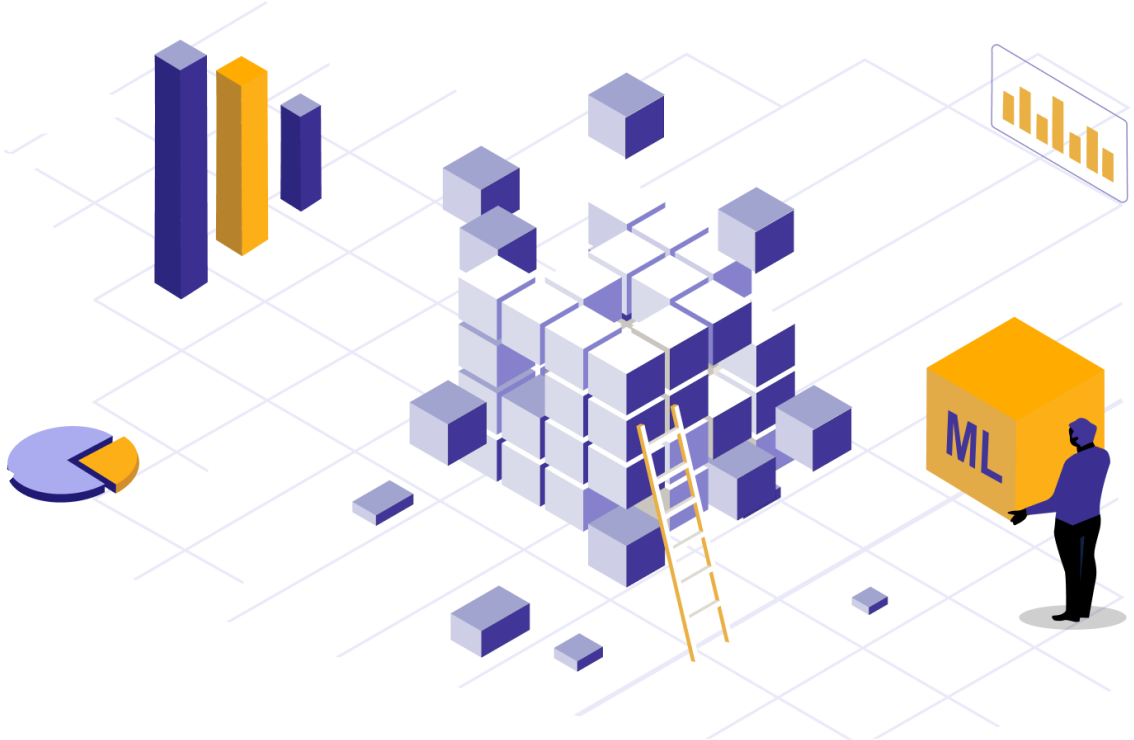


Feature Engineering

Part I

Nguyen Ngoc Thao

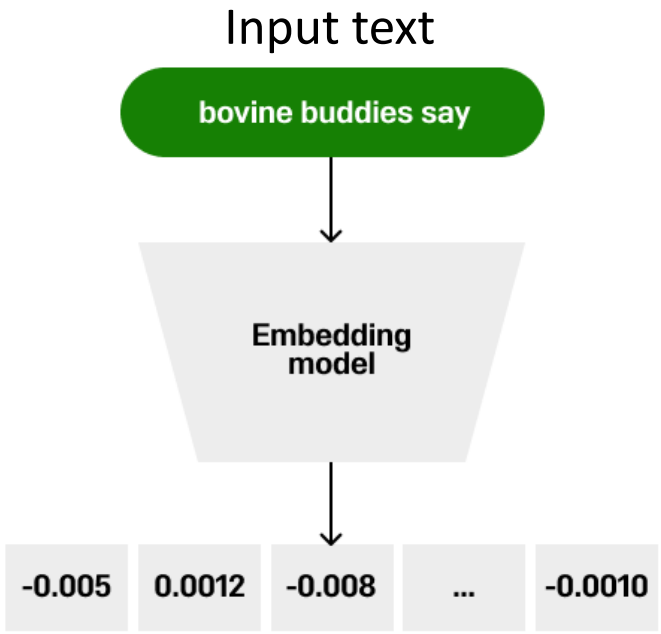
nnthao@fit.hcmus.edu.vn

Content outline

- Feature engineering: A definition
- Conventional text encodings
- Modern word embeddings
- Similarity metrics for text

Feature engineering

- Feature engineering involves selecting, transforming, and creating relevant features from raw data.
- This aims to enhance the performance of ML models.



Word embedding

Input image Edge features

Feature engineering: Key steps

- Feature selection: Pick the most relevant features from the data to reduce noise and enhance model accuracy.
  - Statistical tests, correlation analysis, or feature importance scores.
- Feature transformation: Modify features to suit the model.
  - Encoding (e.g., one-hot encoding, label encoding), normalization (e.g., min-max, standardization, log/power transforms).
- Feature creation: Generate new features from existing data to capture valuable patterns and insights.
- Dimensionality reduction: Lower the dimensionality of the feature space while retaining the most important information.
  - E.g., Principal Component Analysis (PCA)

Feature engineering: Importance

- Improve model performance: Good features make patterns in the data more apparent, leading to better predictions.
- Reduce overfitting: Deleting irrelevant features help simplify the model.
- Optimize training time: A focused feature set can minimize computational requirements.

- Feature engineering involves creativity, domain expertise, and experimentation to achieve the best results.

Feature extraction

- Handcrafted features are manually engineered by the scientist.

Traditional Machine Learning Flow

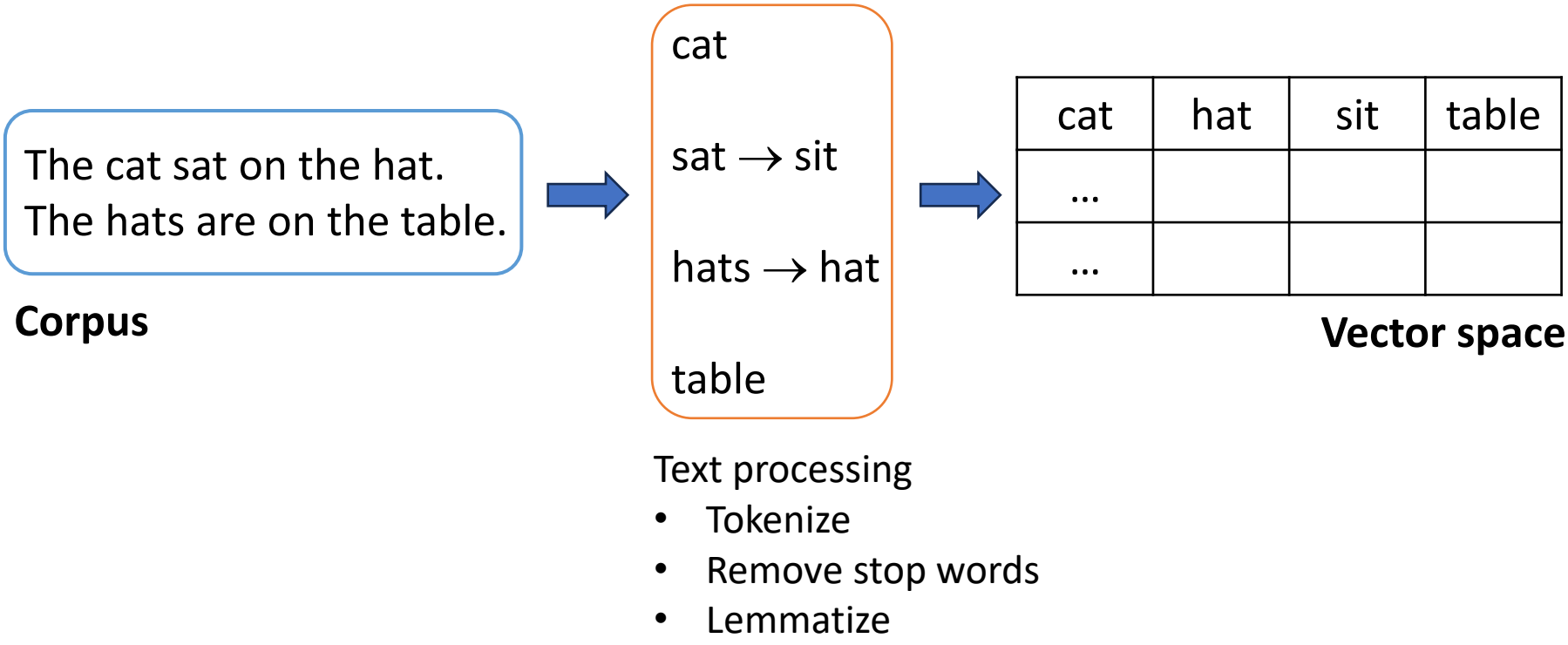
Deep Learning Flow

- Automatic features are inherently learned from a ML algorithm.

Text encodings

Document vectorization

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



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 corpus.
  - 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 → suitable for models like RNN, etc.

Loss

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

One-hot encoding: An example

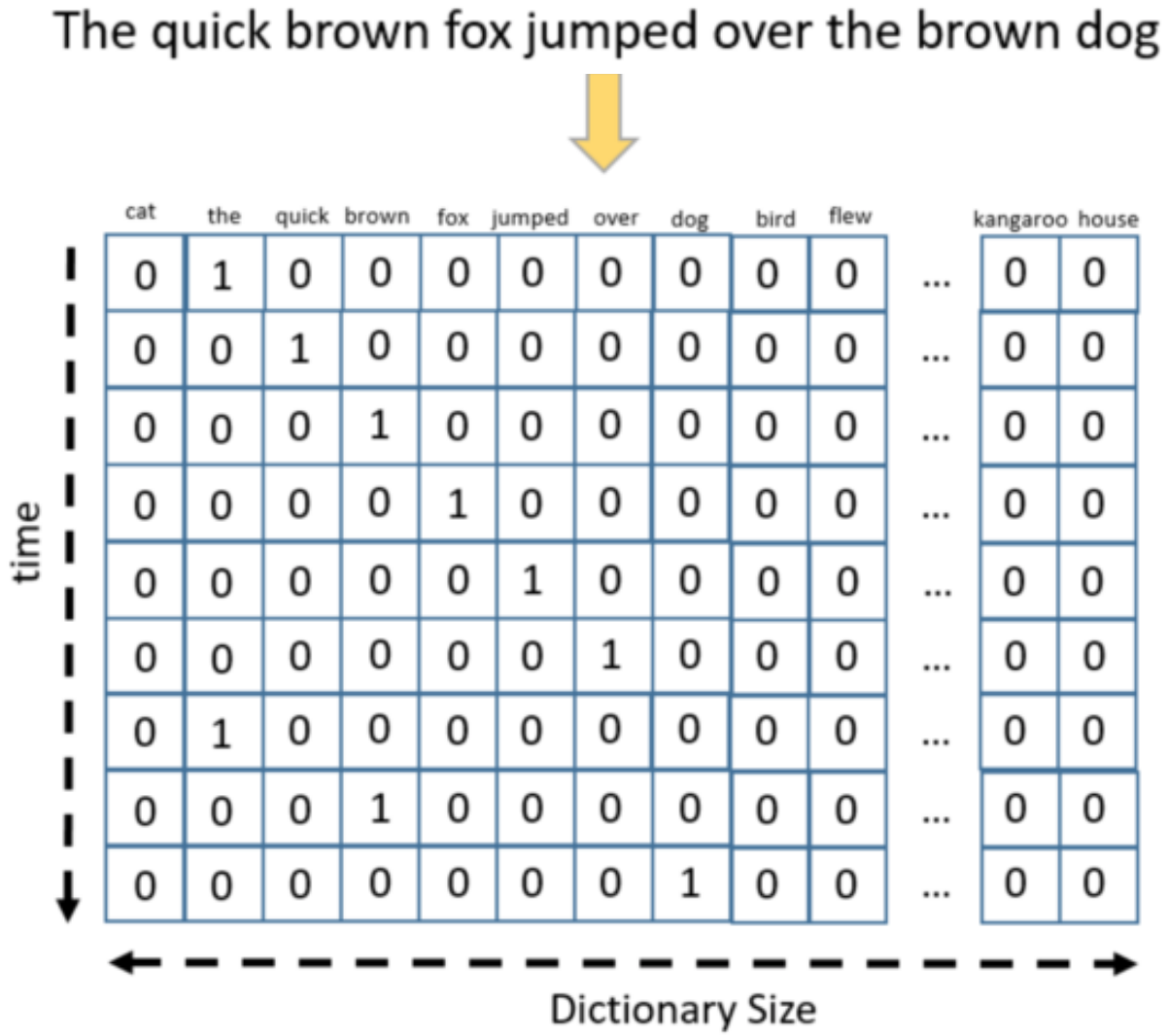


Image credit: Data Science Central

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 → lose the frequency of words.
- Terms are unordered → lose the context of words.

