How do I embed a sequence of words? From context-free to contextual embeddings

Fabio Fumarola

Milan, July 18 2019

prometeia

- Language and Machine Learning
- Distributional Semantics
- Context-free word embeddings:
  - Count based
  - Context based
- Contextual embeddings
- Final Considerations & Conclusions

prometeia

- **Language and Machine Learning**
- Distributional Semantics
- Context-free word embeddings:
  - Count based
  - Context based
- Contextual embeddings
- Final Considerations & Conclusions

Human language: what is special

- It is distinct from other known form of communications since it is ***compositional***

- It allows speakers to express/reference ideas/opinions in sentences by **combining** subjects, verbs and objects

- *Compositionality* gives human language an endless ***capacity for generating new sentences*** as speakers recombine set of words taken from a vocabulary

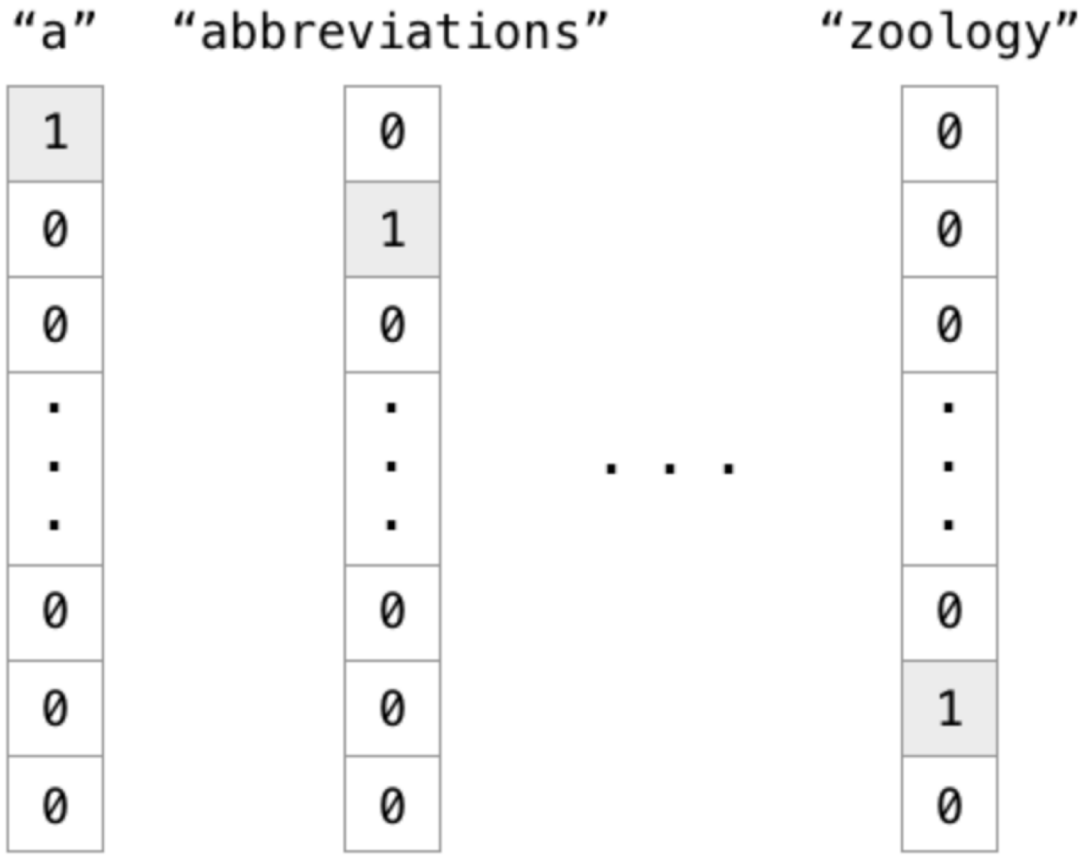
- For instance, with just 25 words it is possible to generate over 15K distinct sentences

Pagel M. Q&A: What is human language, when did it evolve and why should we care?. *BMC Biol*. 2017 Jul 24
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5525259/

Human language in Machine Learning 1/2

- A machine learning model to understand and process natural language needs to transform it into numeric values

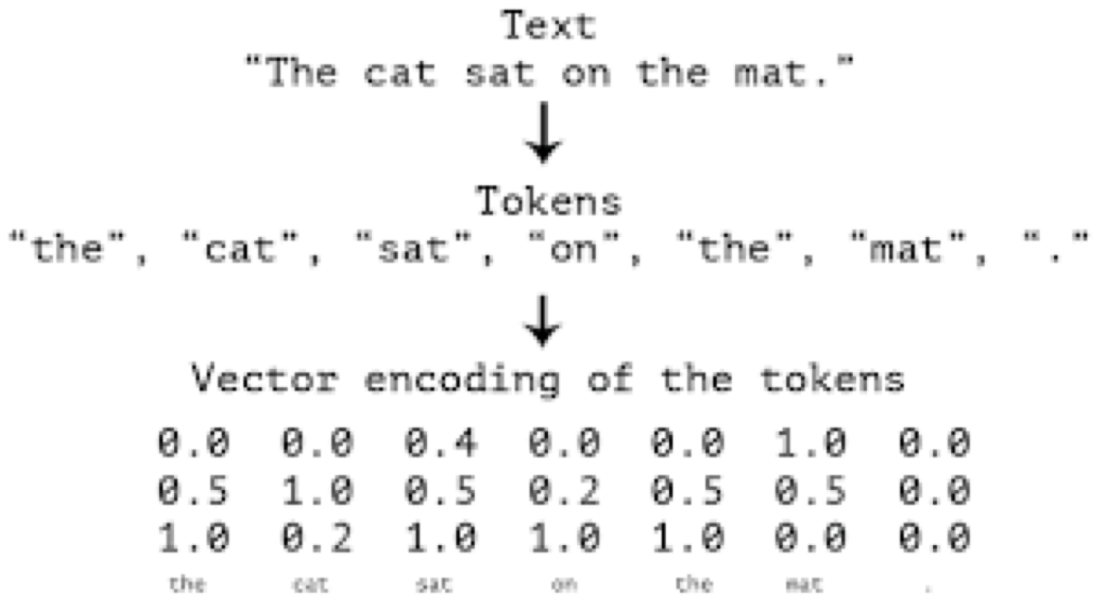
- **Simple Solution:** to do one-hot encoding for each word in the vocabulary
- **Complexity:** one-hot encoding is impractical computationally for large vocabulary
- And …



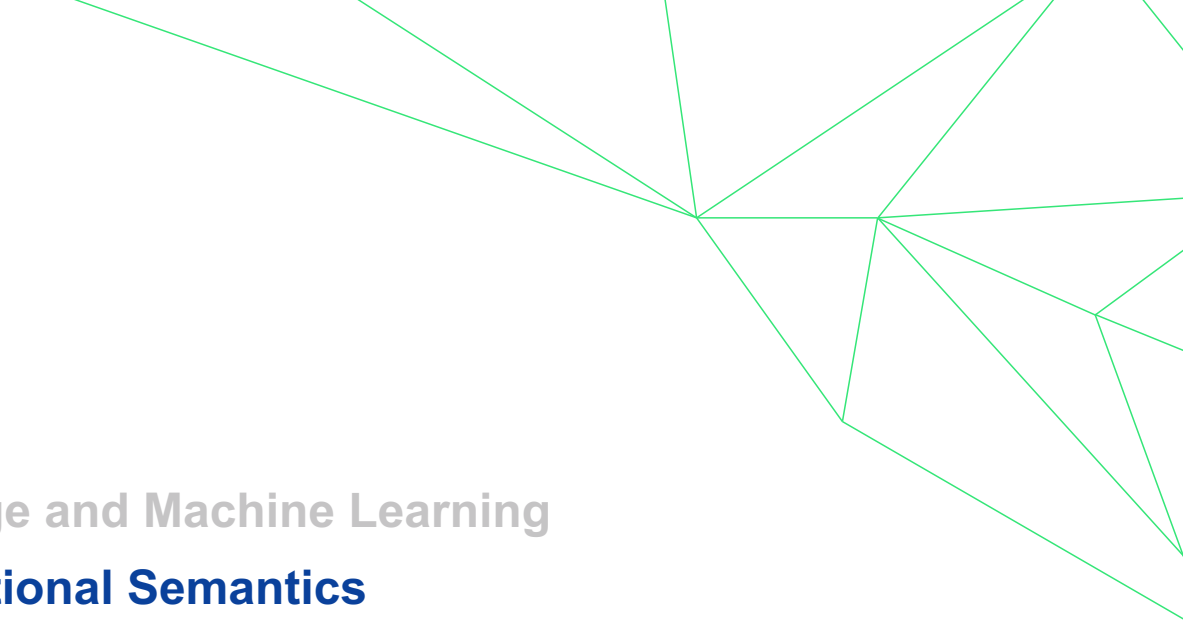
prometeia

Human language in Machine Learning 2/2

- **Word Embeddings** represent text in vectors with much more lower and denser dimensions
- Are based on the idea that contextual information alone constitutes a viable representation of linguistic items (in contrast with Chomsky)
- Open questions: When, Why, How, Limitations ?



prometeia



- Language and Machine Learning
- **Distributional Semantics**
- Context-free word embeddings
  - Count based
  - Context based
- Contextual embeddings
- Final Considerations & Conclusions

A brief History of Word Embeddings 1/3

Theoretical roots are in structural linguistics and language philosophy:

- In 1950: Harris Z., Firth J. and Wittgenstein L. published works that used feature representations to quantify semantic-similarity (hand-crafted features)

- In 1957: Osgood C., Suci G. and Tannenbaum P. with *“The measurement of meaning”* (Urbana, IL press) found three recurring attitudes of people to evaluate words and phrase: ‘good-bad’, ’strong-weak’, ‘active-passive’

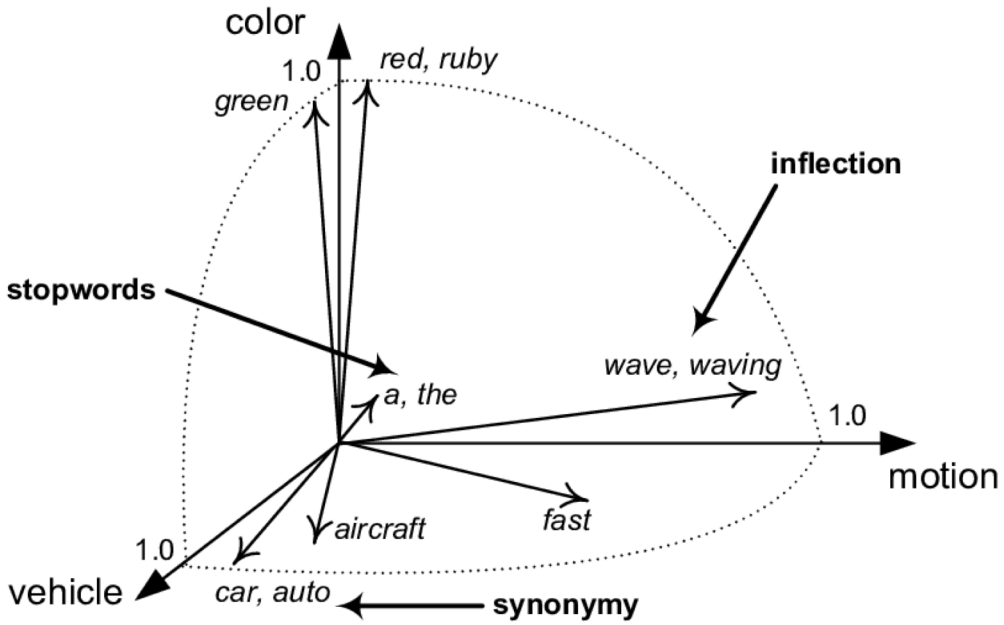
- From 1990s: many methods to generate contextual features were developed

prometeia

A brief History of Word Embeddings 2/3

All the works are based on the idea that:

”*Semantically similar words tends to have similar contextual distributions*” (Miller and Charles, 1991)



Geometric techniques
to measure similarity

prometeia

Miller, G. A., & Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and Cognitive Processes, 6*(1), 1-28.

