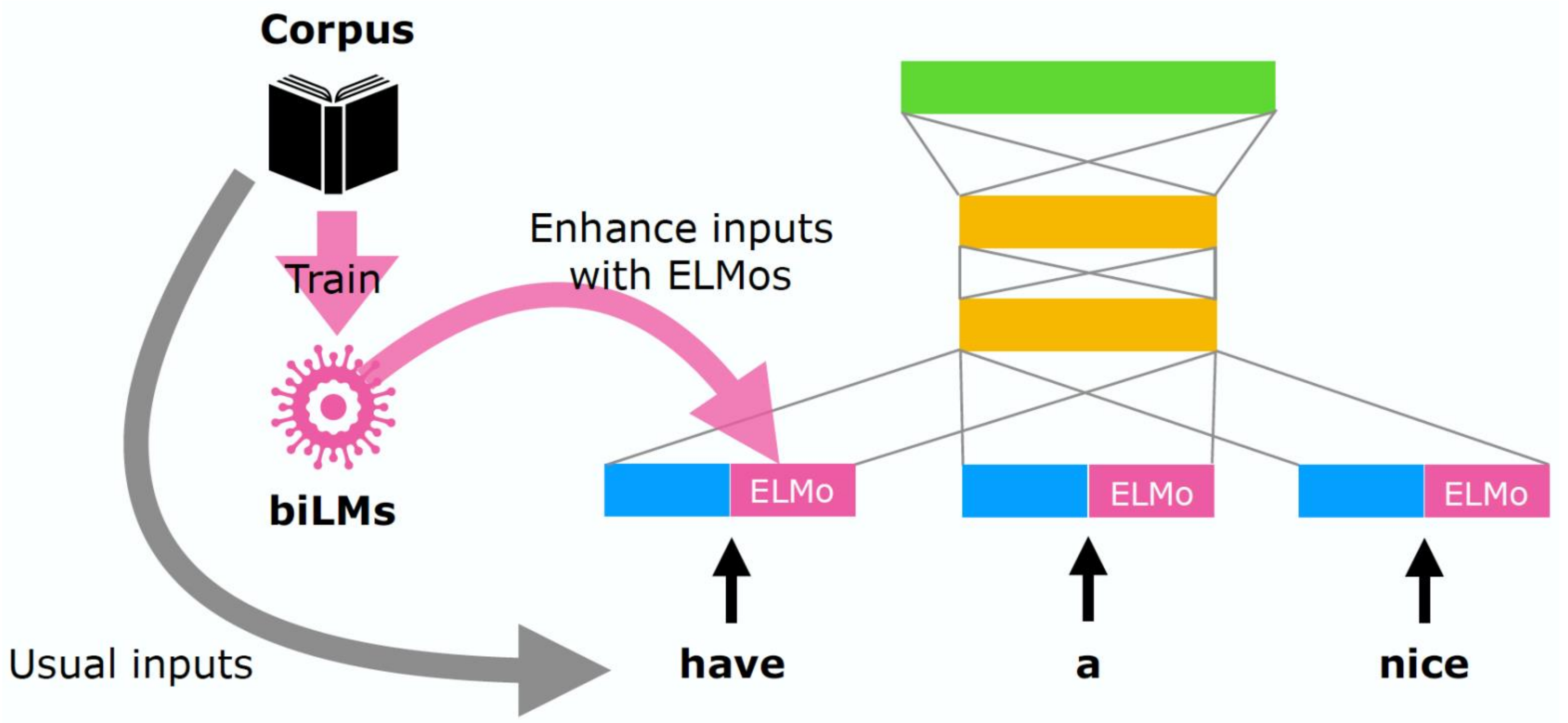
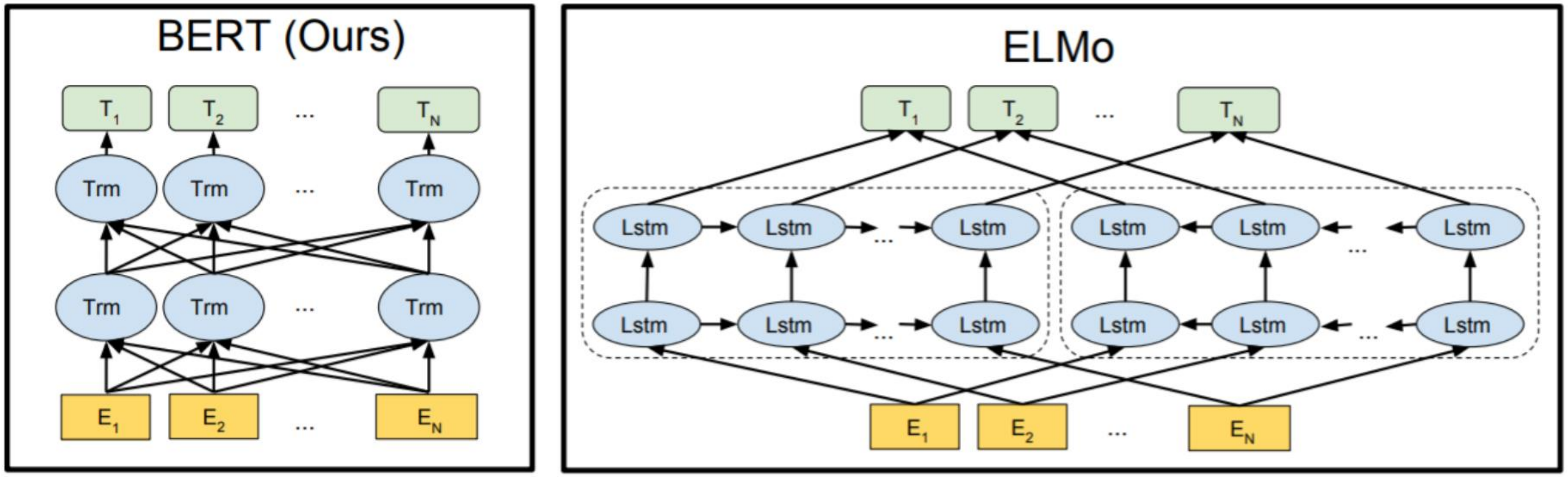