Binary encoding: An example

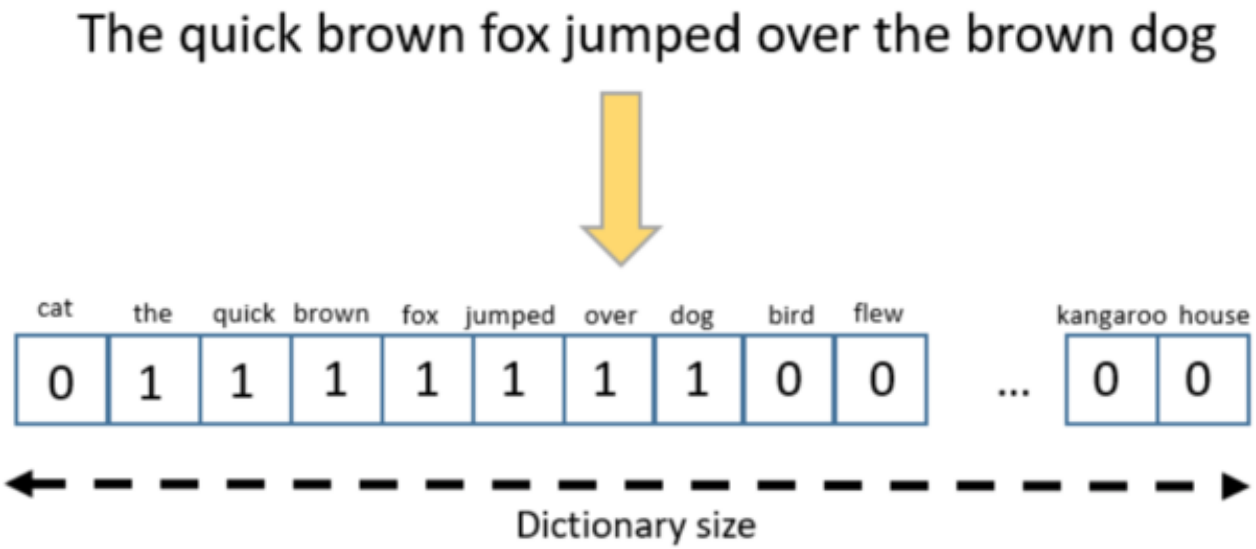


Image credit: Data Science Central

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.

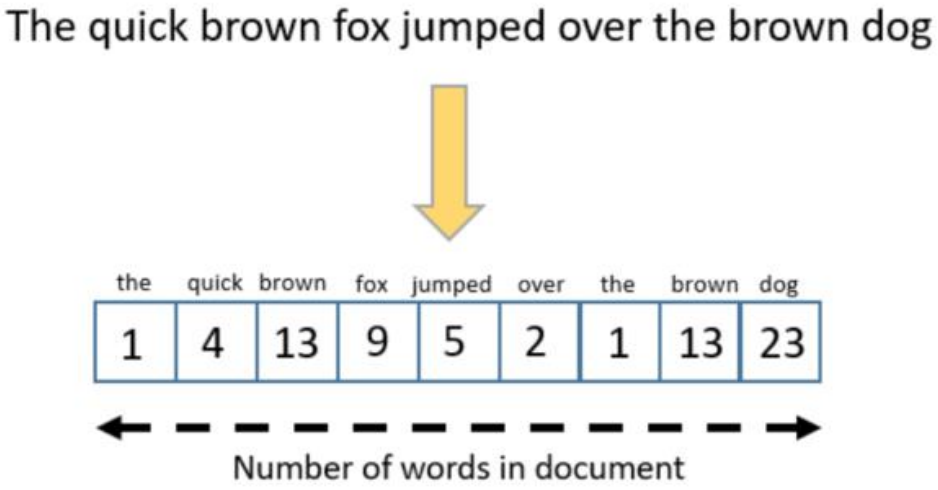


Image credit:
Data Science Central

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 $\boldsymbol{tf(t,d)}$ of a term t in doc d is given by

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

  - 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 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

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 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)

TF-IDF

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

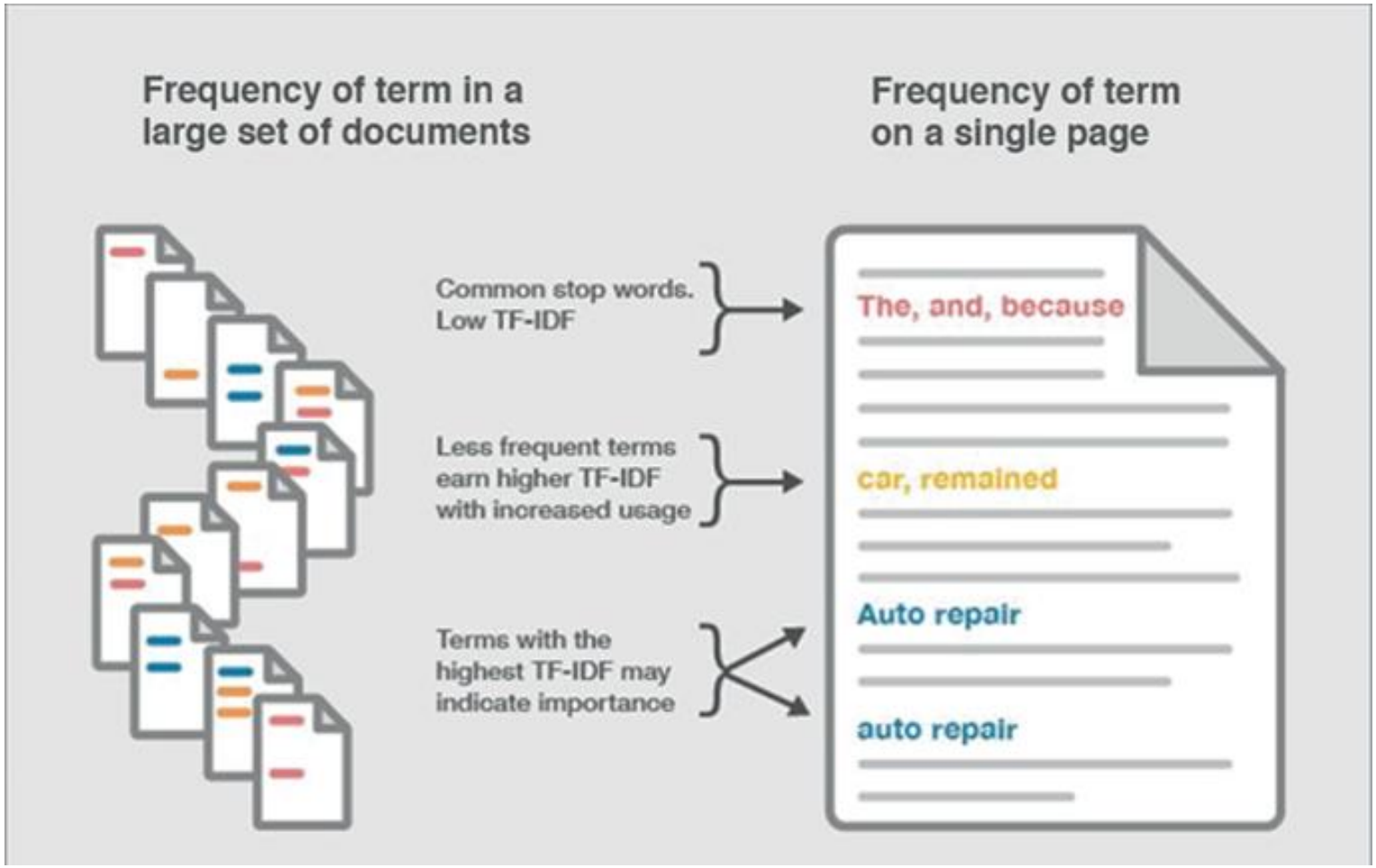


Image credit: Springer

Term frequency (TF): Variants

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

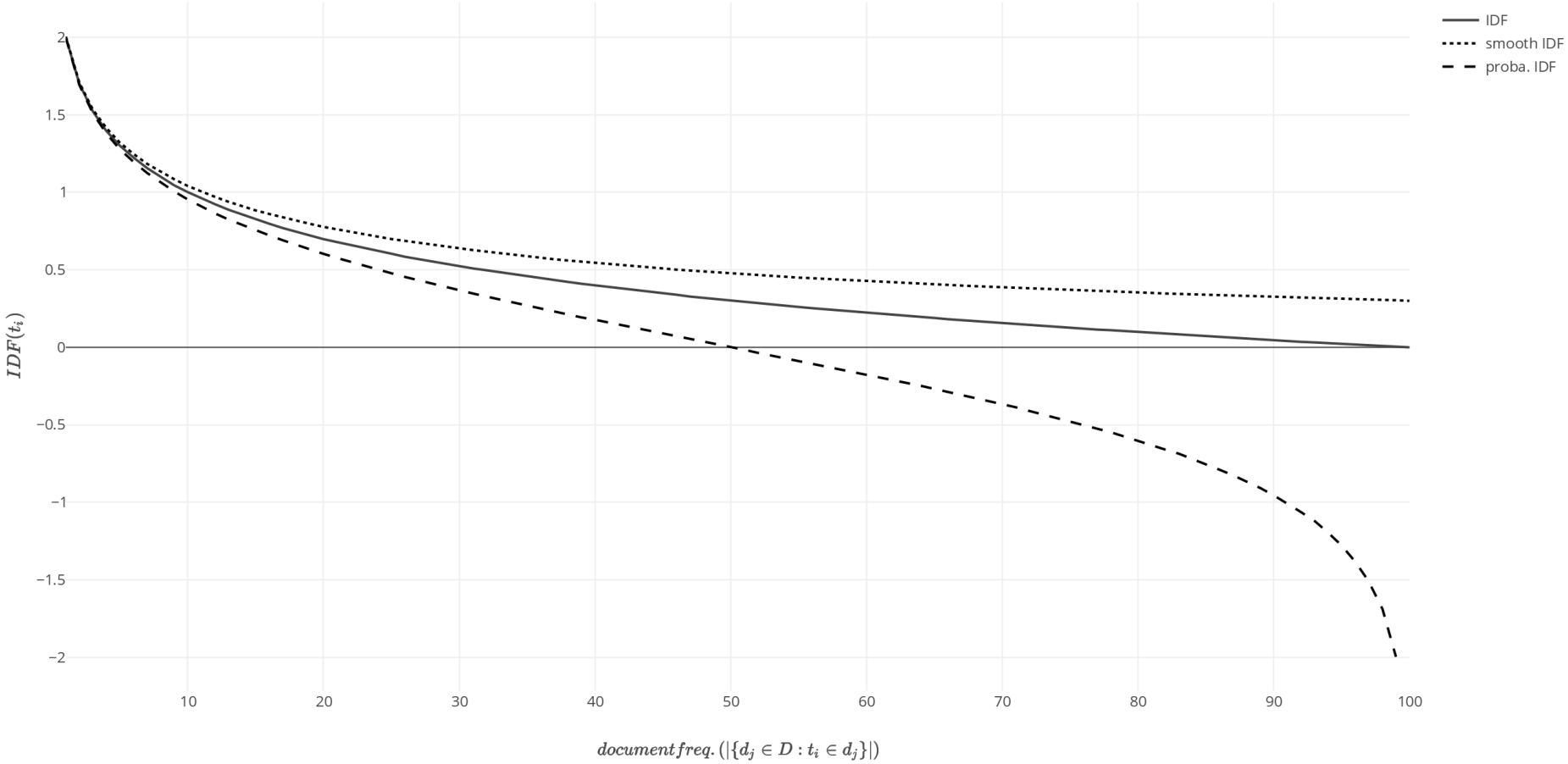
Weighting scheme	tf weight
Binary	0, 1
Raw count	$f_{t,d}$
Log normalization	$\log(1 + f_{t,d})$
Double normalization 0.5	$0.5 + 0.5 \frac{f_{t,d}}{\max_{t' \in d} f_{t',d}}$
Double normalization K $K \in [0,1]$	$K + (1 - K) \frac{f_{t,d}}{\max_{t' \in d} f_{t',d}}$

IDF: Variants

- There are also many ways to define IDF.