Word Embeddings: a note

There are also many other alternative terms in use

- from the very general **distributed representation**
- to the more specific **semantic vector space** or simply **word space**.
- For consistency, I will adhere to current practice and use the term word embedding in this presentation.

Distributional Semantic Model: Two Main Approaches

1. **Count-based:** based on matrix factorization of a word co-occurrence matrix (unsupervised)

2. **Context-based** (or predictive models) **:** given a local context, we want to predict the target word and in parallel to learn an efficient vector word representation (supervised)

prometeia

Baroni, Dinu and Kruszewsky, Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors ACL 2014 (https://www.aclweb.org/anthology/P14-1023)

prometeia

- Language and Machine Learning
- Distributional Semantics
- **Context-free word embeddings:**
  - **Count based**
  - Context based
- Contextual embeddings
- Final Considerations & Conclusions

Count-based methods

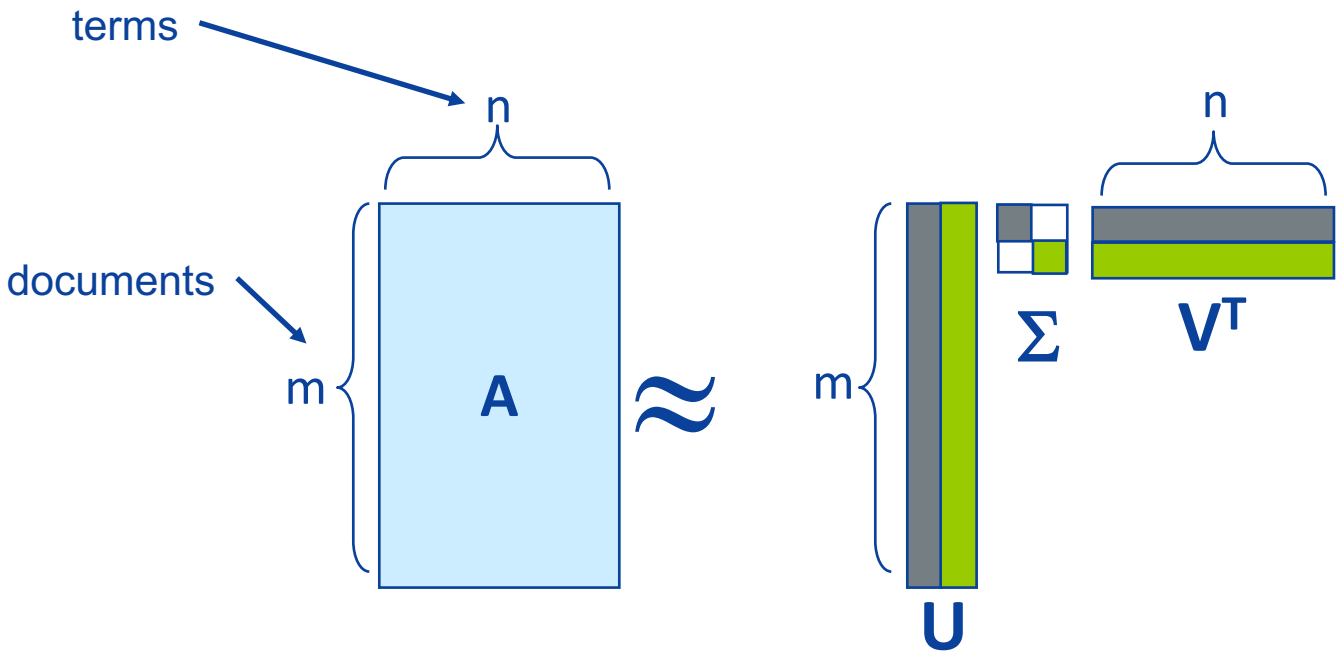
Starting from raw co-occurrence counts they apply a function (Matrix Factorization) to map them down to small and dense word vectors:

- **Latent Semantic Analysis/Indexing** (LSA/LSI), develop in the context of IR
- **Topics models**, **Probabilistic LSA** (PLSA) and **Latent Dirichlet Allocation** (LDA)

Matrix Factorization: SVD

$$\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i$$

T



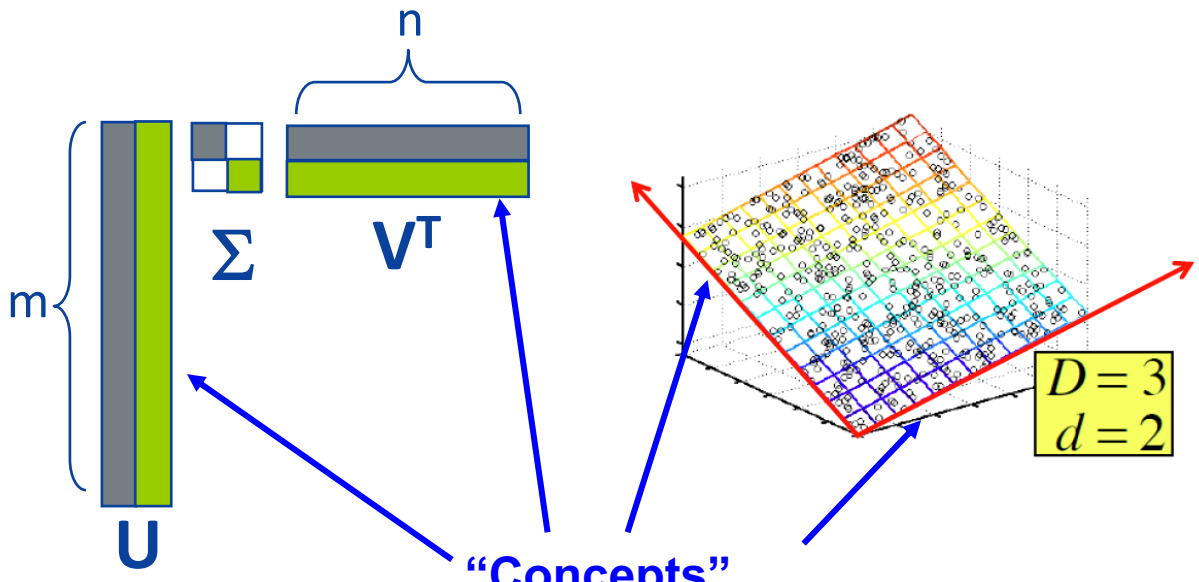
- A: m x n (documents, terms)
- U: m x n (documents concepts)
- **Σ**: r x r diagonal matrix (strength of each ‘concept’)
- V: n x r (terms, concepts)

prometeia

SVD: Users to Movies

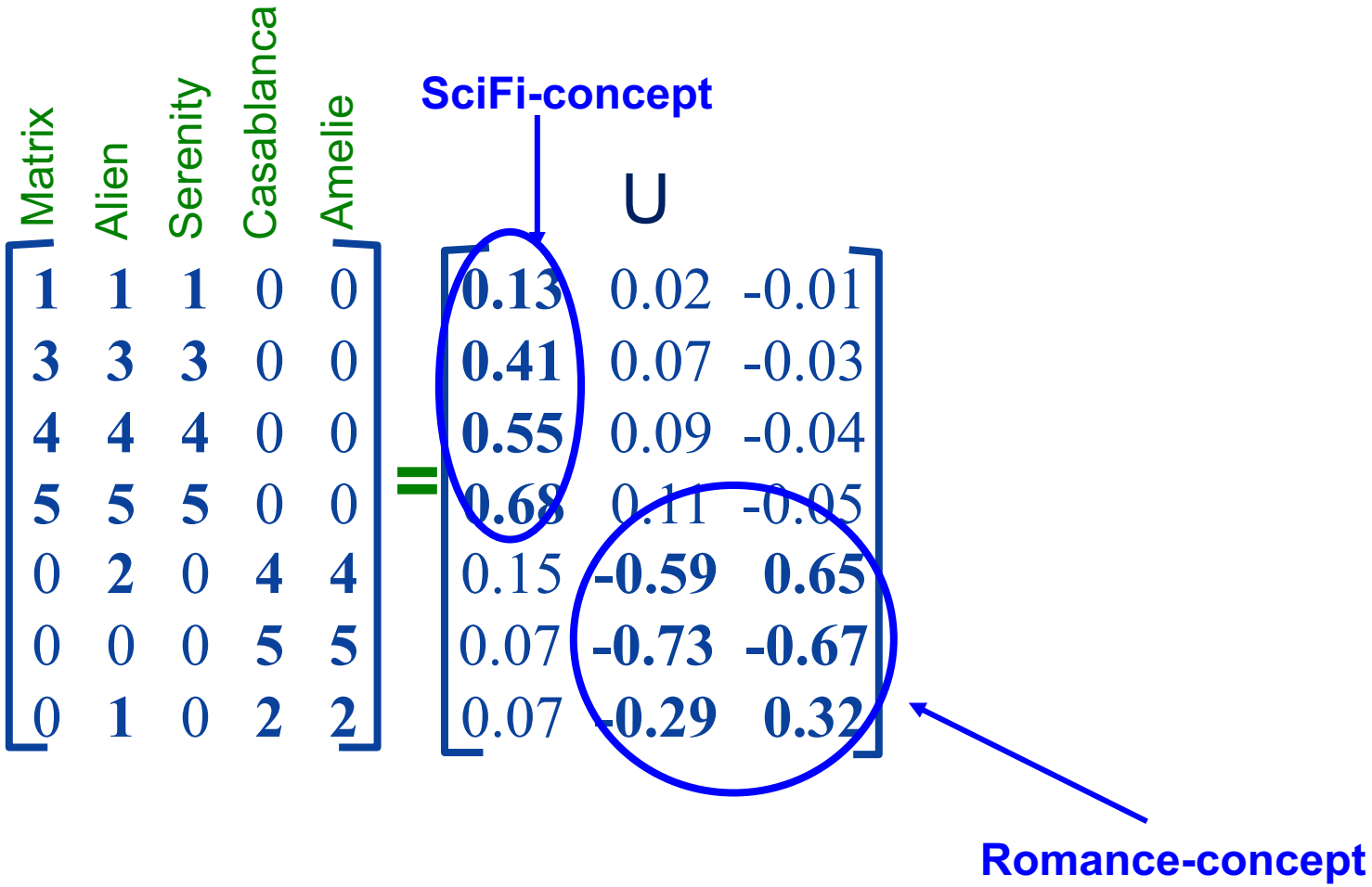
	Matrix	Alien	Serenity	Casablanca	Amelie
	1	1	1	0	0
	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
	0	2	0	4	4
	0	0	0	5	5
	0	1	0	2	2