Image credit: SlideShare, A Review of Deep Contextualized Word Representations. Yada, Shuntaro (2018)

Latent semantic analysis (LSA)	
Capture the meaning	Difficult to interpret
word2vec	
Intuitive	Local context
GloVe	
Global context	No Out-of-Vocabulary handling
fastText	
OOV handling	Not contextual
CoVe	
Contextual	Supervised
ELMo	
Unsupervised	Not truly bidirectional

BERT (2018)

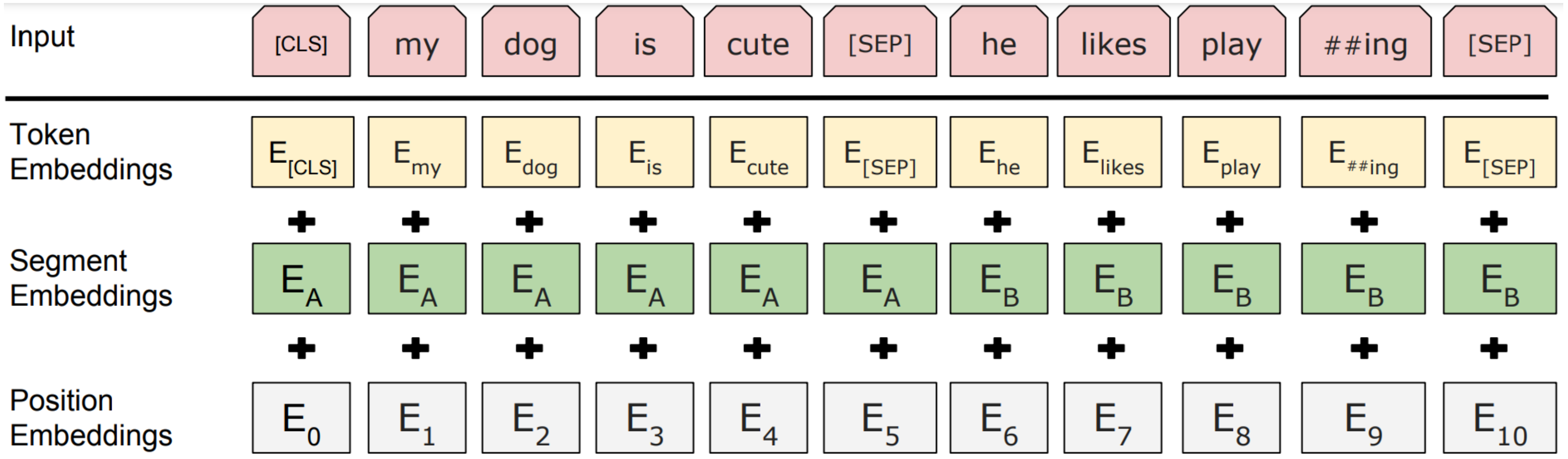
- **B**i-directional **E**ncoder **R**epresentations from **T**ransformers
- The representations are jointly conditioned on left and right context in all layers.