Weighting scheme	idf weight ($n_t = \lvert\{d \in D : t \in d\}\rvert$)
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

Image credit: Wikipedia

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
$f_{t,d} \cdot \log\left(\frac{N}{n_t}\right)$	$\left(0.5 + 0.5\frac{f_{t,q}}{\max\limits_{t' \in q} f_{t',q}} f_{t,d}\right) \cdot \log\left(\frac{N}{n_t}\right)$
$\log(1 + f_{t,d})$	$\log\left(1 + \frac{N}{n_t}\right)$
$(1 + f_{t,d}) \cdot \log\left(\frac{N}{n_t}\right)$	$(1 + f_{t,d}) \cdot \log\left(\frac{N}{n_t}\right)$

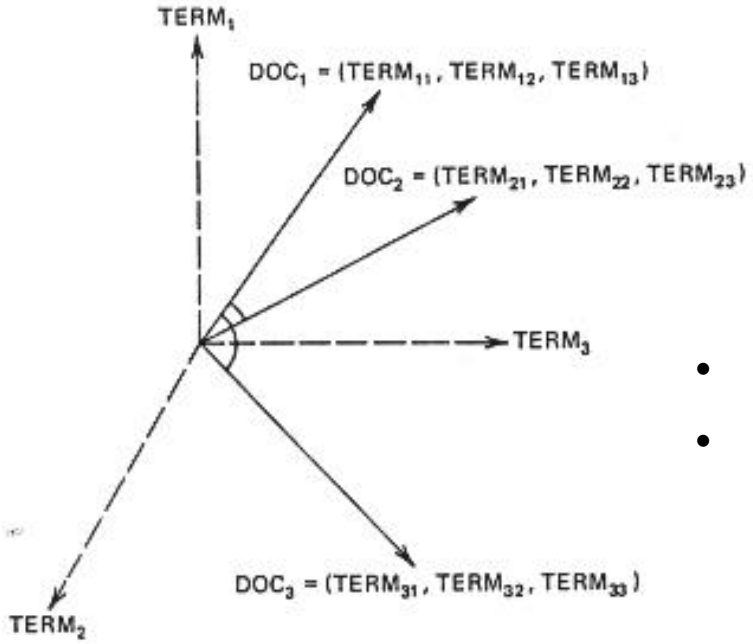
TF-IDF: Calculate the similarity

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

- Cosine similarity: $cosine(\mathbf{d}, \mathbf{q}) = \frac{\langle \mathbf{d} \cdot \mathbf{q} \rangle}{\|\mathbf{d}\| \times \|\mathbf{q}\|}$

document query

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



- $|V|$ 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

$$\text{sim(Doc 2, Q)} = 0.967 > \text{sim(Doc 4, Q)} = 0.315$$
$$> \text{sim(Doc 1, Q)} = 0.299 > \text{sim(Doc 3, Q)} = 0.102$$

Quiz 01: Construct TF-IDF vectors

1. 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

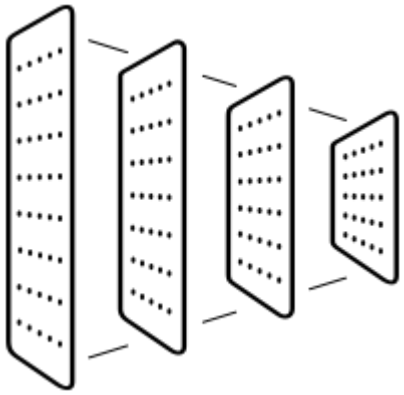
   d3: delivery of silver arrived in a silver truck

   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.

2. Discover how to perform the above task in **scikit-learn**.

Word Embeddings



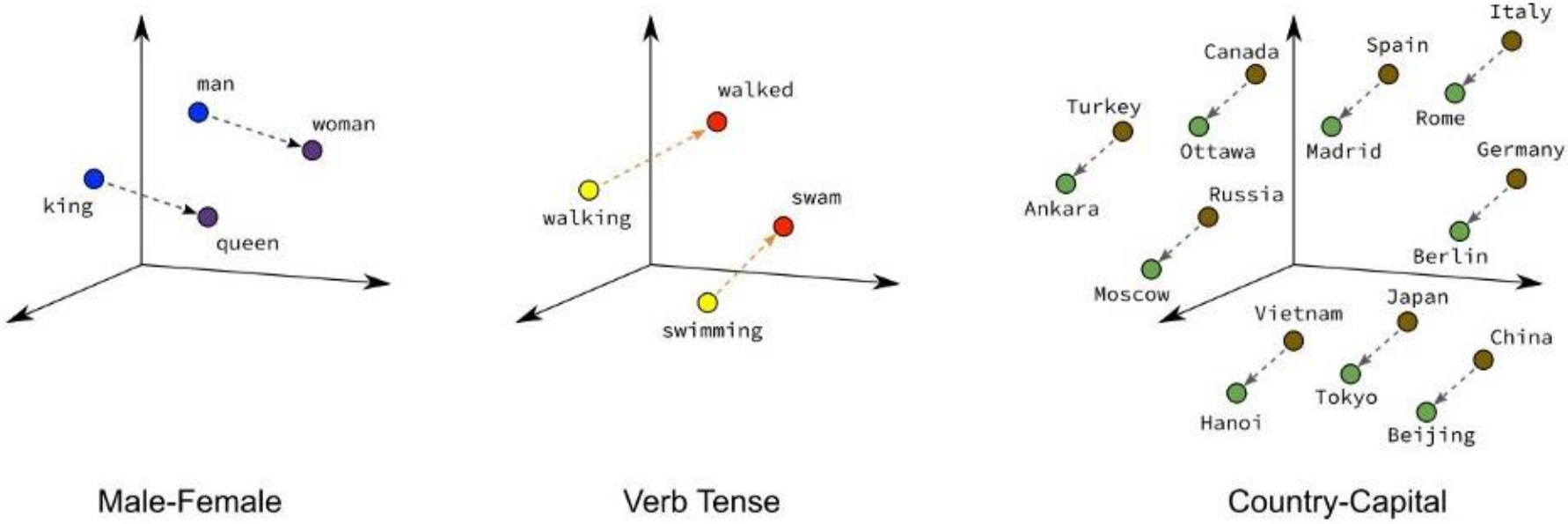
Michael Heck, 2019. “The World of Vectors. Dialog Systems and Machine Learning”. Heinrich-Heine-Universität Düsseldorf.

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

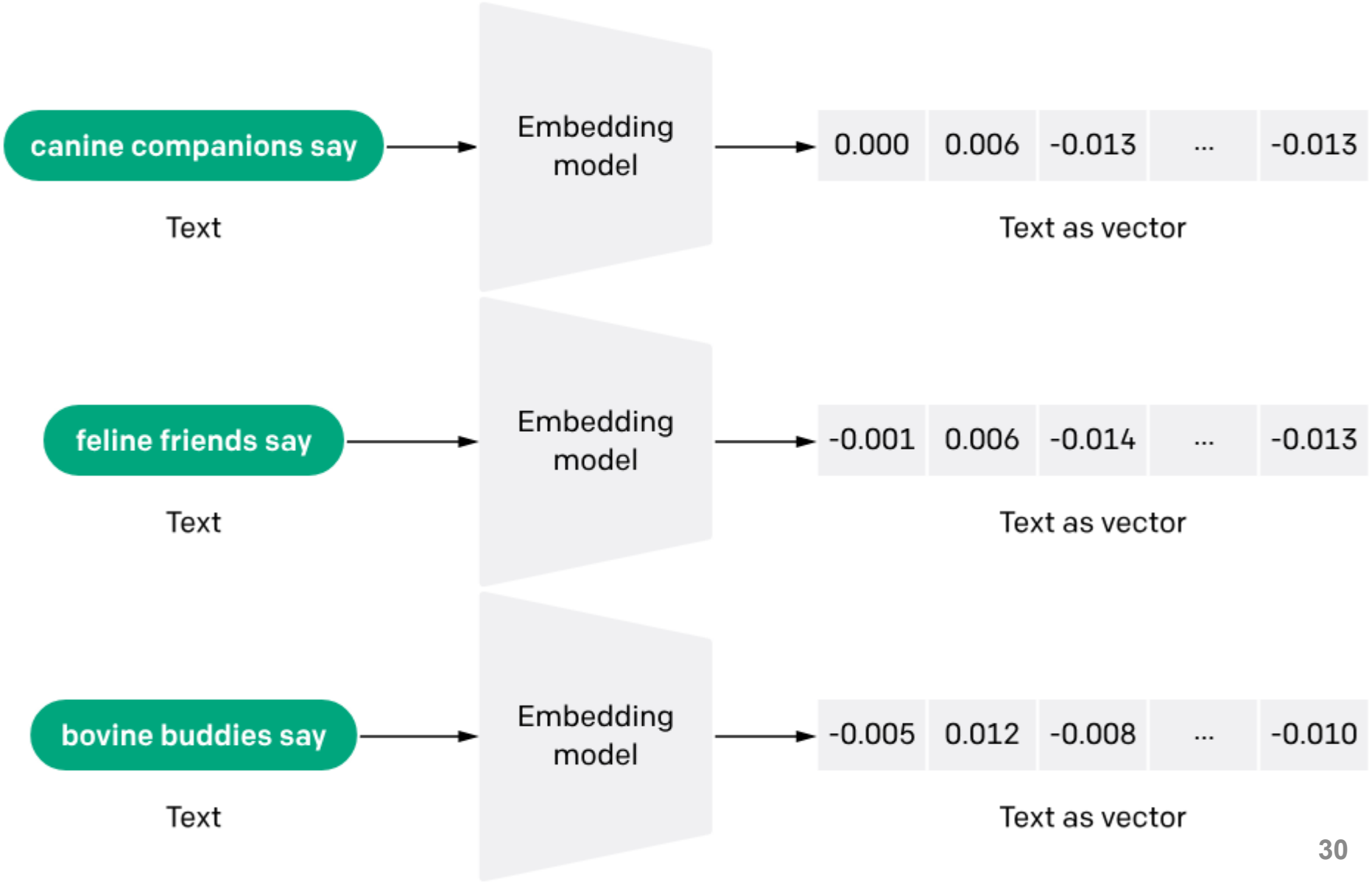
- **Key idea:** The embeddings of similar words are closer in the vector space than those of different ones.