=



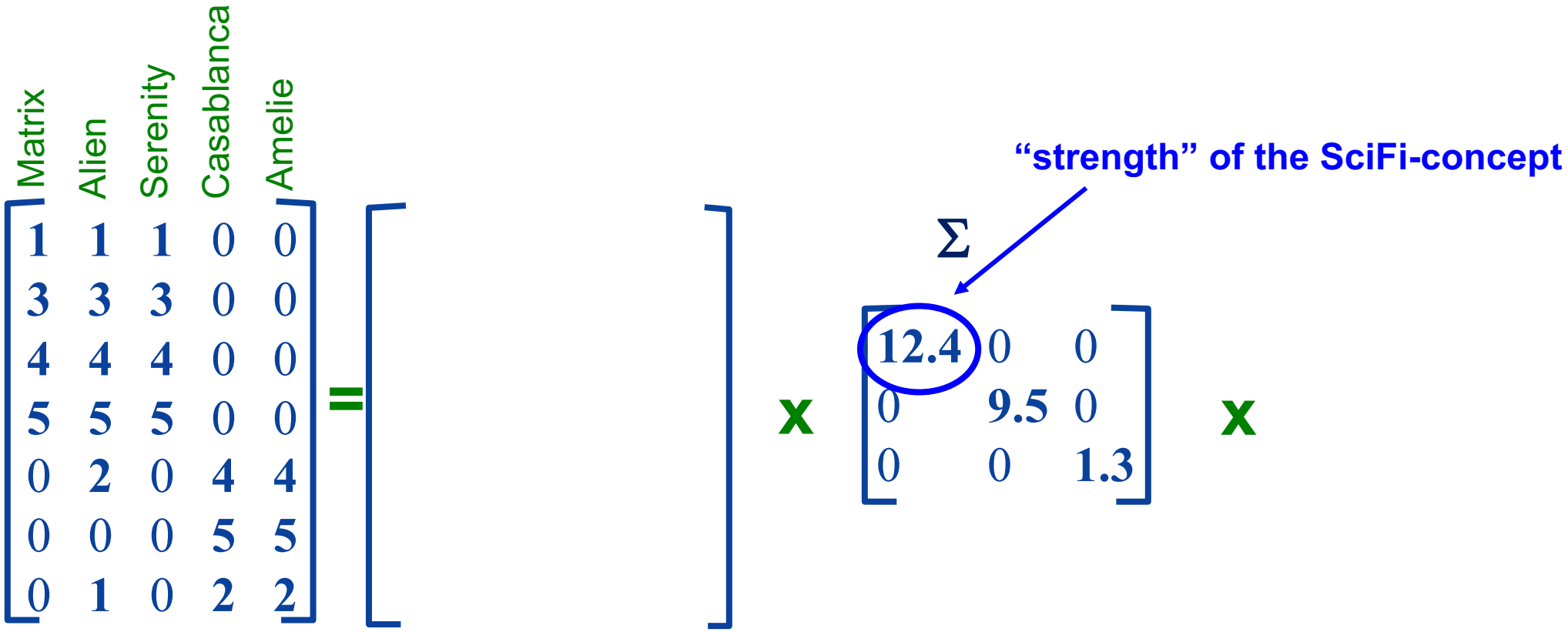
"Concepts"
AKA Latent dimensions
AKA Latent factors

SVD: users to concept similarity matrix



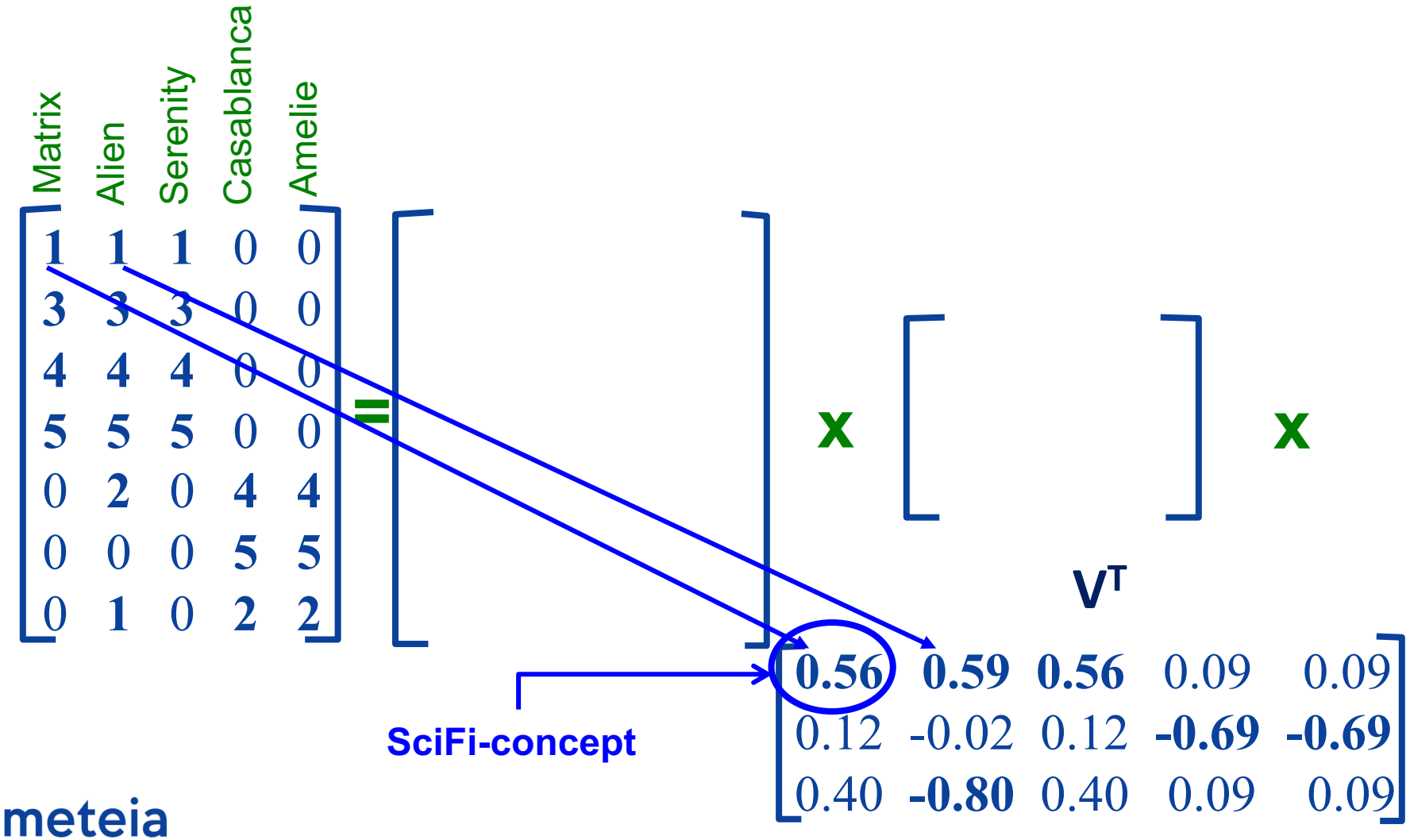
prometeia

SVD: Concepts Strength



prometeia

SVD: Movie to Concept similarity



prometeia

- Language and Machine Learning
- Distributional Semantics
- **Context-free word embeddings:**
  - Count based
  - **Context based**
- Contextual embeddings
- Final Considerations & Conclusions

History

Build models to predict a word given its neighbors

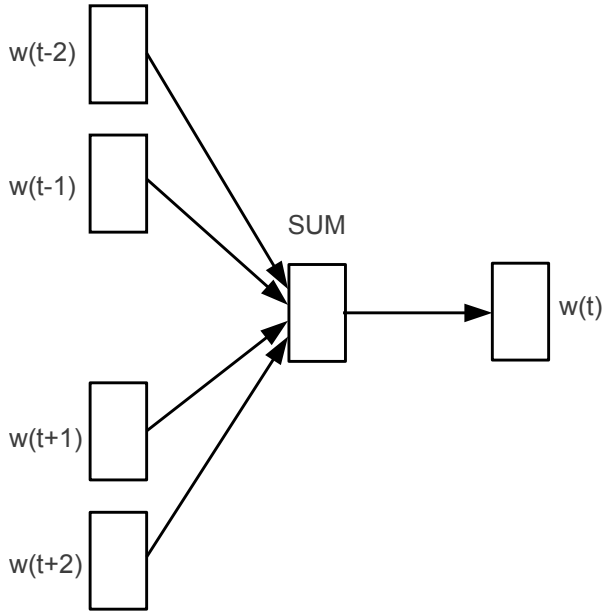
- In 2003: Bengio et al. [1] coin the term *word embeddings* and train them in a neural language model
- In 2008: Collobert and Weston [2]: show the utility of pre-trained word embeddings for downstream tasks
- In 2013: Mikolov et al. [3] created word2vec toolkit that allows us a seamless training and use o pre-trained embeddings

1. Yoshua Bengio et al. A neural probabilistic language model. 2003 *Journal of Machine Learning Research*
2. *Collobert and Weston, A unified architecture for natural language processing, ICML 2008*
3. *Mikolov et al., Efficient Estimation of Word Representations in Vector Space, ICLR 2013*

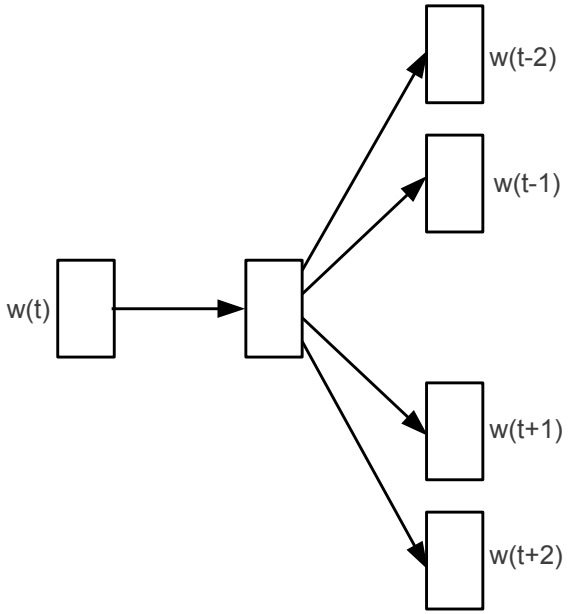
Idea

The construction of context vectors is turned:

- **From:** co-occurrence statistics + matrix factorization (heuristic stacking of vectors)
- **To:** vectors learned by training a supervised model (single supervised model)



CBOW



Skip-gram

prometeia

Notation

- We assume a corpus containing sequence of *T* training words $w_1, w_2, w_3, \ldots, w_t$
- The words belong to a vocabulary V of size |V|
- The models consider a context of *n* words
- We associate to every word an input embedding v_w with *d* dimensions and an output embedding v'_w

Data Preparation

- We use a sliding window of a fixed size moving along the given text
- The word in the middle is the *target* and the left-right neighborhood in the window is the *context*

Example: “The man who passes the sentence should swing the sword”

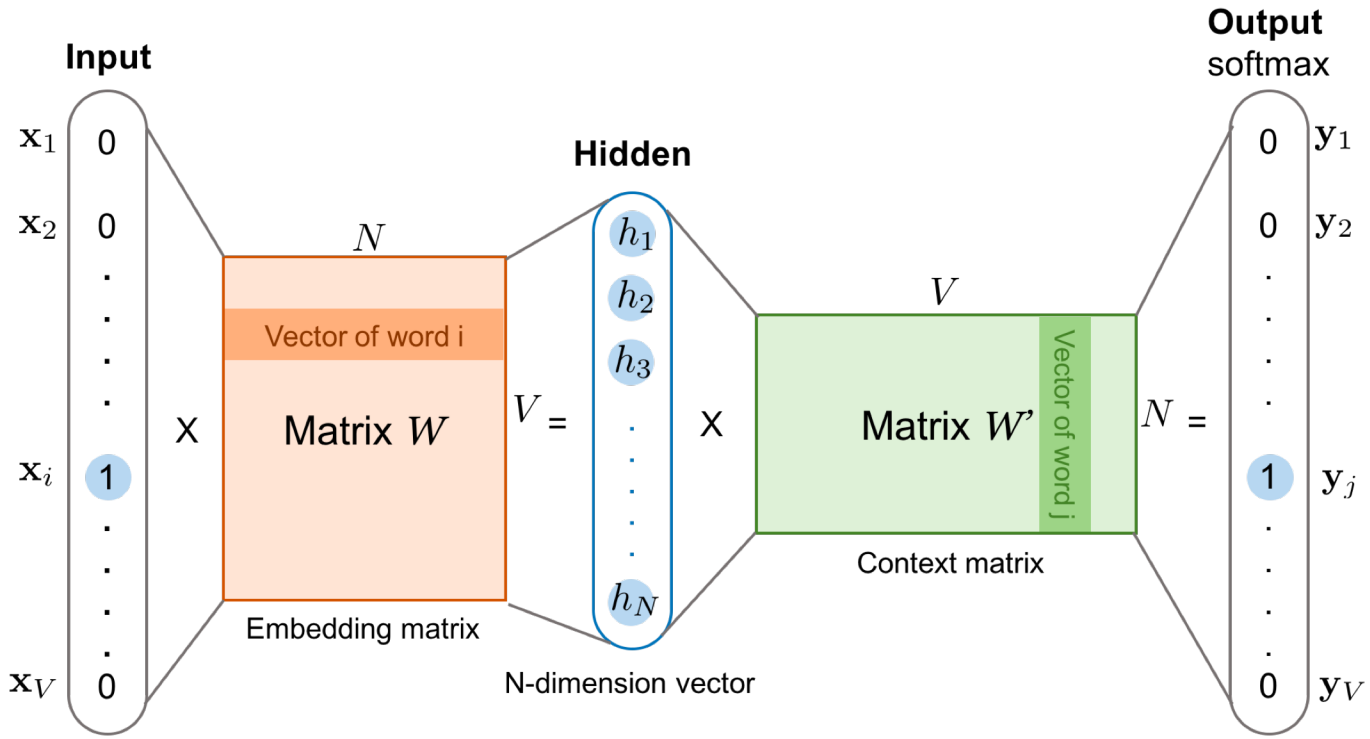
Sliding window (size = 5)	Target word	Context
[The man who]	the	man, who
[The man who passes]	man	the, who, passes
[The man who passes the]	who	the, man, passes, the
[man who passes the sentence]	passes	man, who, the, sentence
…	…	…
[sentence should swing the sword]	swing	sentence, should, the, sword
[should swing the sword]	the	should, swing, sword
[swing the sword]	sword	swing, the

prometeia

Skip-Gram Model

- Given a w_i the model is trained to predict the words in the window $[w_{t-c}, ..., \_, ..., w_{t+c}]$ (Mikolov et al., 2013)
- The target word is predicted by randomly select words close-by the target

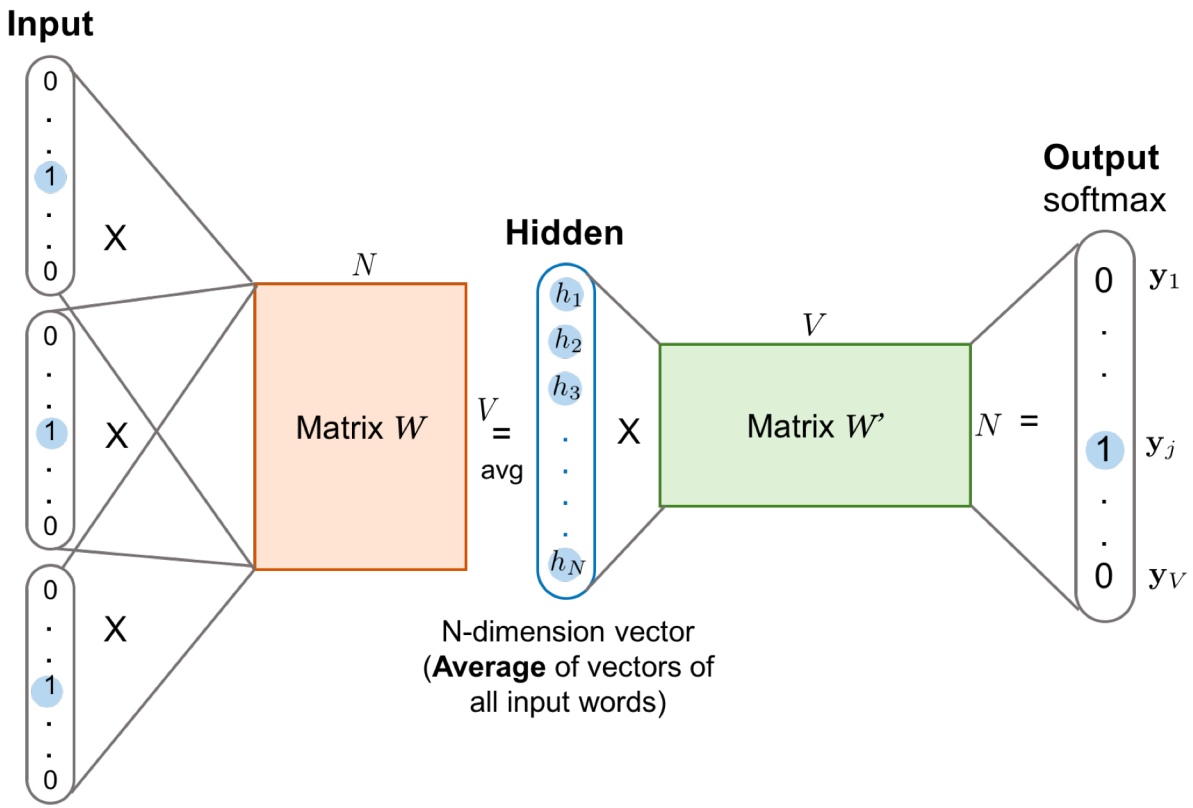
- Input w_i and output w_y are one-hot encoded
- The multiplication of **x** and W gives the embedding vector of input word w_i
- The multiplication of the hidden layer and W' produces the output one-hot vector **y**
- W' encoded the meaning of words as context



prometeia

Continuous Bag-of-Words (CBOW)

- **The Continuous Bag-of-Words** (Mikolov et al., 2013) predicts the target word from the surrounding window



CBOW model is better for small dataset

Loss function: Negative Sampling

- Considering an input word w_I, and the correct word w
- We sample N other words from a noise distribution $\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_N \sim Q$.

$$\mathcal{L}_\theta = -[\log \sigma({v'_w}^\top v_{w_I}) + \sum_{\substack{i=1 \\ \tilde{w}_i \sim Q}}^{N} \log \sigma(-{v'_{\tilde{w}_i}}^\top v_{w_I})]$$

Distance input target embedding

Distance input noise embedding

1. Get close to 0 when input and target embedding are equal
2. Get close to zero when the input embedding is far from noise word embeddings

Global Vectors (GloVe)

https://nlp.stanford.edu/pubs/glove.pdf

- Proposed by Pennington et al. in 2014, combines count-based matrix factorization to skip-gram
- Word meanings are captured by the ratios of co-occurrence probabilities

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

- large values correlate well with ice, and small values correlate well with steam
- *The ratio of probabilities encodes some crude form of meaning associated with the abstract concept*

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

prometeia

FastText: Enriching Word Vectors with Sub-word Information

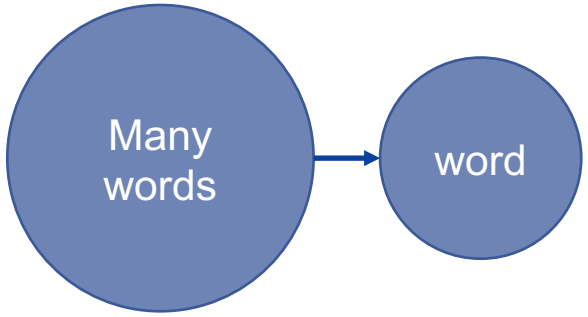
https://arxiv.org/pdf/1607.04606v1.pdf

- Word2Vec has several drawbacks:
  - **No sentence representations**: taking the average pre-trained word vector is popular but does not work always well
  - **Not exploiting morphology:** words with the same radicals are learned as different vectors

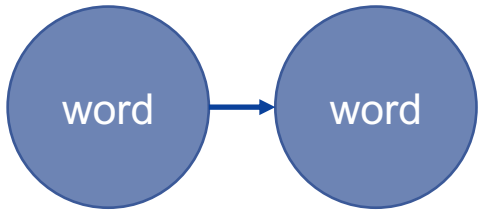
disastrous / disaster

mangera / mangerai

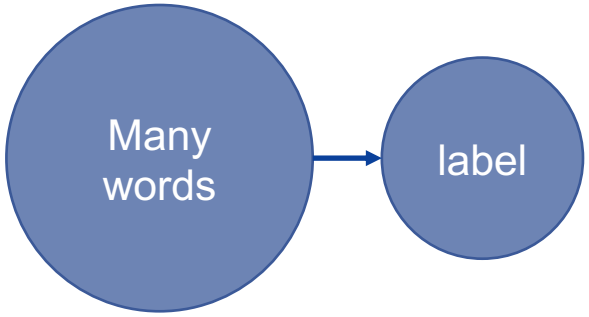
FastText is an extension of word2vec to take into account the morphology of words



cbow



Skip-gram



text classification

prometeia

FastText: Idea

Represent each word as a bag of *character n-grams*

1. A vector representation is associated to each character n-gram
2. Words are represented as the sum of these representations