BERT uses a bidirectional Transformer, while ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs.

BERT (2018): Input representation

- An input embedding is the sum of the token embedding, segmentation embedding and position embedding.


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

BERT (2018): Pre-training

- **Pre-training Task 1 Masked LM**
- Mask 15% of all tokens in each sequence at random
  - Input: the man went to the [MASK1] . he bought a [MASK2] of milk
  - Label: [MASK1] = went – [MASK2] = gallon
- The data generator does not always replace the chosen words with [MASK], instead it will
  - 80% of the time: Replace the word with the [MASK] token
    - E.g., my dog is hairy $\rightarrow$ my dog is [MASK]
  - 10% of the time: Replace the word with a random word
    - E.g., my dog is hairy $\rightarrow$ my dog is apple
  - 10% of the time: Keep the word unchanged
    - E.g., my dog is hairy $\rightarrow$ my dog is hairy
    - This is to bias the representation towards the actual observed word.

BERT (2018): Pre-training

- **Pre-training Task 2 Next Sentence Prediction**
- An LM does not always understand relationships between sentences, which is important for many NLP tasks.
- BERT pre-train a binarized next sentence prediction task.
- Input: the man went to the store [SEP] he bought a gallon of milk

  $\rightarrow$ Label: IsNext
- Input: the man went to the store [SEP] penguins are flightless birds

  $\rightarrow$ Label: NotNext

BERT (2018): Fine-tuning

- The final hidden state for the first token [cls] in the input is taken as the representation of the input sequence.
- During fine-tuning, we only add new parameters for the classification layer.
- All the parameters are fine-tuned jointly to maximize the log-probability of the correct label.

BERT (2018): Feature extraction

- Pre-trained embeddings can be fed to an existing model.
- The results are not far behind fine-tuning BERT on a task such as named-entity recognition.