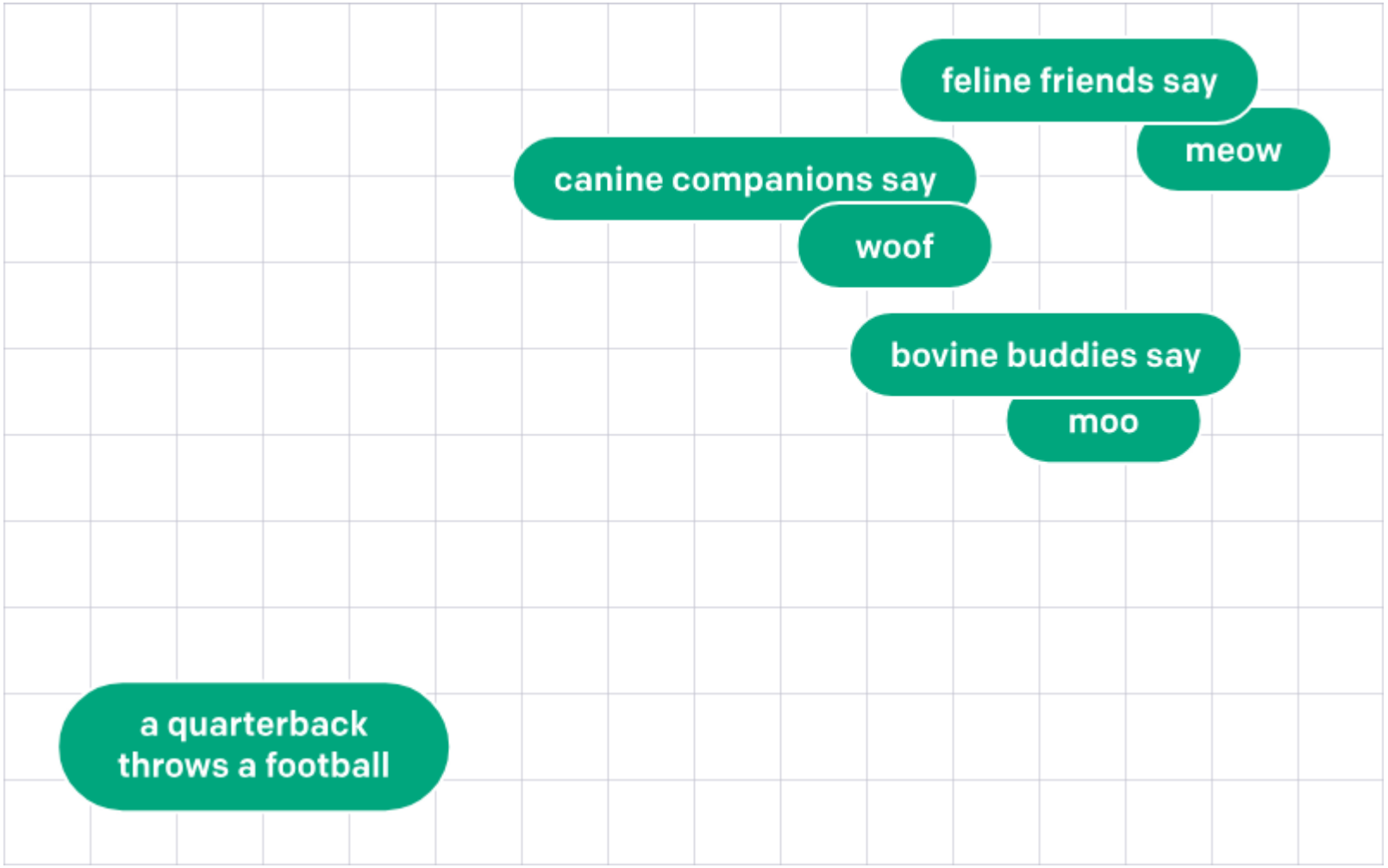
Male-Female Verb Tense Country-Capital

Image credit: Google Developer

Word embeddings: An example



Word embeddings: An example



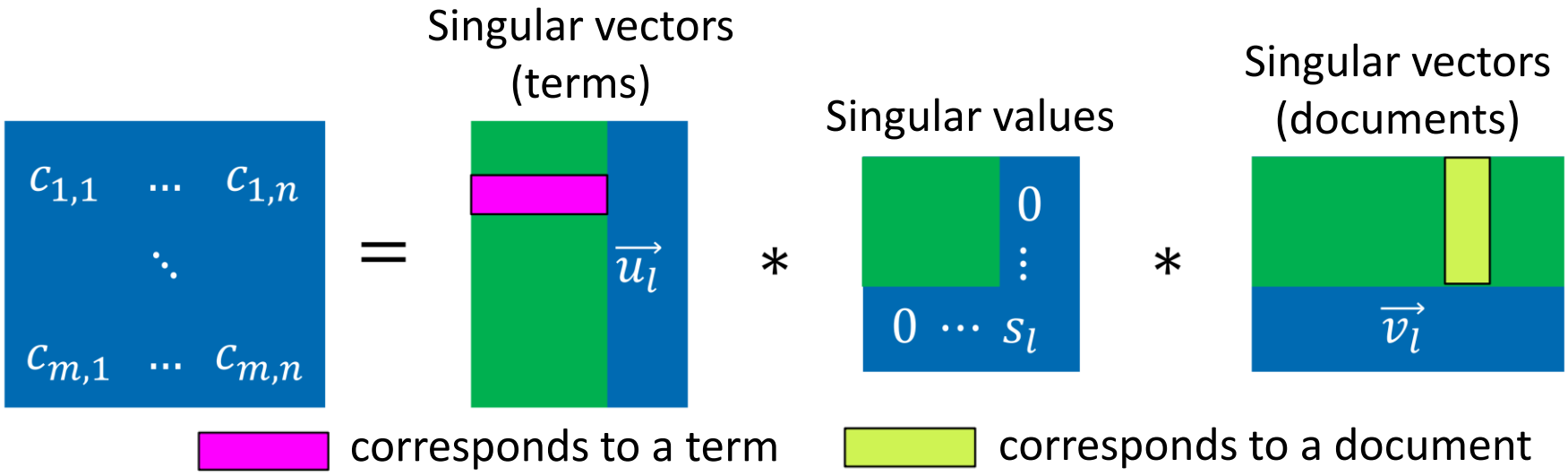
Word embeddings: Applications

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

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.

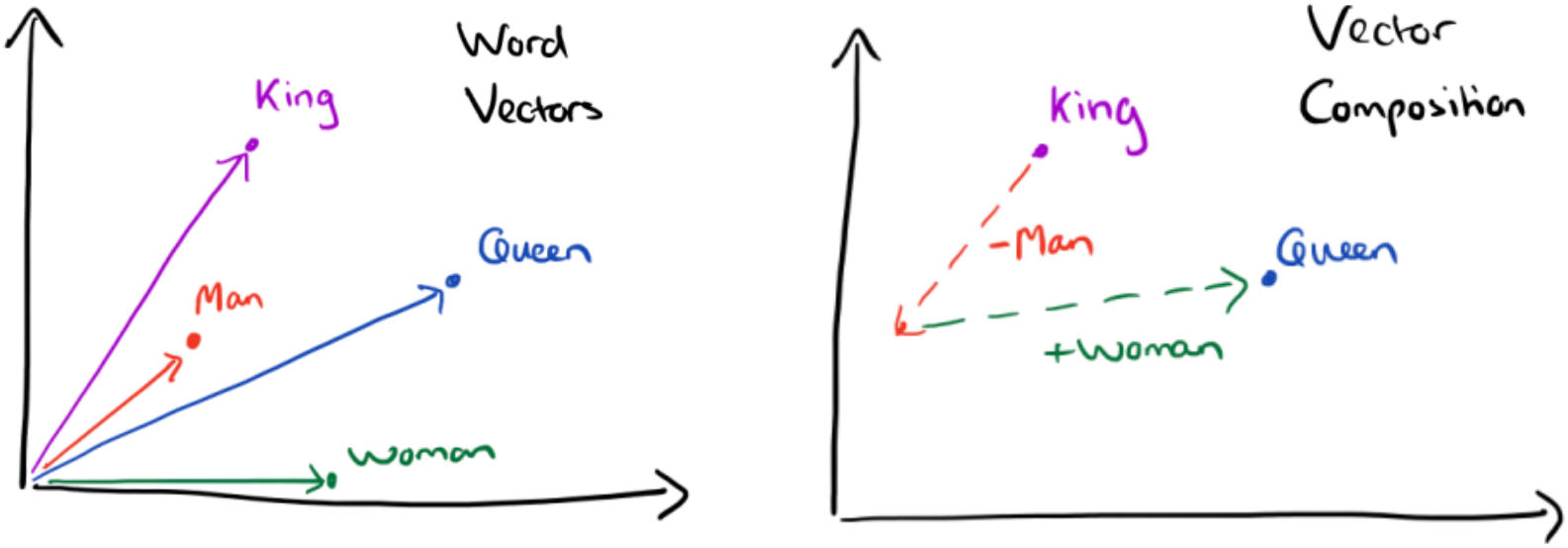
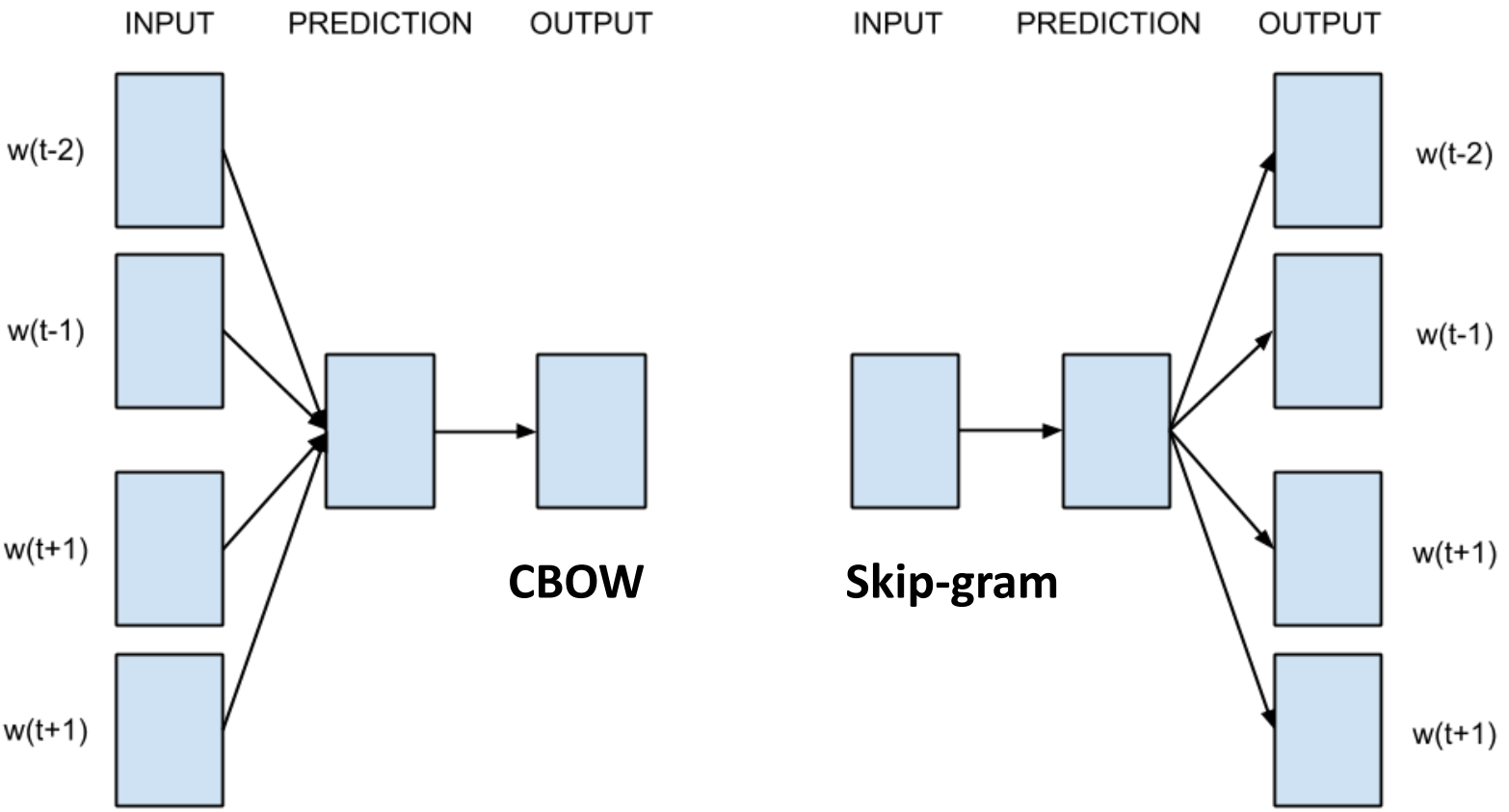