Example: “matter”, with n=3 is represented as

^ma, mat, att, tte, ter, er**$**

Where **^** and **$** are special symbols to distinguish the n-grams from words in the vocabulary

- in this way compounds name and out of vocabulary words are easy to model

https://arxiv.org/pdf/1607.04606v1.pdf

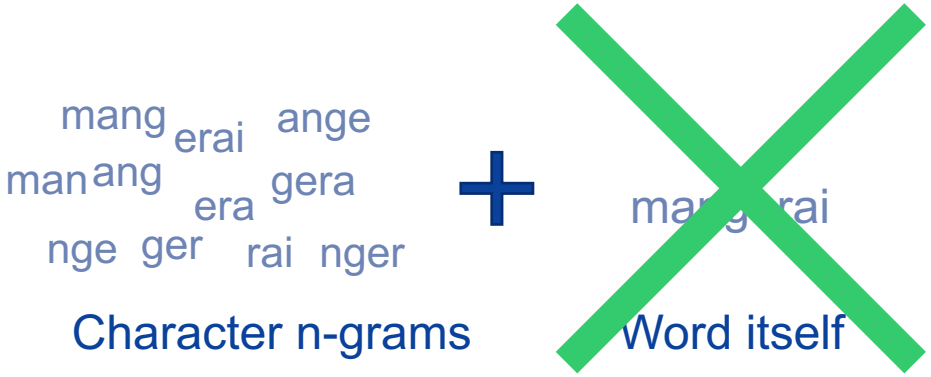
FastText: Features

- Features of a word computed using n-grams

$$h_w = \sum_{g \in w} x_g$$

- Out Of Vocabulary (OOV) words representations

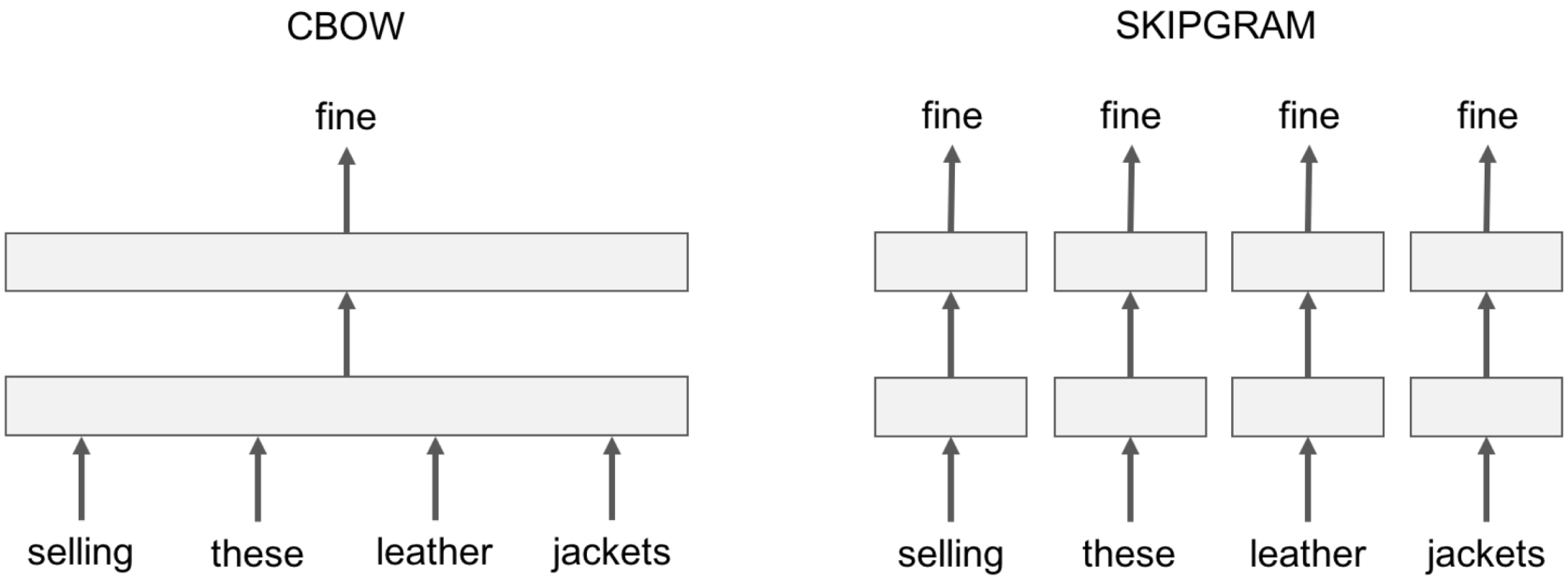
$$h_w = \sum_{g \in w} x_g$$



prometeia

FastText: Model

- the features preprocessed can used to train a CBOW or Skip-Gram (this works better) model



I am selling these fine leather jackets

prometeia

FastText: Extensions

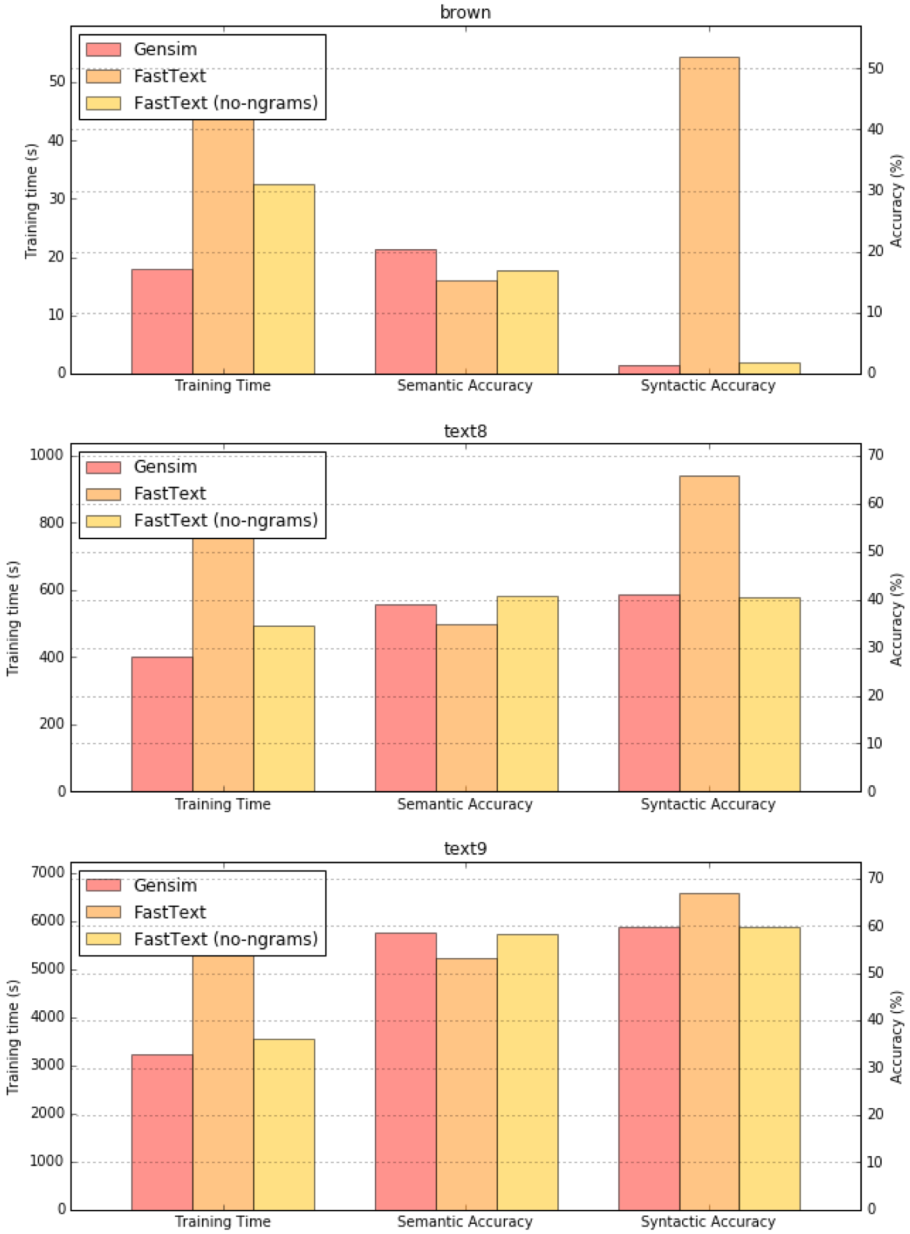
It can be used for:

1. **Classification** A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of Tricks for Efficient Text Classification, https://arxiv.org/abs/1607.01759

2. **Translation**: A. Joulin, P. Bojanowski, T. Mikolov, H. Jegou, E. Grave, Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion, https://aclweb.org/anthology/D18-1330

Word2Vec vs. FastText: Results Comparison

1. fastText with n-grams do better on syntactic tasks
2. Gensim word2vect and fastText with no n-grams do slightly better on semantic task (due to the test dataset)
3. In general, the performance of the models seems to get closer increasing the corpus size
4. The semantic accuracy increase with the corpus size
5. The syntactic accuracy slowly increase with the corpus size



prometeia

Recap: Count vs. Context

The main difference between the approaches is :

- **Documents as context:** for LSA and topic models (legacy from Information Retrieval)
  - *The document-based*: semantic relatedness (e.g. ‘boat’ – ‘water’)

- **Words as context:** for word skip-gram, CBOW, glove and FastText (linguistic and cognitive perspective)
  - *The word-based:* semantic similarity (e.g. ‘boat’ - ‘ship’)