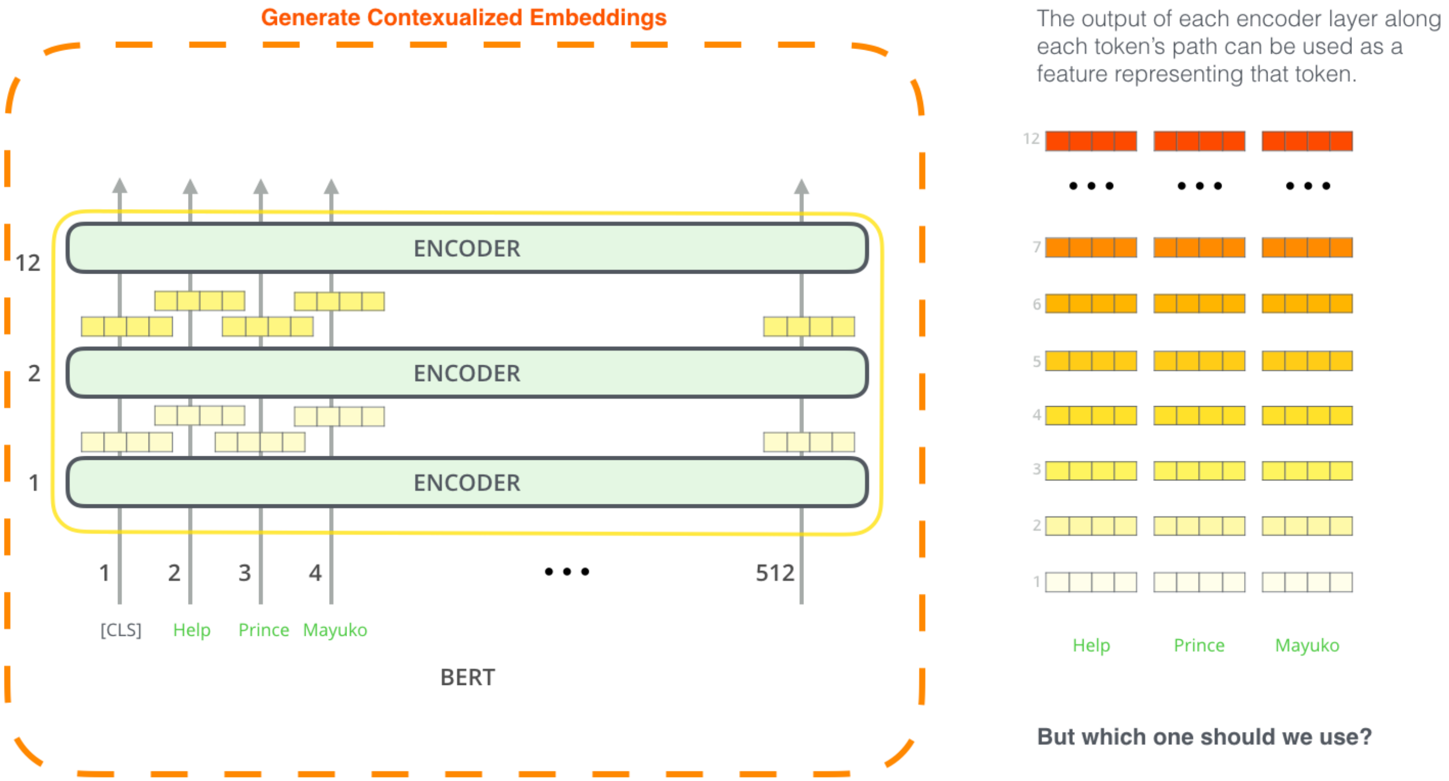
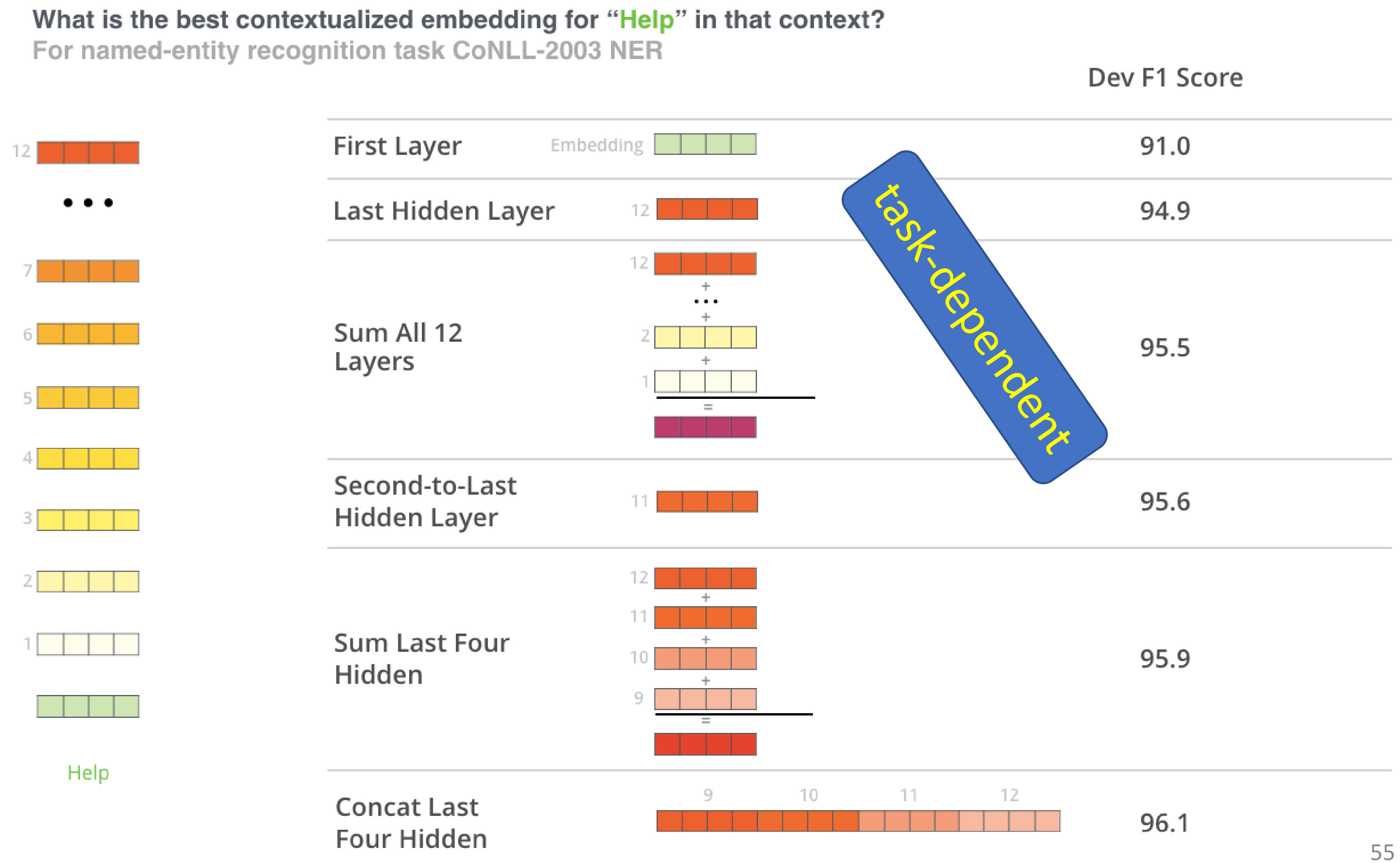