Image credit: KDnuggets

word2vec (2013)

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



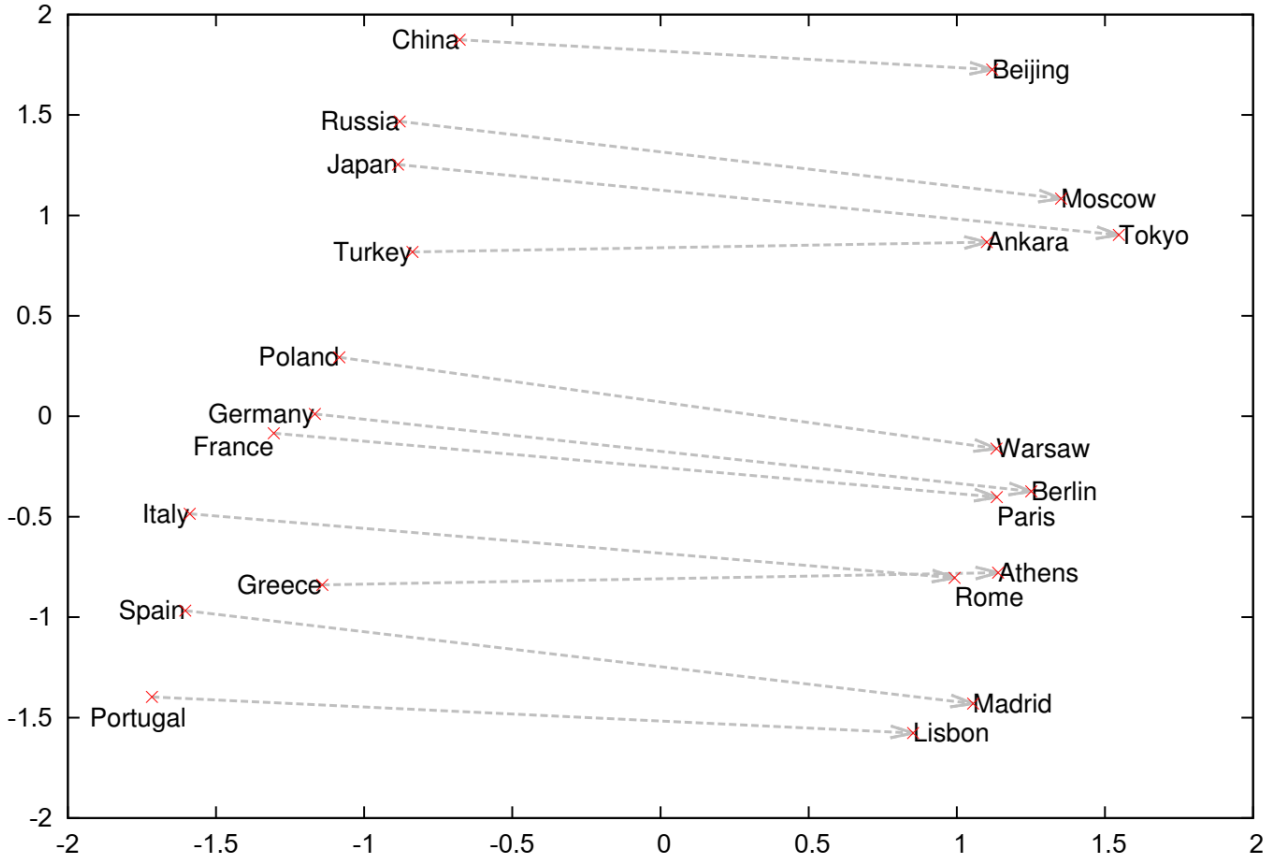
word2vec: Word pair relationships

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, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

word2vec (2013)



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

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems 26 (2013).

word2vec: Vector compositionality

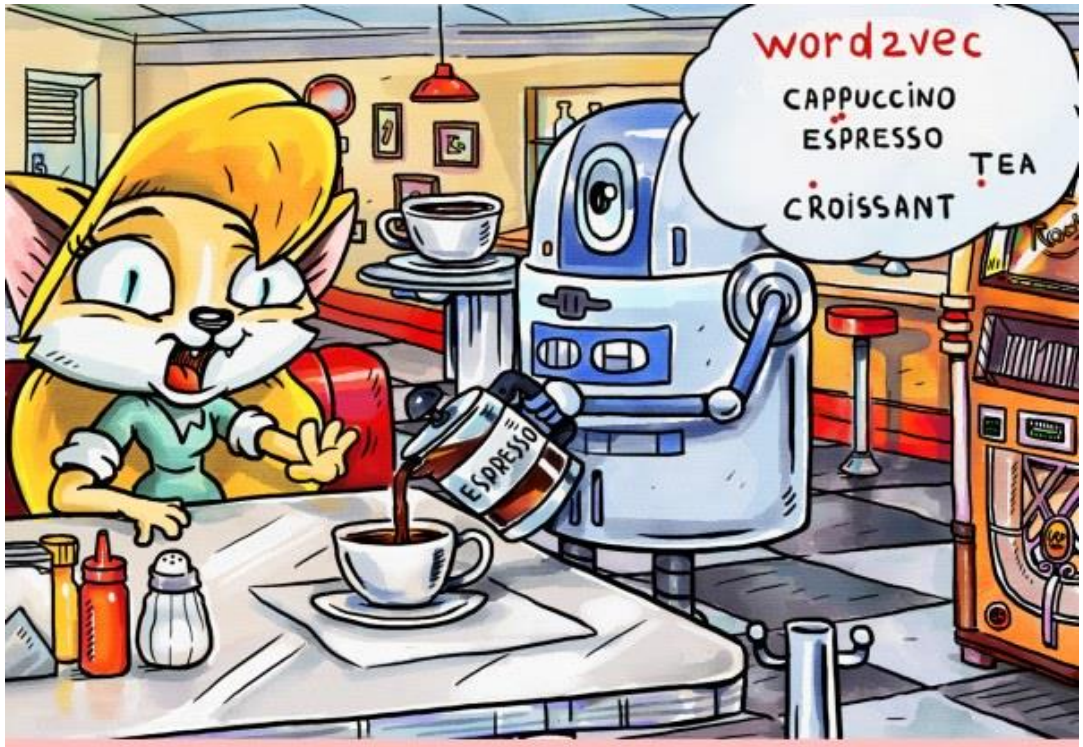
Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

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, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems 26 (2013).

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, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532-1543. 2014.

GloVe (2014)

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

co-occurrence probabilities

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

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.

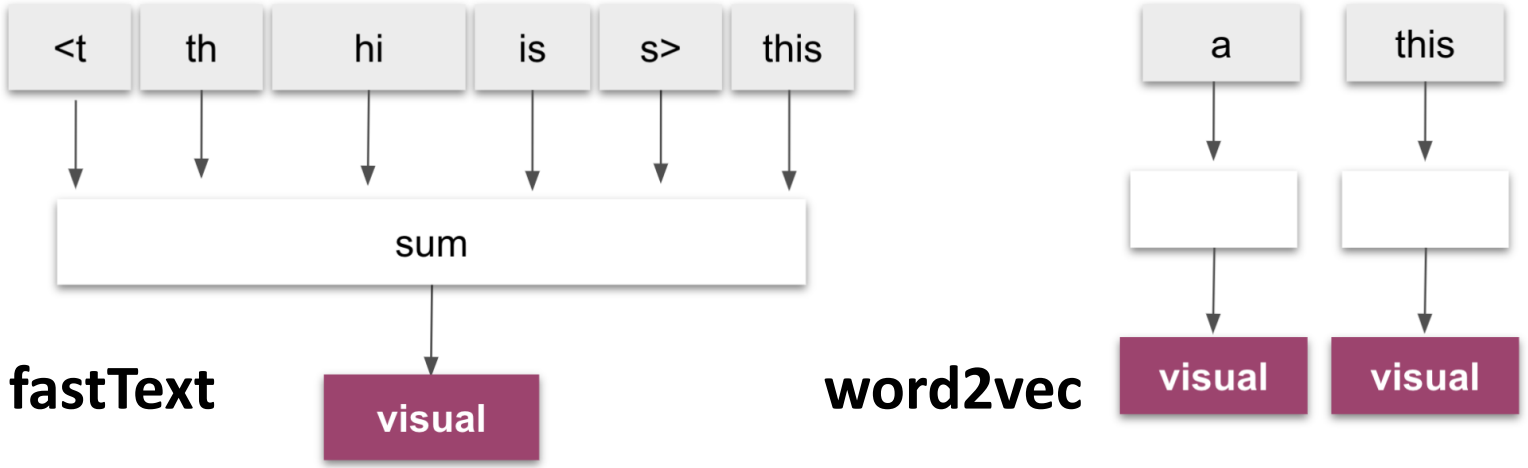
Latent semantic analysis (LSA)	
Capture the meaning	Difficult to interpret

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.

word2vec:

typo

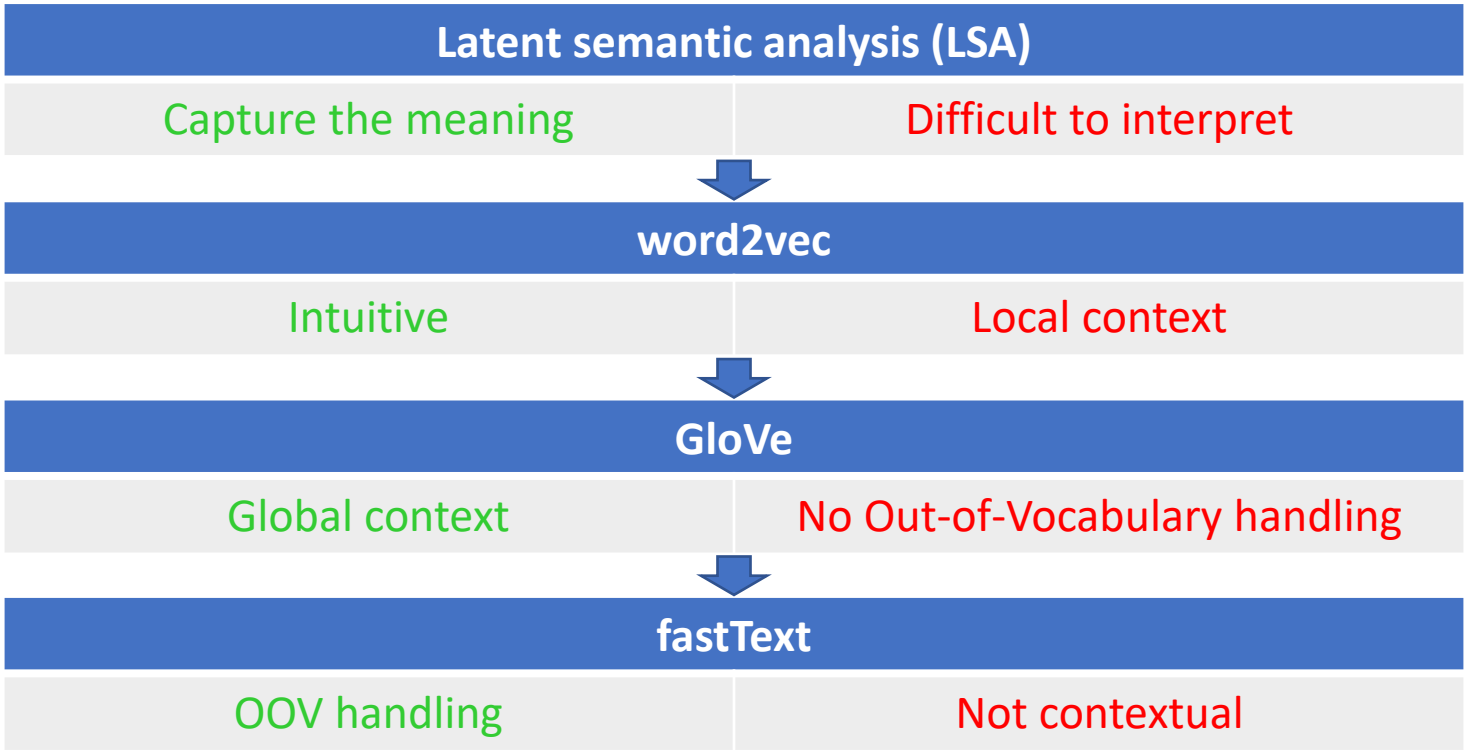
```
Query word? accomodation
    sunnhordland 0.775057
    accomodations 0.769206
    administrational 0.753011
    laponian 0.752274
    ammenities 0.750805
    dachas 0.75026
    vuosaari 0.74172
    hostelling 0.739995
    greenbelts 0.733975
    asserbo 0.732465
```