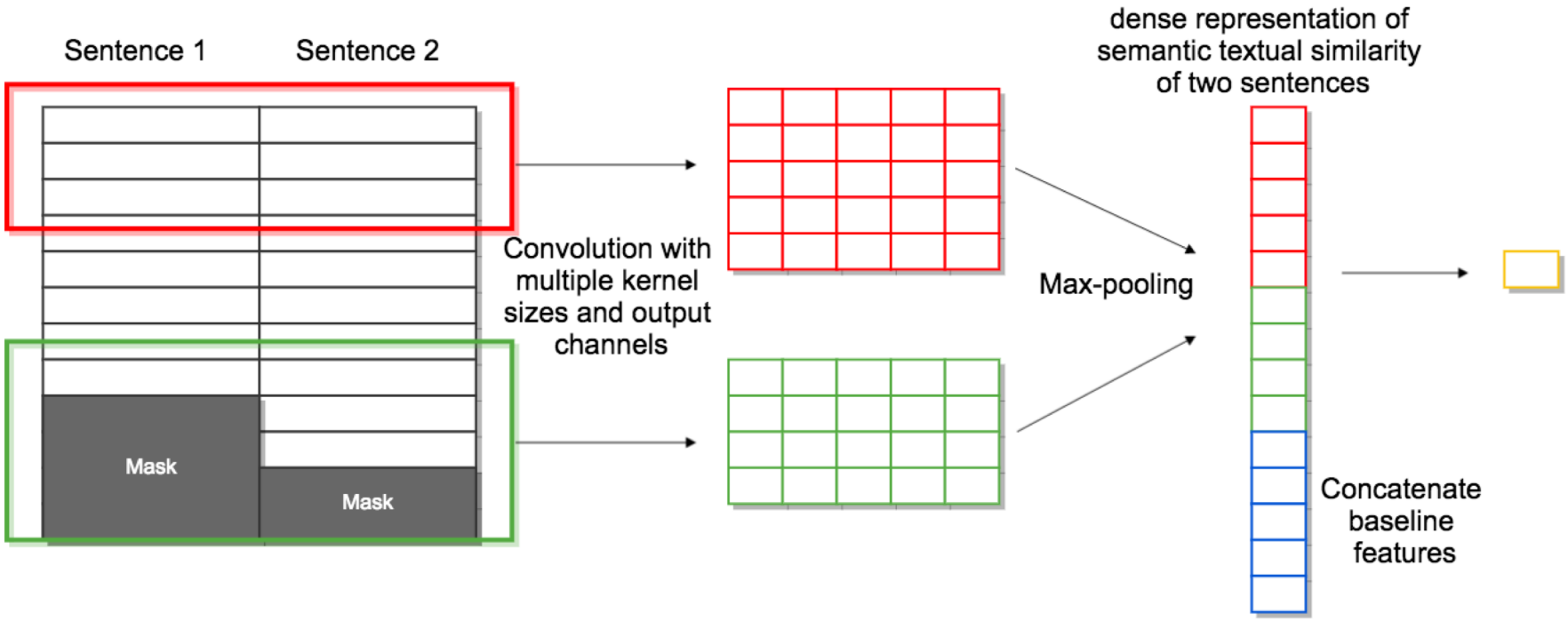
Interesting extensions not covered

- Poincaré Embeddings (https://arxiv.org/pdf/1705.08039.pdf)

prometeia

Interesting extensions not covered

- Sent2Vec Embeddings NAACL 2018 (https://aclweb.org/anthology/N18-1049)



- An extension of FastText and CBOW to sentences

prometeia

- Language and Machine Learning
- Distributional Semantics
- Context-free word embeddings:
  - Count based
  - Context based
- **Contextual embeddings**
- Final Considerations & Conclusions

prometeia

Contextualized Embeddings

- The introduced embeddings suffer of a major problem they cannot model ***polysemy***

- In the sentences:
  - *“I have an Apple phone”*, and
  - *“I am eating an apple”*

  the two “apple” words refer to very **different things** but share the **same embedding** vector

- **Solution**: Contextual Embeddings

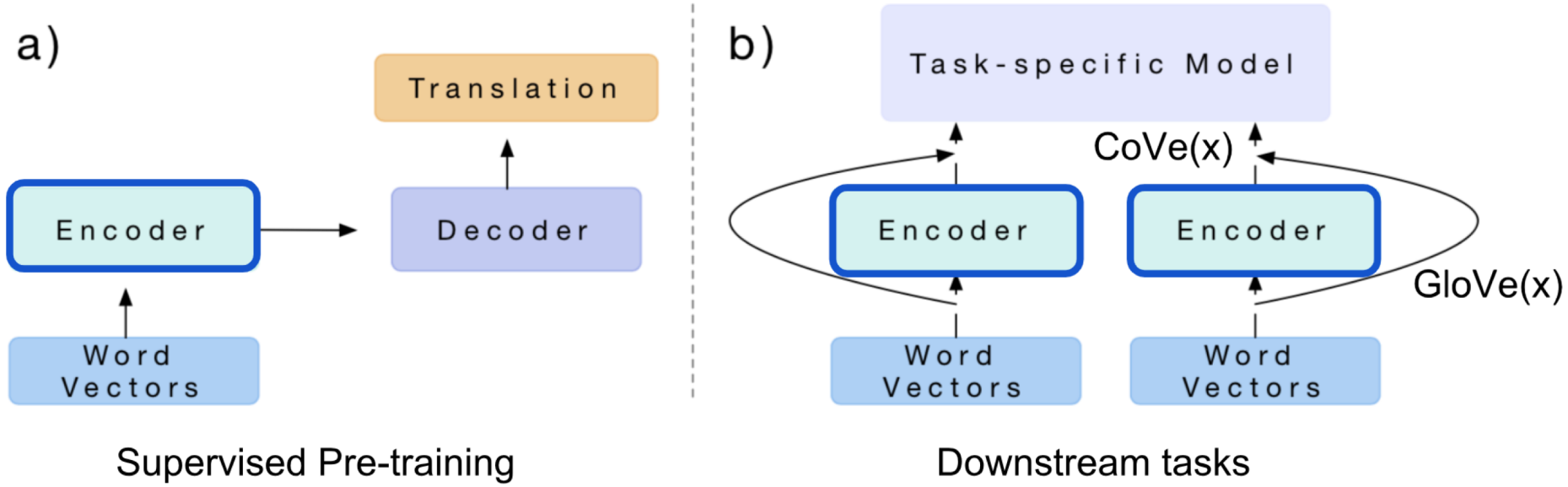
Contextualized Embeddings

- CoVe
- ELMO
- ULMFit
- OpenAI GPT
- BERT
- OpenAI GPT 2

prometeia

Contextual Word Vectors (CoVe)

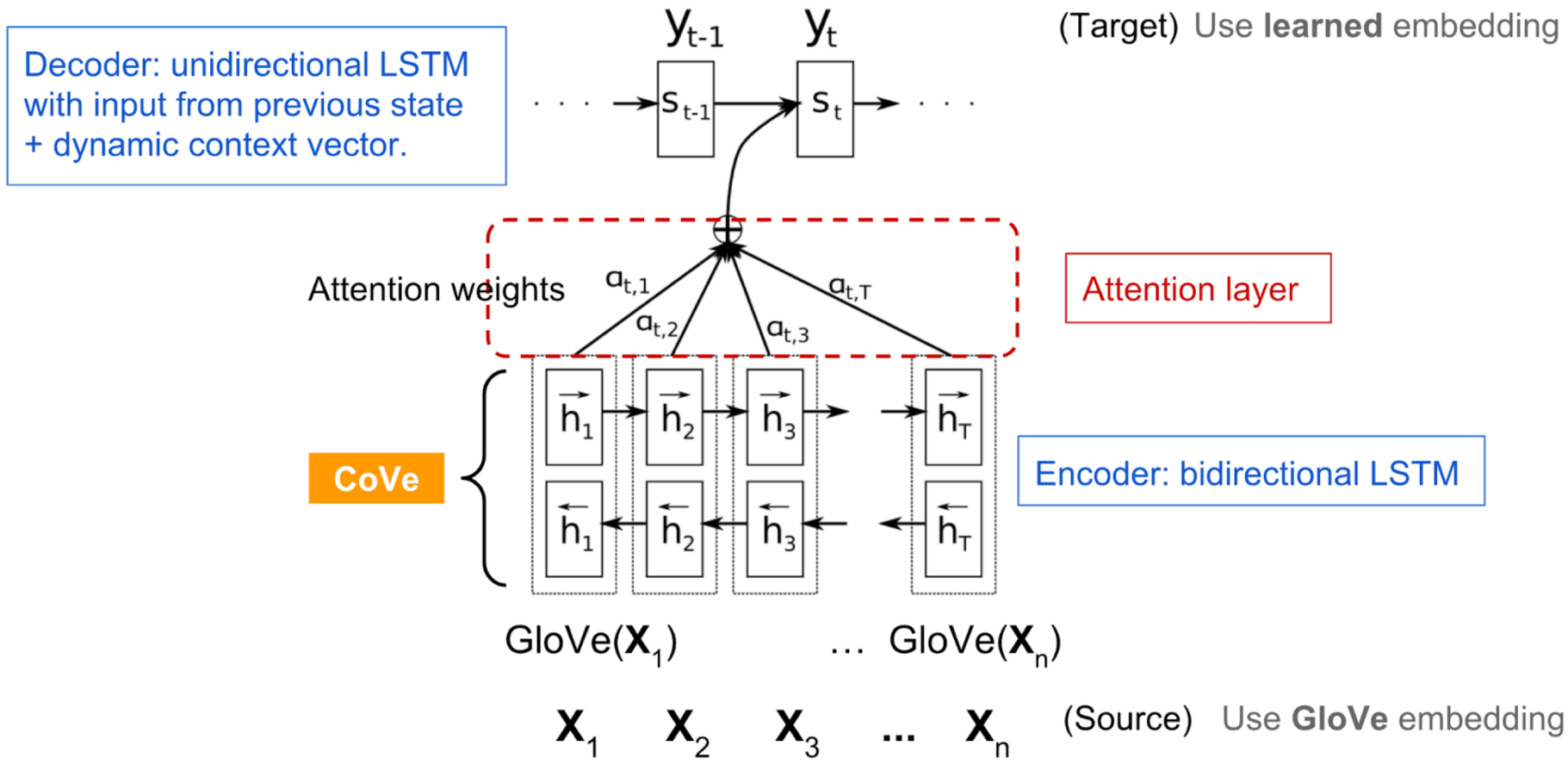
- Learns contextual word embeddings by using the encoder in an attentional sequence to sequence model trained for machine translation
- CoVe word representations are function of the entire input sequence



prometeia

McCann B, Bradbury J., Xiong c., Socher R. Learned in Translation: Contextualized Word Vectors NIPS 2017 https://arxiv.org/pdf/1708.00107.pdf

Contextual Word Vectors (CoVe)



McCann B, Bradbury J., Xiong c., Socher R. Learned in Translation: Contextualized Word Vectors NIPS 2017 https://arxiv.org/pdf/1708.00107.pdf