Generate Contexualized Embeddings
12
2
1
ENCODER
ENCODER
ENCODER
1
2
3
4
• • •
512
[CLS]
Help
Prince
Mayuko
BERT
The output of each encoder layer along each token's path can be used as a feature representing that token.
12
7
6
5
4
3
2
1
Help
Prince
Mayuko
But which one should we use?

Image credit: jalammar.github.io

BERT (2018): Feature extraction


What is the best contextualized embedding for “Help” in that context?

For named-entity recognition task CoNLL-2003 NER

	Dev F1 Score
First Layer	91.0
Last Hidden Layer	94.9
Sum All 12 Layers	95.5
Second-to-Last Hidden Layer	95.6
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1

Latent semantic analysis (LSA)	Capture the meaning	Difficult to interpret
word2vec	Intuitive	Local context
GloVe	Global context	No Out-of-Vocabulary handling
fastText	OOV handling	Not contextual
CoVe	Contextual	Supervised
ELMo	Unsupervised	Not truly bidirectional
BERT	Bidirectional	Computationally expensive

Exercises

1. Construct TF-IDF vectors

- Consider a collection of three documents. No preprocessing required.
- The vocabulary is V = { arrived, gold, shipment, silver, truck }.

  d1: shipment of gold damaged in a fire

  d2: shipment of gold arrived in a truck

  d3: delivery of silver arrived in a silver truck

- Define the TF-IDF vectors for the above documents on the given vocabulary
- Consider the queries q: **gold silver truck.** Compute the Cosine similarity between q and each of the above documents.

References

- Michael Heck, 2019. The World of Vectors. Dialog Systems and Machine Learning. Heinrich-Heine-Universität Düsseldorf.
- Wikipedia: tf-idf

  https://en.wikipedia.org/wiki/Tf%E2%80%93idf
- GloVe: Global Vectors for Word Representation

  https://nlp.stanford.edu/projects/glove/
- A Visual Guide to FastText Word Embeddings

  https://amitness.com/2020/06/fasttext-embeddings/
- The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)

  http://jalammar.github.io/illustrated-bert/