fastText:

```
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
```

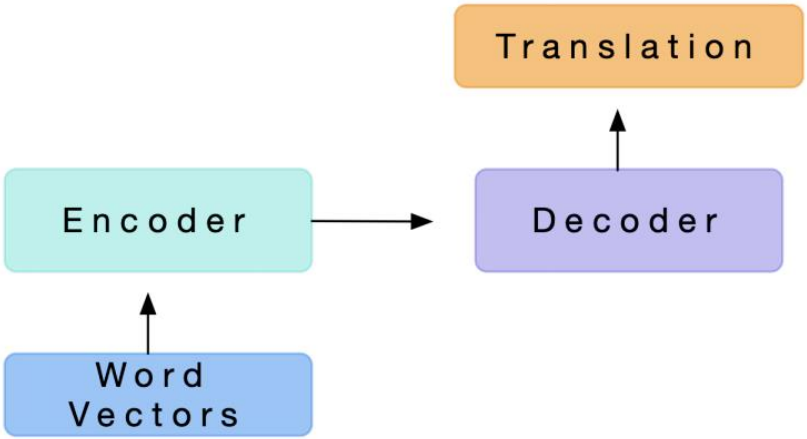
Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching word vectors with subword information." Transactions of the association for computational linguistics 5 (2017): 135-146.

Image credit: fastText

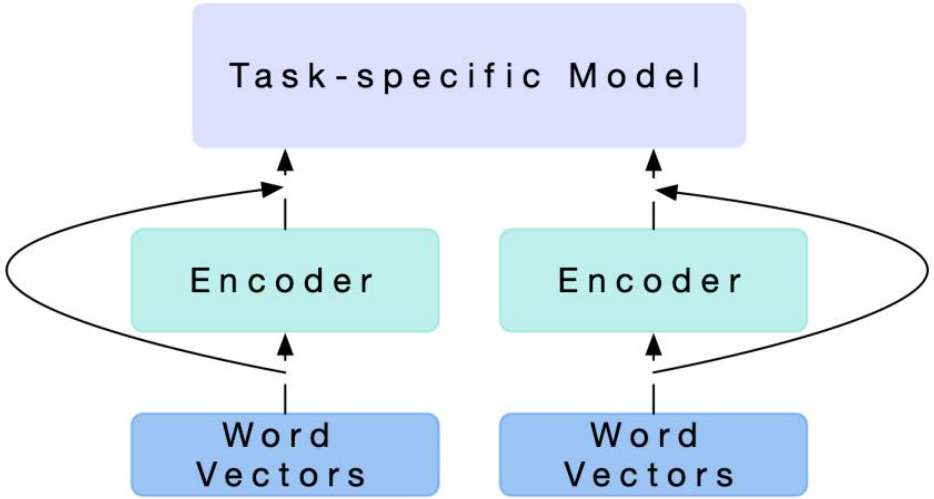

Latent semantic analysis (LSA)
Capture the meaning
Difficult to interpret
word2vec
Intuitive
Local context
GloVe
Global context
No Out-of-Vocabulary handling
fastText
OOV handling
Not contextual

CoVe (2017)

- Contextualized word Vectors (CoVe) are attained by using 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

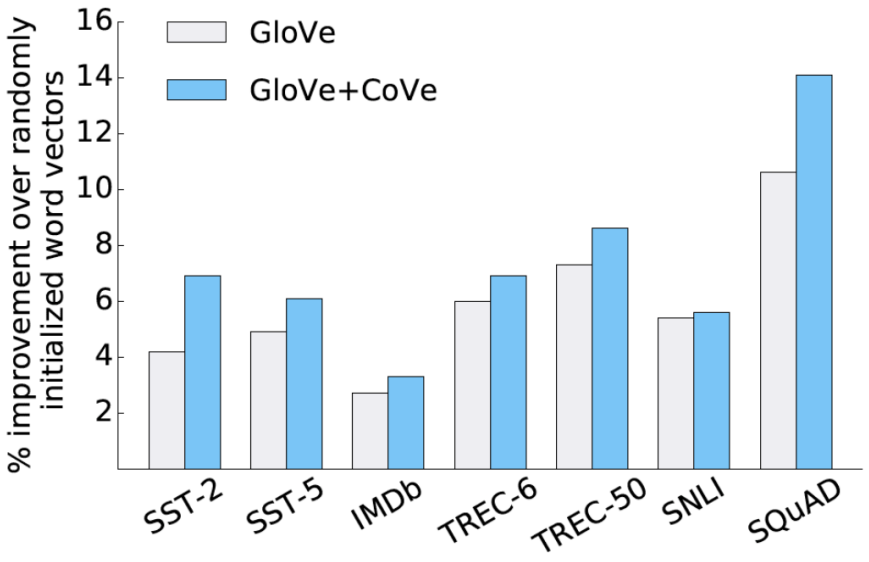
$$\mathbf{CoVe}(\boldsymbol{w}) = \mathbf{MT-LSTM}\big(\mathbf{GloVe}(\boldsymbol{w})\big)$$

sequence of words

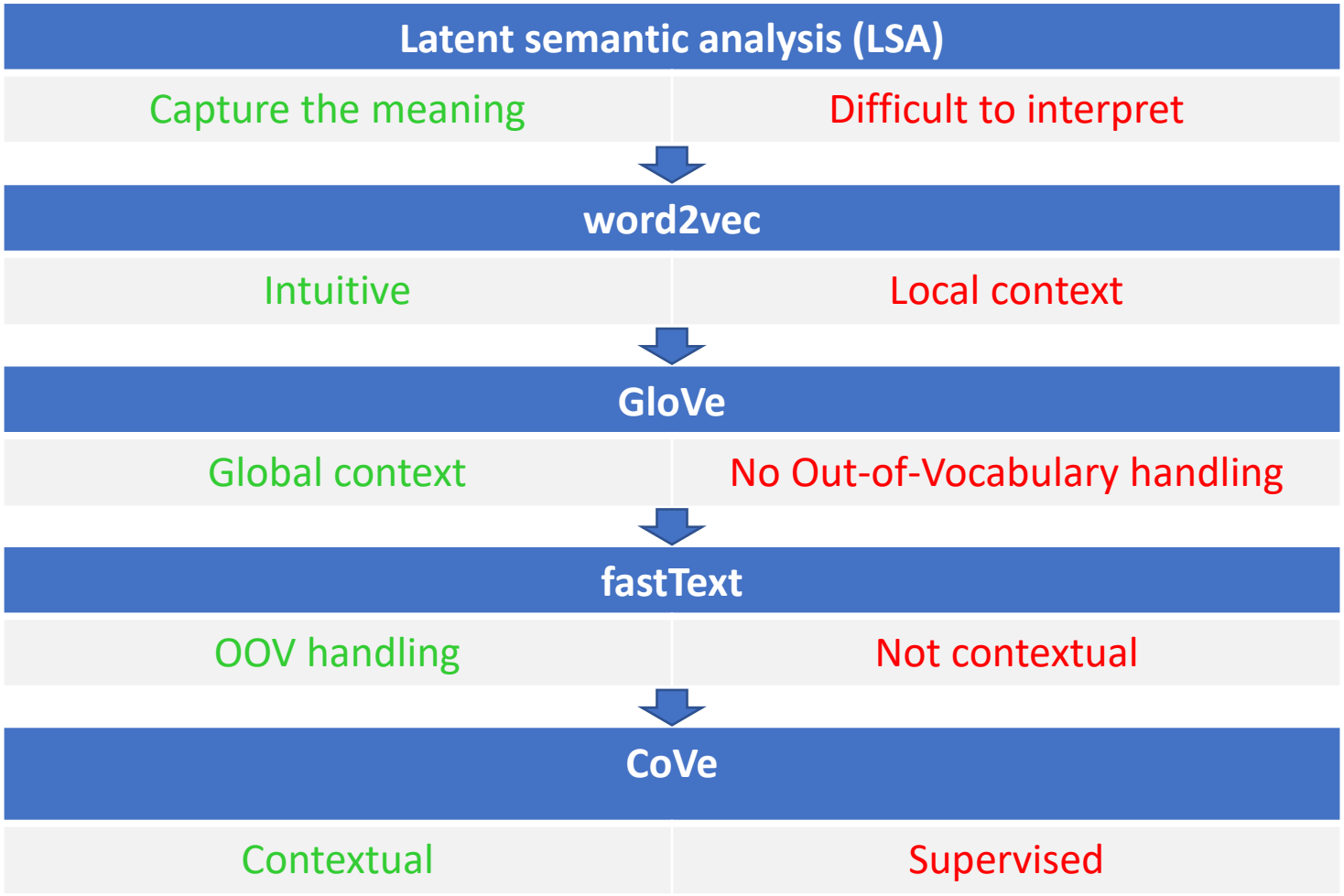
sequence of GloVes

- It serves as input for downstream tasks.

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



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

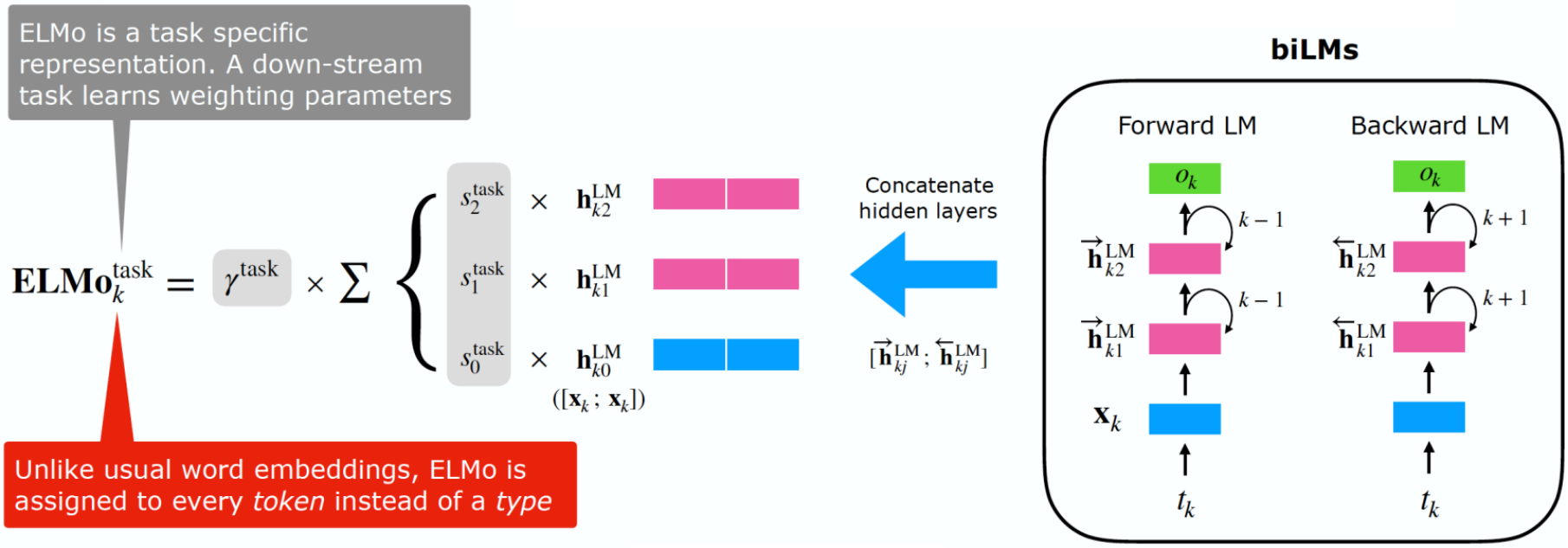
ELMo (2018)

- Embeddings from Language Model (ELMo)) learns word embeddings via building bidirectional language models.
- 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): Build bidirectional LMs

- We can build a bidirectional language model (biLM) with two LSTMs predicting the next words in both directions.