CoVe: Limitations

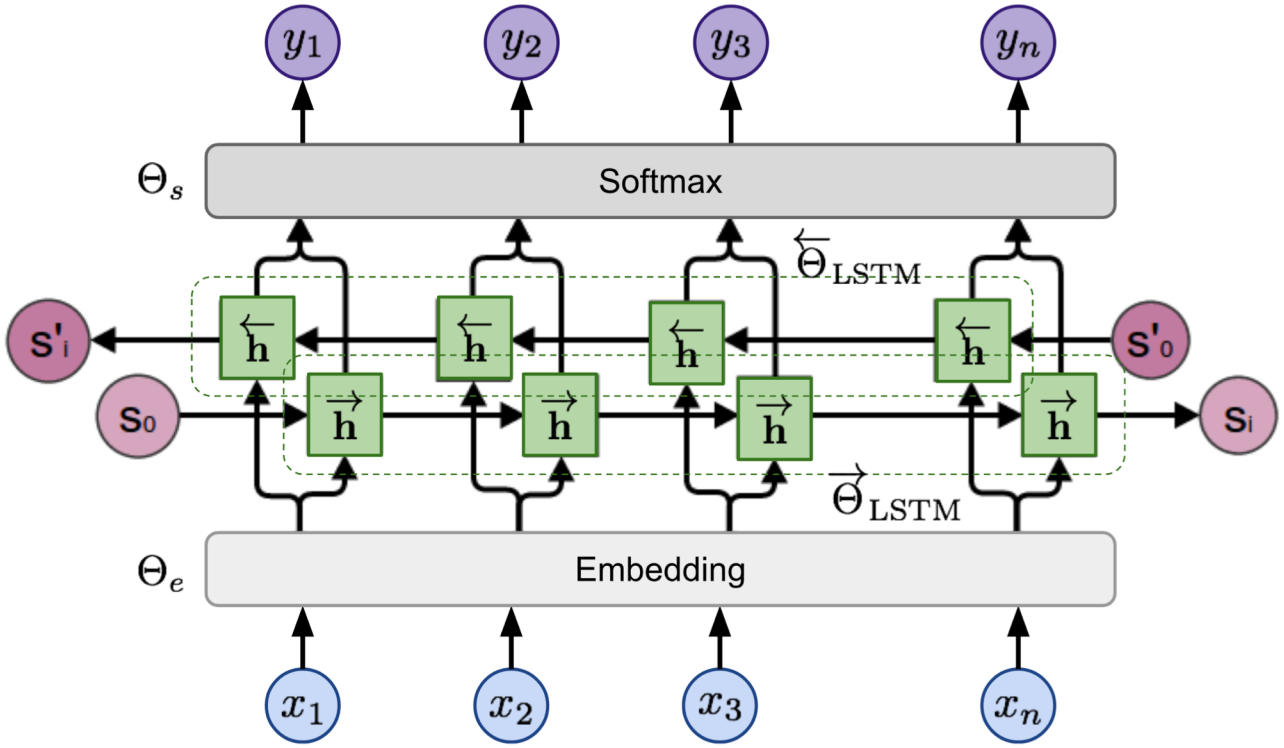
1. Pre-training is bounded by datasets for supervised translation

2. Limited contribution of CoVe when used as layer for other tasks

Embeddings from Language Model (ELMo)

Peters, Matthew E. and Neumann, Mark and Iyyer, Mohit and Gardner, Matt and Clark, Christopher and Lee, Kenton and Zettlemoyer, Luk
Deep contextualized word representations NAACL 2018

- Learns contextualized embeddings by pre-training a language model that learns to predict the next token given the history



prometeia

ELMo Representations

- ***Contextual***: the representation of each word depends on the sentence in which it is used

- ***Deep***: the word representation is computed by stacking the internal states of all the layers bidirectional language model

$$R_i = \{\mathbf{h}_{i,\ell} \mid \ell = 0, \ldots, L\} \text{ where } \mathbf{h}_{0,\ell} \text{ is the embedding layer output and } \mathbf{h}_{i,\ell} = [\overrightarrow{\mathbf{h}}_{i,\ell}; \overleftarrow{\mathbf{h}}_{i,\ell}]$$

Different layers encodes different kind of information (e.g. PoS tagging from lower layers and word-sense disambiguation in higher levels)

- ***Character based:*** the representation is based on characters rather than words (can take advantage of sub-words to compute ...)

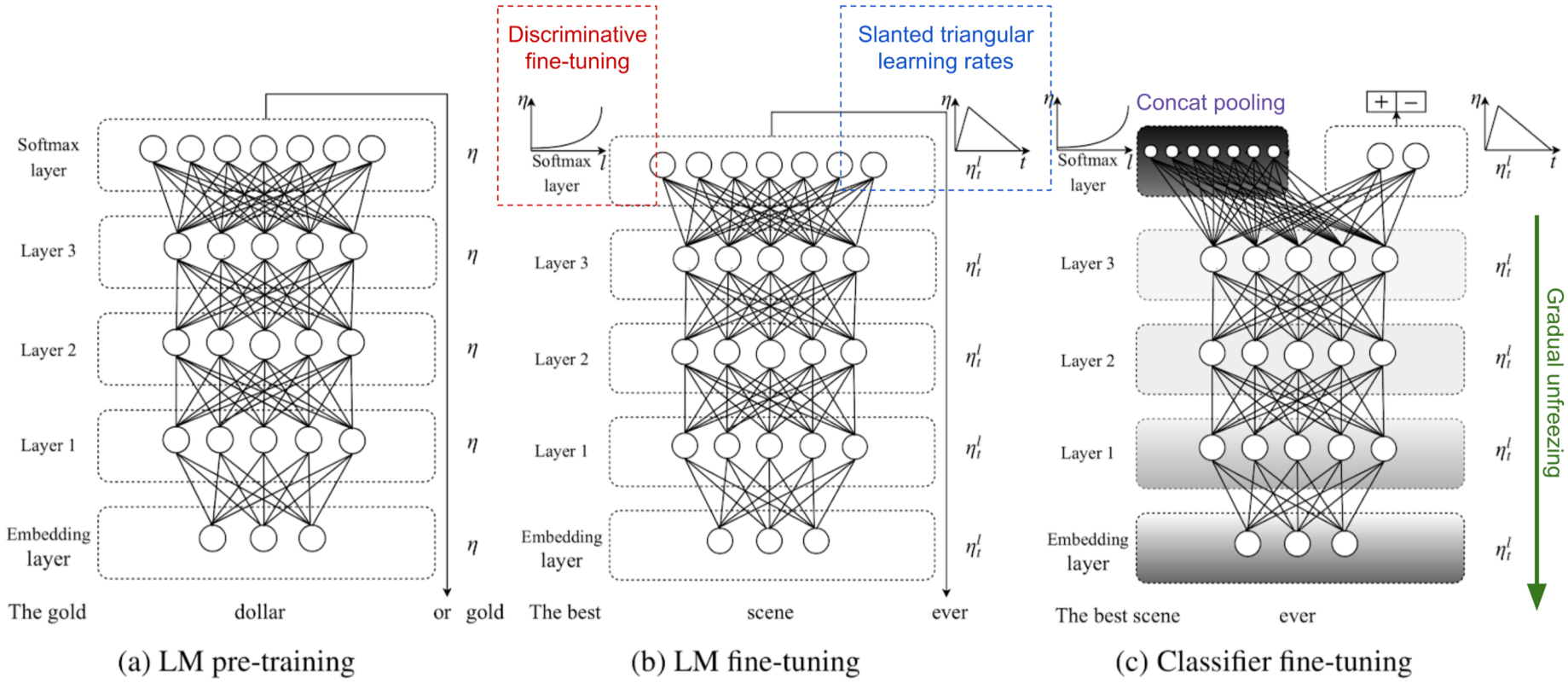
ELMo: Key Results

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%

Tasks and Datasets

- SQuAD: Stanford Question Answering Dataset
- SNLI: Stanford Natural Language inference (entailment, contradiction, neutral)
- SRL: Semantic Role Labeling
- Coref: Clustering mentions in text that refer to the same underlying real world entities
- NER: Named Entity Extraction
- Sentiment: The fine-grained sentiment classification task in the Stanford Sentiment Treebank

Universal Language Model Fine-tuning (ULMFiT)



Fine-tuning: (i) different learning rate per layer (ii) scheduled increase decrease of the learning rate

prometeia

Jeremy Howard, Sebastian Ruder Universal Language Model Fine-tuning for Text Classification, ACL 2018
https://arxiv.org/abs/1801.06146

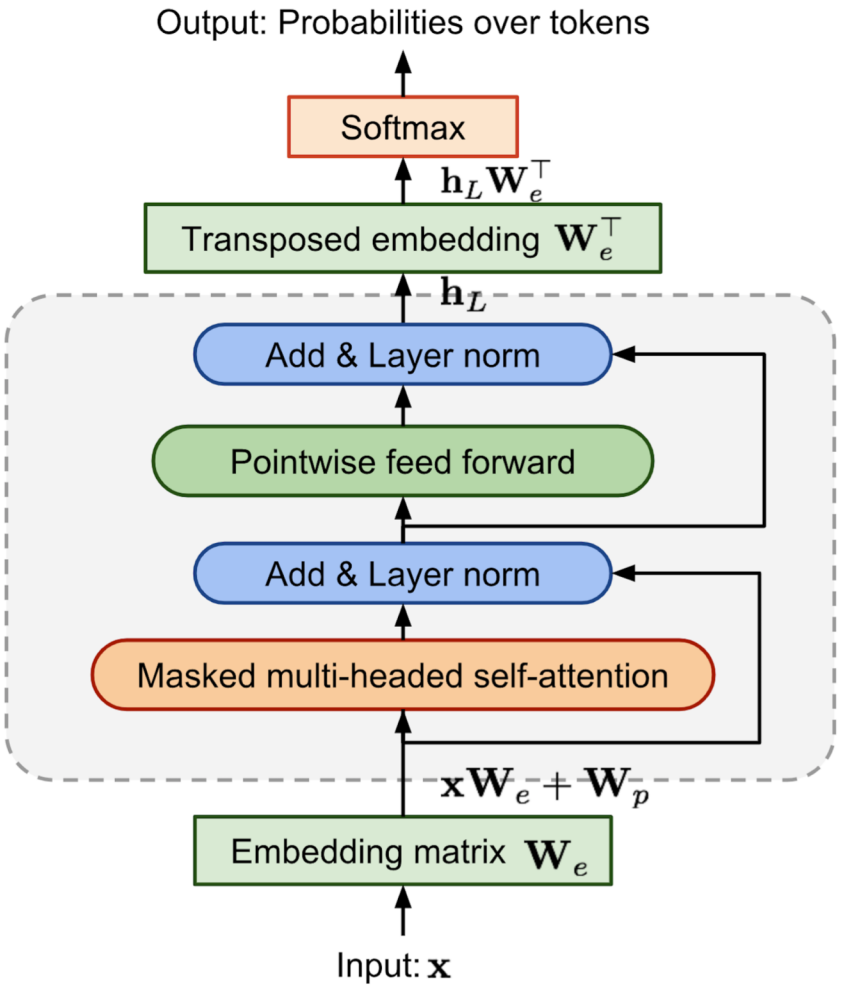
OpenAI Generative Pre-training Transformer

GPT has two major differences from ELMo:

Model	ELMo	GPT
Architecture	BiLSTM	Transformer
Type of Embeddings	Customized per task	Same embeddings fine tuned for al tasks

Radford A., et al. Improving Language Understanding by Generative Pre-Training
https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

GPT: Transformer Architecture



Transformer Block
Repeat x L=12

$$\mathbf{h}_\ell = \text{transformer\_block}(\mathbf{h}_{\ell-1})$$
$$\ell = 1, \ldots, L$$

prometeia

Radford A., et al. Improving Language Understanding by Generative Pre-Training
https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

GPT: Byte Pair Encoding

https://arxiv.org/abs/1508.07909

- Is a data compression technique that iteratively replaces the most frequent pair of symbols (originally bytes) in a given dataset with a single unused symbol.

- In each iteration, the algorithm finds the most frequent (adjacent) pair of symbols, each can be constructed of a single character or a sequence of characters, and merged them to create a new symbol

- It adopted to solve the open-vocabulary issue

https://github.com/google/sentencepiece

https://github.com/soaxelbrooke/python-bpe

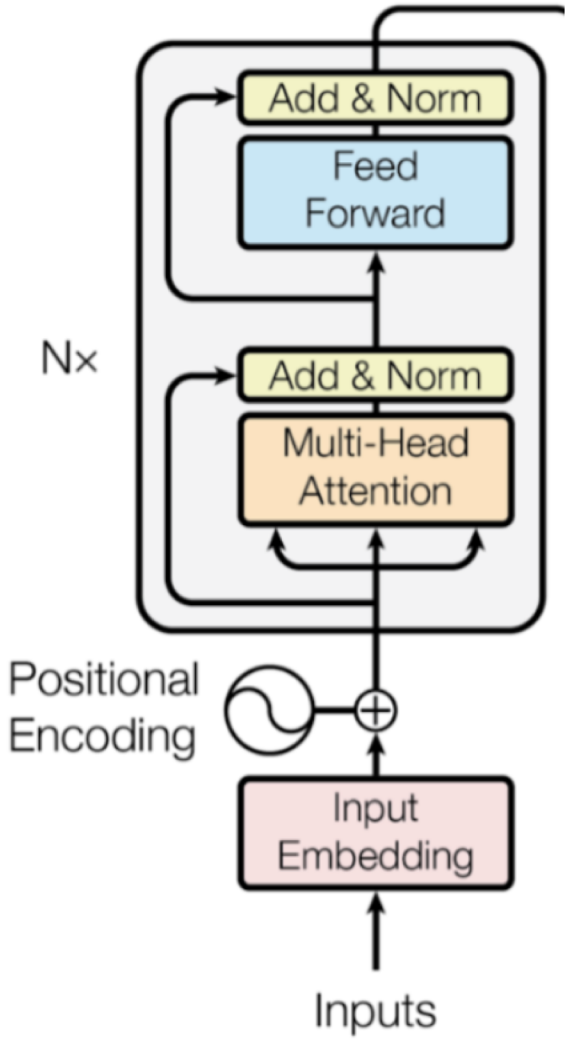
Bidirectional Encoder Representations from Transformer (BERT)

- Compared to GPT the largest difference is that BERT is Bidirectional

Contextual Embedding example: *“I made a bank deposit”*

1. **Unidirectional** : *“bank”* is represented only based on *“I made”* but not *“deposit”*
2. **Bidirectional:** *“bank”* is represented by using the left and right context

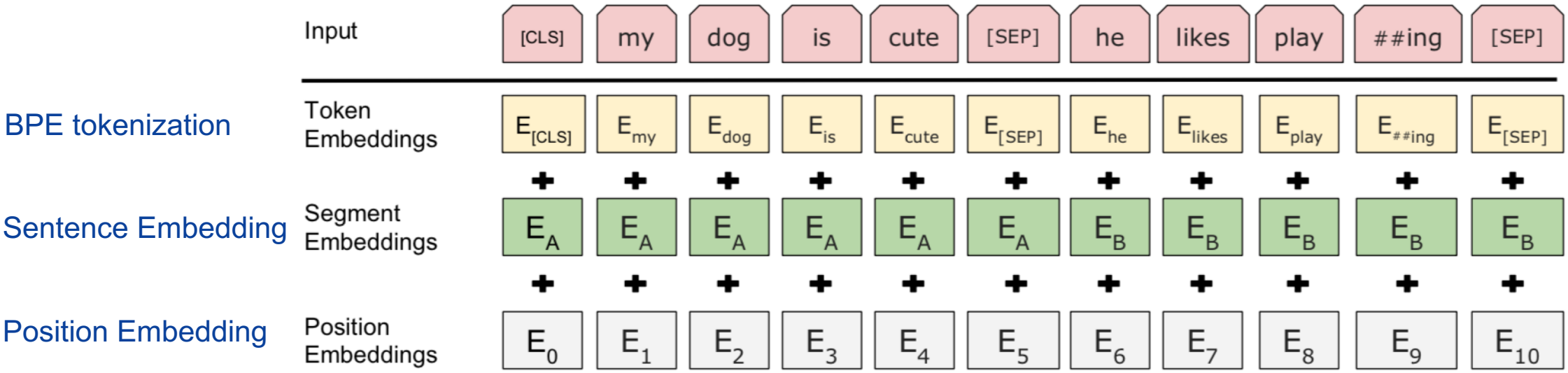
Moreover, BERT is available with **multilingual** where sentences from different language are decoded in the same latent space



prometeia

BERT input representation

- The input embedding is the sum of three parts:



prometeia

OpenAI GPT-2

Extension of GPT (1.5B parameters, 10x more) that:

1. **Zero-Shot transfer**: no need for transfer learning. All the downstream task are modeled as predicting conditional probabilities

2. **Model Modifications**
   - Layer normalization was moved to the input of each sub-block
   - Additional layer normalization after the final self-attention block
   - The weights of residual layers are scaled by 1/sqrt(N) where N is the number or residual layers
   - Use large vocabulary size and context

- Language and Machine Learning
- Distributional Semantics
- Context-free word embeddings:
  - Count based
  - Context based
- Contextual embeddings
- **Final Considerations & Conclusions**

prometeia

Final considerations & Conclusions

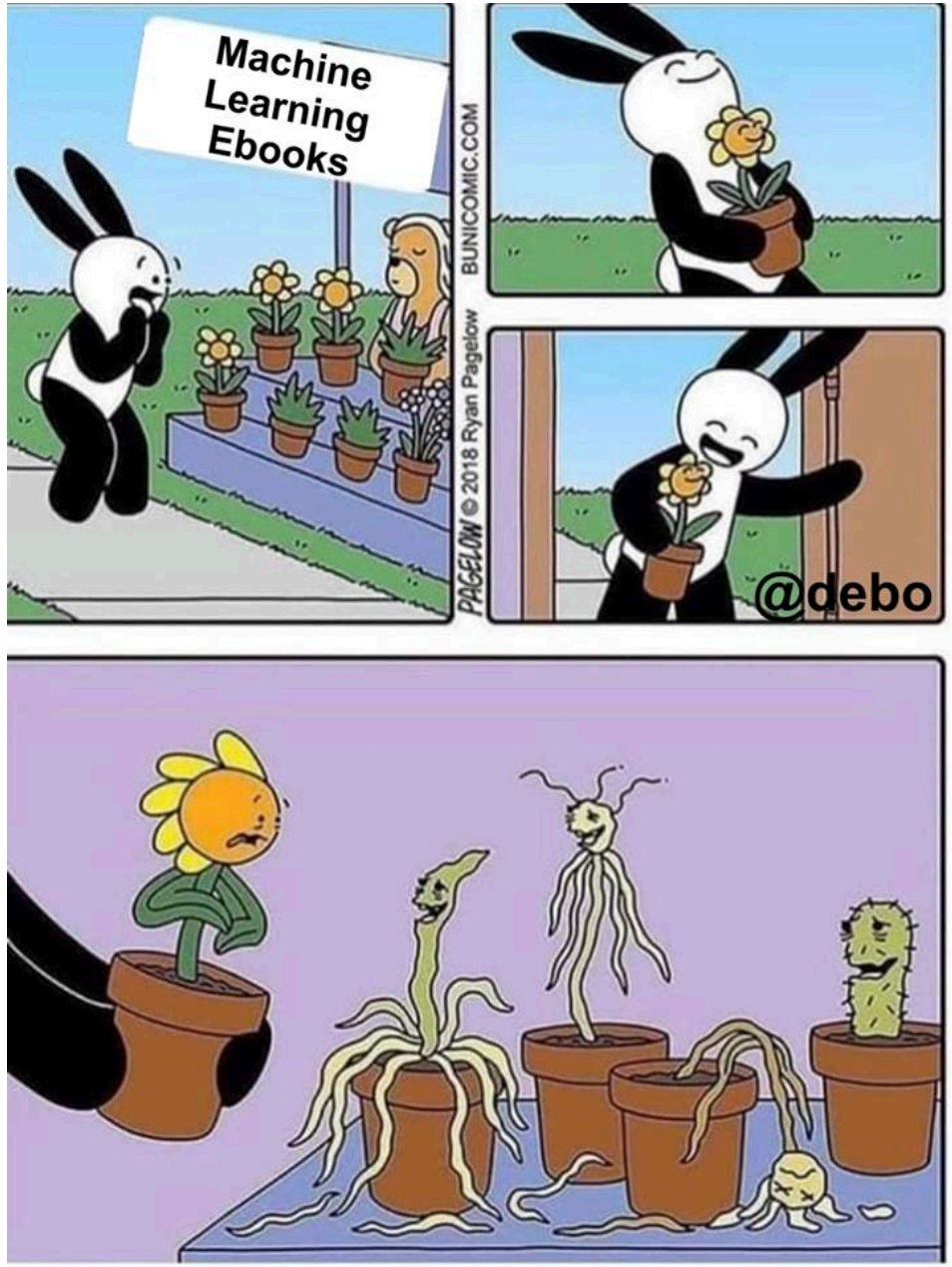
- We evaluated the evolution of context-free word embeddings to deal with:
  - Subword-level
  - OOV handling

- Then we considered contextual embeddings to address:
  - Polysemy and multi-sense word representation
  - Sentence embeddings
  - Embeddings for multiple languages

- What we do not covered are:
  - Temporal dimension
  - bias

prometeia

Confidentiality



Contacts

Bologna
Via Guglielmo Marconi, 43
+39 051 6480911
italy@prometeia.com

Milan
Via Brera, 18
Viale Monza, 265
+39 02 80505845
italy@prometeia.com

Rome
Viale Regina Margherita, 279
italy@prometeia.com

London
Dashwood House 69 Old Broad Street
EC2M 1QS
+44 (0) 207 786 3525
uk@prometeia.com

Istanbul
River Plaza, Kat 19
Büyükdere Caddesi Bahar Sokak
No. 13, 34394
| Levent | Istanbul | Turkey
+ 90 212 709 02 80 – 81 – 82
turkey@prometeia.com

Cairo
Smart Village - Concordia Building, B2111
Km 28 Cairo Alex Desert Road
6 of October City, Giza
egypt@prometeia.com

Moscow
ul. Ilyinka, 4
Capital Business Center Office 308
+7 (916) 215 0692
russia@prometeia.com

Prometeia

@PrometeiaGroup

Prometeiagroup

Prometeia

www.prometeia.com

prometeia