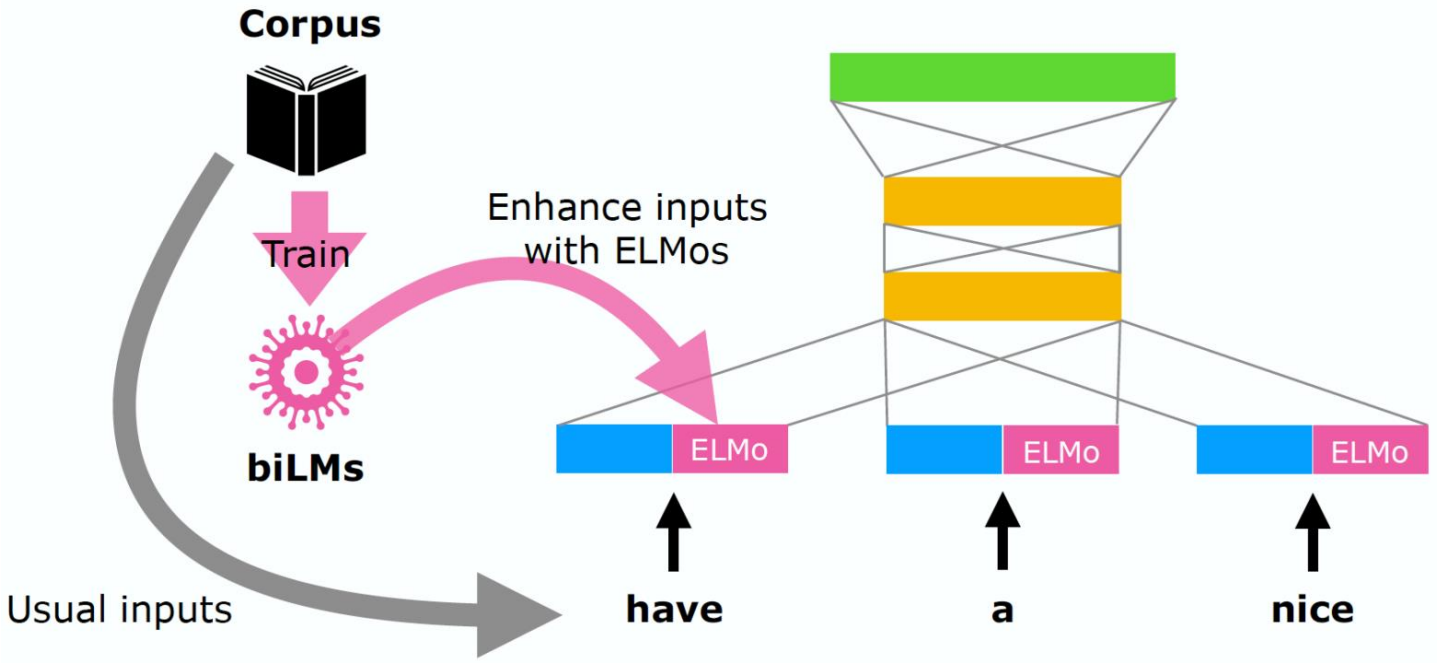


Yada, Shuntaro. A Review of Deep Contextualized Word Representations. 2018.

Image credit: SlideShare

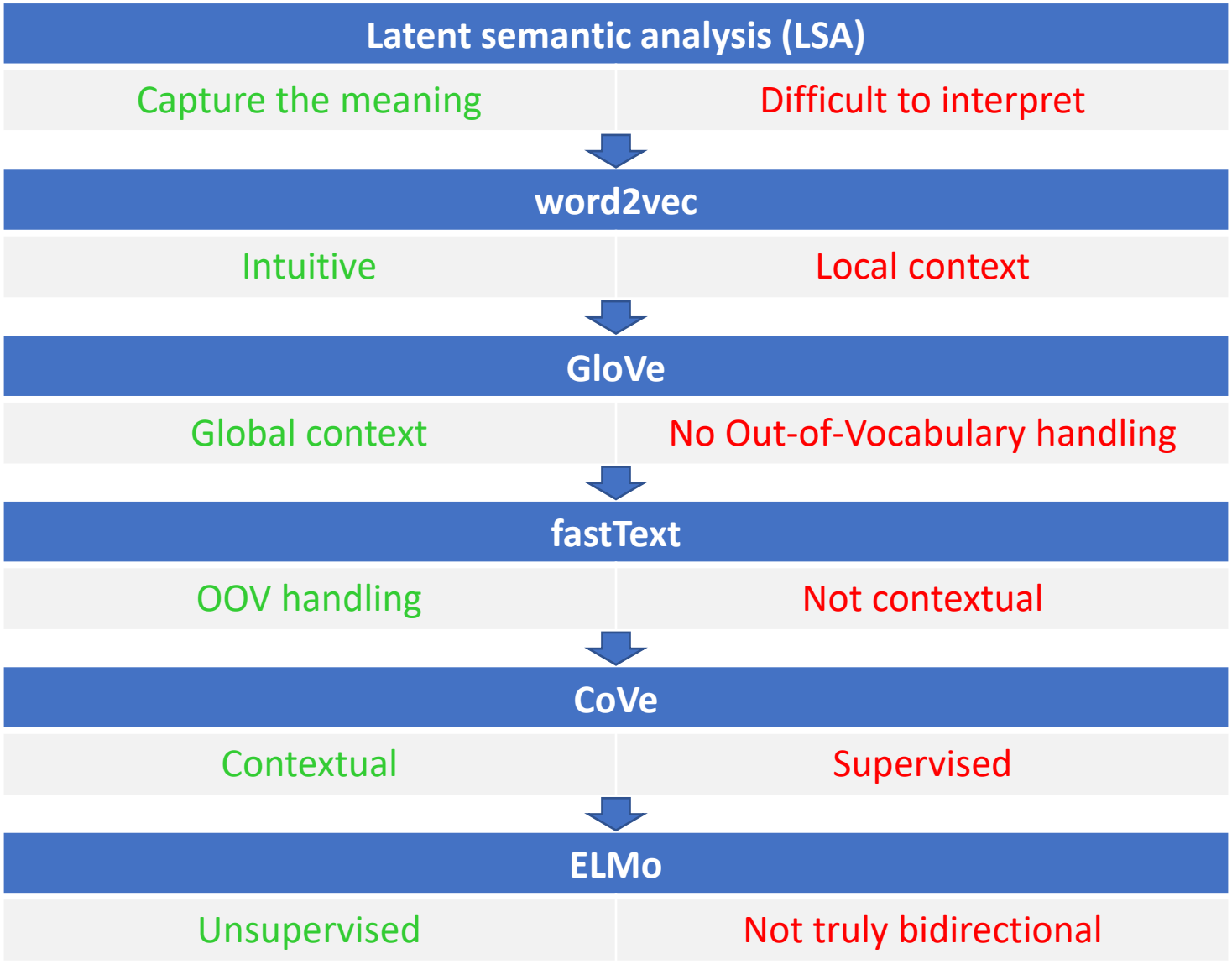
ELMo (2018): Usage

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



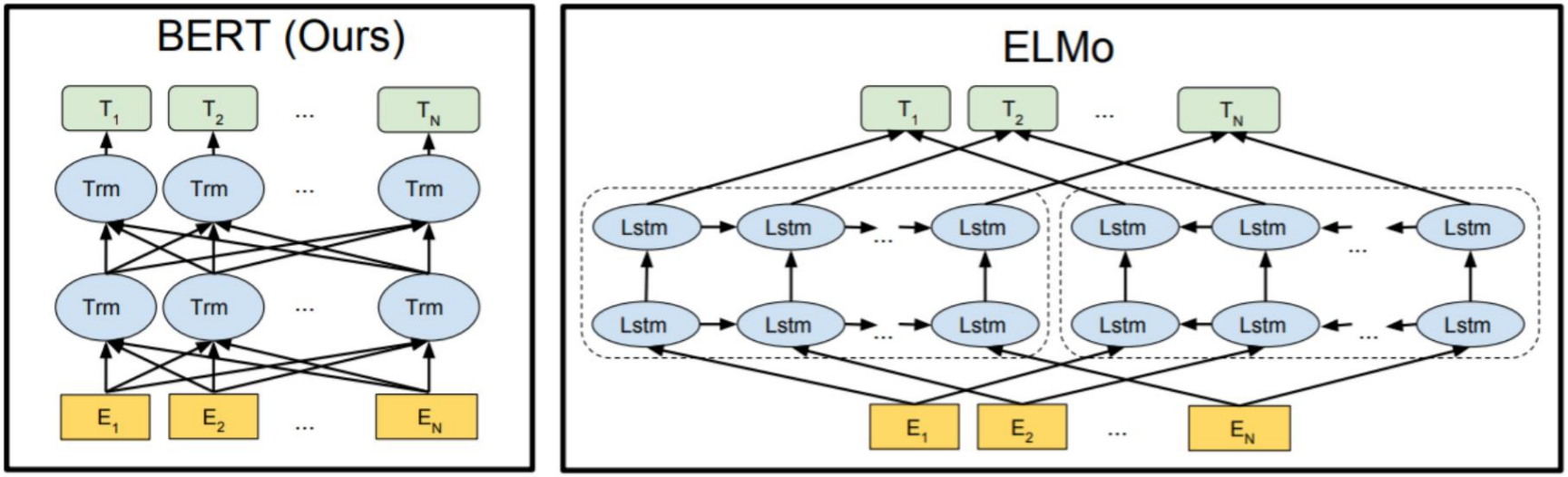
Yada, Shuntaro. A Review of Deep Contextualized Word Representations. 2018.

Image credit: SlideShare


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)

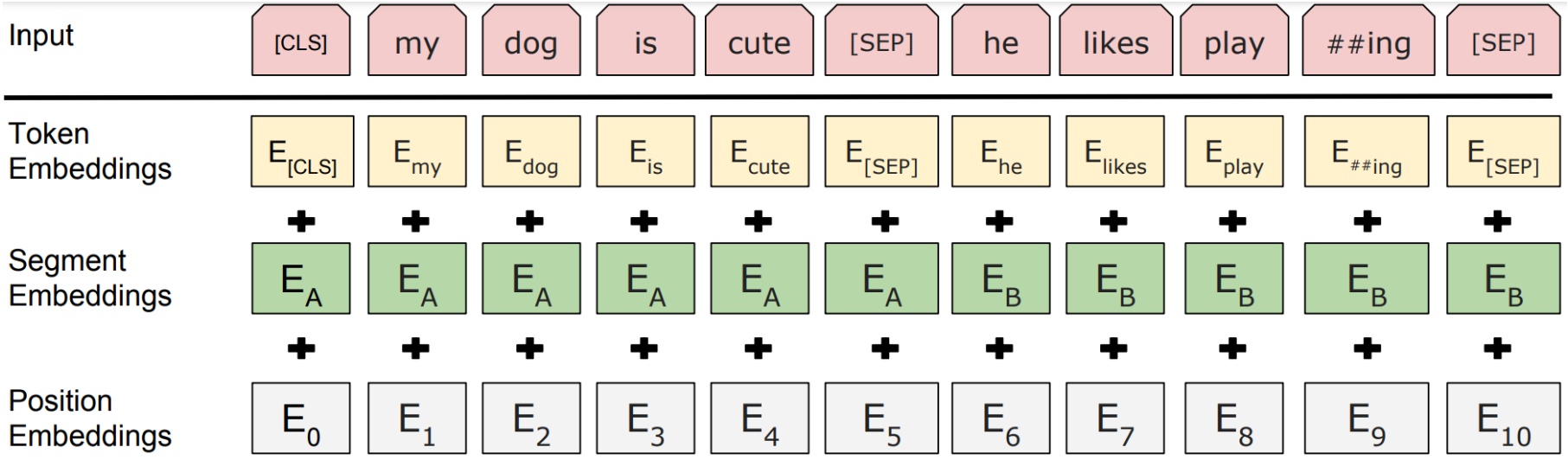
- **B**i-directional **E**ncoder **R**epresentations from **T**ransformers
- The representations are jointly conditioned on left and right context in all layers.



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.



Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), pp. 4171-4186. 2019.

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 → my dog is [MASK]
  - 10% of the time: Replace the word with a random word
    - E.g., my dog is hairy → my dog is apple
  - 10% of the time: Keep the word unchanged
    - E.g., my dog is hairy → 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

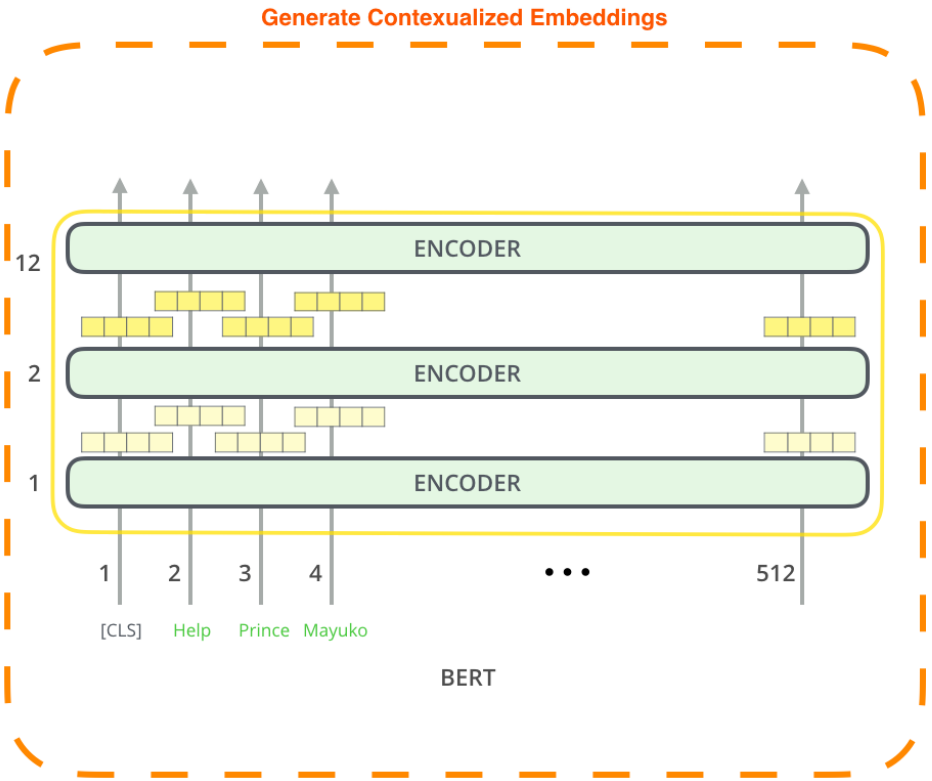
  $\rightarrow$ Label: NotNext

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.



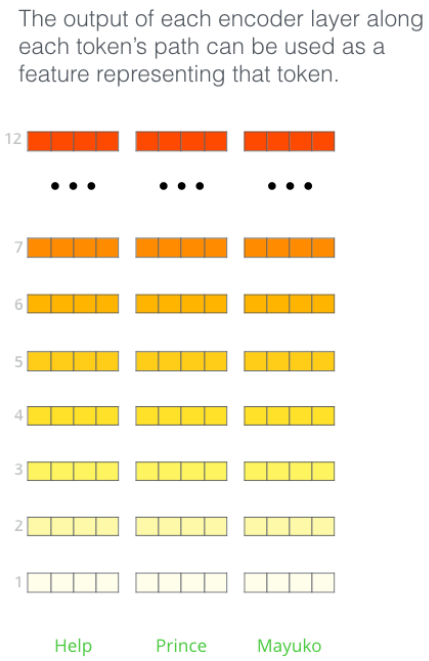
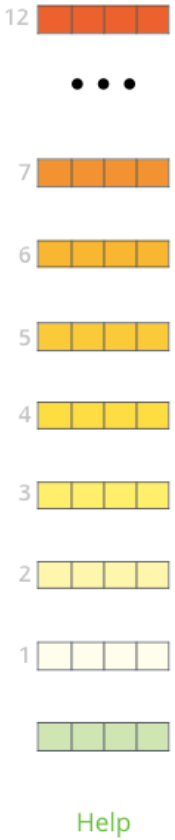


Image credit: jalammar.github.io

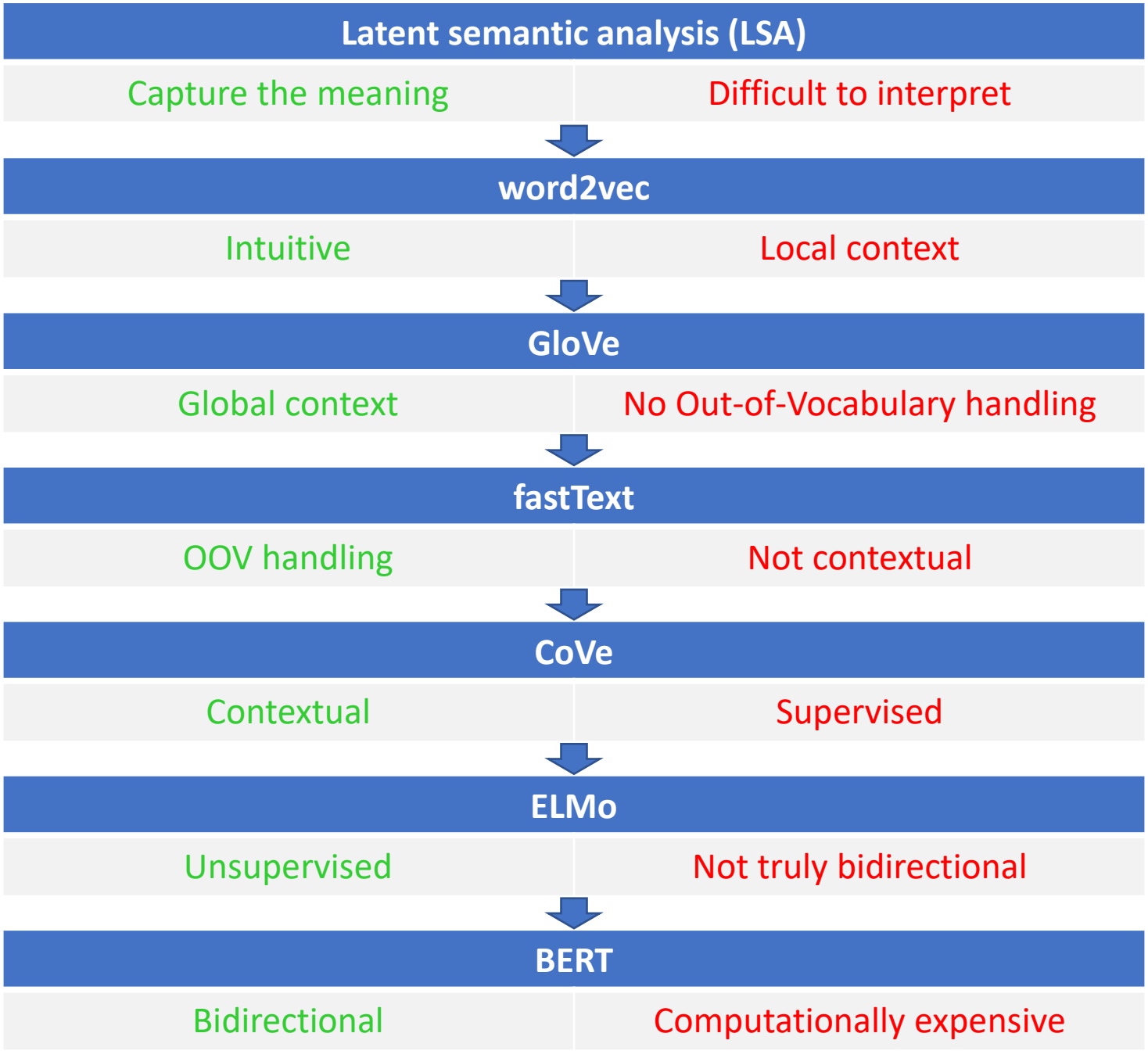
BERT (2018): Feature extraction

What is the best contextualized embedding for "Help" in that context?
For named-entity recognition task CoNLL-2003 NER



		Dev F1 Score
First Layer	Embedding	91.0
Last Hidden Layer	12	94.9
Sum All 12 Layers	12 + ... + 2 + 1 =	95.5
Second-to-Last Hidden Layer	11	95.6
Sum Last Four Hidden	12 + 11 + 10 + 9 =	95.9
Concat Last Four Hidden	9 10 11 12	96.1

task-dependent


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

Similarity metric for text

Bilingual Evaluation Understudy (BLUE)

- BLEU measures k-grams overlap between the output text and reference text.
- A $BLEU_N$ score consider k-grams with $k = 1,2, \ldots N$.

$$\boldsymbol{BLEU_N} = \underset{\boldsymbol{Penalty}}{\boldsymbol{Brevity}} \times \underset{\boldsymbol{Precision\ Score}}{\boldsymbol{Weighted\ Geometric\ Mean}}$$

- Geometric mean precision score is a weighted geometric mean of the clipped k-gram precisions for $k = 1,2, \ldots N$.
- The brevity penalty favors longer output texts.

.

- Reference: Natural Language Processing: Bleu Score

Other metrics

- **METEOR** focuses on exact, stemmed, synonym, and paraphrase matches.
- **CIDEr** (Consensus-based Image Description Evaluation) measures consensus in multiple reference texts.
- **SPICE** (Semantic Propositional Image Caption Evaluation) assesses semantic meaning.
- **Perplexity** measures how well a language model predicts a sample.
- **ROUGE** (Recall-oriented Understudy for Gisting Evaluation):
  - ROUGE-1, ROUGE-2: Unigram and bigram overlap.
  - ROUGE-L: Longest common subsequence overlap.
- **BERTScore** uses embeddings from BERT model to compare semantic similarity.

Quiz 02: Large language models

1. Distinguish these terminologies: language models (LMs) and large language models (LLMs).
2. Classify the following models to either LM or LLM:
   - PaLM
   - GPT-4
   - LLaMA
   - Skip-gram
   - BERT
   - word2Vec
3. Discover the above models: Open-source? Downloadable?

References

- Michael Heck, 2019. The World of Vectors. Dialog Systems and Machine Learning. Heinrich-Heine-Universität Düsseldorf.
- Wikipedia: tf-idf (link)
- GloVe: Global Vectors for Word Representation (link)
- A Visual Guide to FastText Word Embeddings (link)
- The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) (link)


...the end.