Word Embeddings, Attention, NLP tasks
2024

M. Vazirgiannis

https://bit.ly/2rwmvQU

LIX, Ecole Polytechnique

OUTLINE

- **Representation Learning for Text**
  - SVD
  - Word2Vec
  - Fast Text Embeddings – subword information
  - Document Representations
- CNNs for text classification
- Attention based architectures
- NLP tasks and evaluation
  - GLUE, FLUE
  - French Linguistics

Language model

- Goal: determine $\mathrm{P}(s = w_1 \dots w_k)$ in some domain of interest

$$P(s) = \prod_{i=1}^{k} P\left(w_i \mid w_1 \dots w_{i-1}\right)$$

e.g., $P(w_1 w_2 w_3) = P(w_1)\, P(w_2 \mid w_1)\, P(w_3 \mid w_1 w_2)$

- Traditional n-gram language model assumption:
  "probability of a word depends only on **context** of $n-1$ previous words"

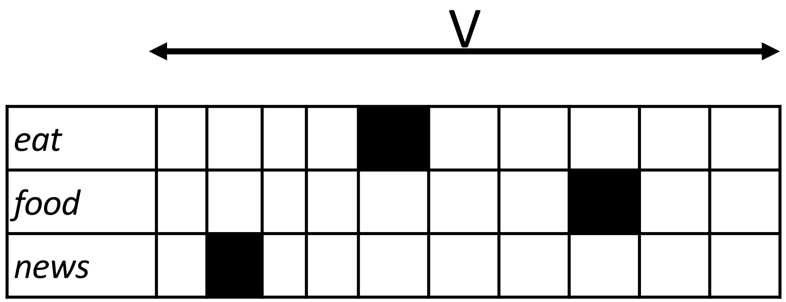
$$\Rightarrow \hat{P}(s) = \prod_{i=1}^{k} P\left(w_i \mid w_{i-n+1} \dots w_{i-1}\right)$$

- i.e. "Paris is the capital of France located in Ile de ...."

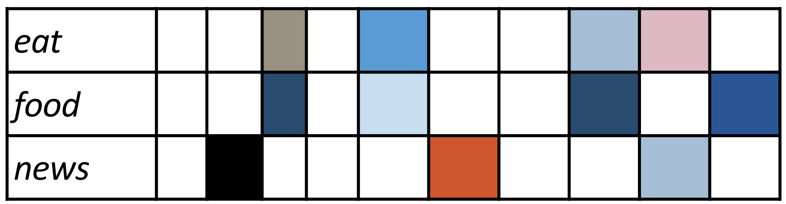
- Typical ML-smoothing learning process (e.g., Katz 1987):
  1. compute $\hat{P}\left(w_i \mid w_{i-n+1} \dots w_{i-1}\right) = \frac{\#w_{i-n+1} \dots w_{i-1} w_i}{\#w_{i-n+1} \dots w_{i-1}}$ on training corpus
  2. smooth to avoid zero probabilities

Representing Words

- **One-hot vector**
  - high dimensionality
  - sparse vectors
  - dimensions=|V| (10^6<|V|)
  - unable to capture semantic similarity between words

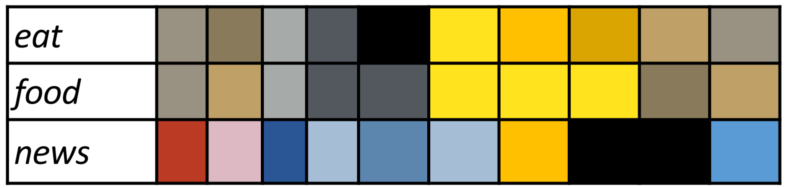


- **Distributional vector**
  - words that occur in similar contexts, tend to have similar meanings
  - each word vector contains the frequencies of all its neighbors
  - dimensions=|V|
  - computational complexity for ML algorithms



Representing Words

- **Word embeddings**
  - store the same contextual information in a low-dimensional vector
  - **densification** (sparse to dense)
  - **compression**
    - dimensionality reduction
    - dimensions=m $100<m<500$
  - able to capture semantic similarity between words
  - learned vectors (unsupervised)
  - Learning methods
    - **SVD**
    - **word2vec**
    - **GloVe**



Text Similarity

- We should assign similar probabilities (discover similarity) to *Obama speaks to the media in Illinois* and the *President addresses the press in Chicago*

- This does not happen because of the "one-hot" vector space representation

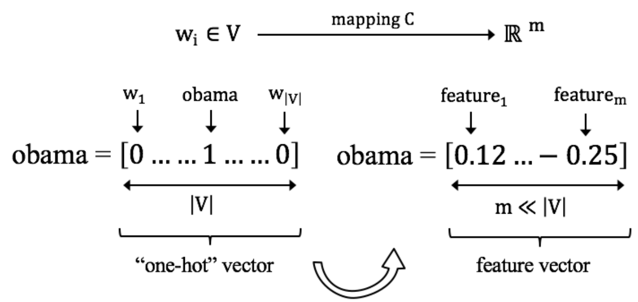
One hot

$$\left.\begin{aligned} \text{obama} &= [0\,0\,0\,0\ \dots\ 0\,1\,0\,0] \\ \text{president} &= [0\,0\,0\,1\ \dots\ 0\,0\,0\,0] \end{aligned}\right\} \overrightarrow{\text{obama}}.\overrightarrow{\text{president}} = \vec{0}$$

$$\left.\begin{aligned} \text{speaks} &= [0\,0\,1\,0\ \dots\ 0\,0\,0\,0] \\ \text{addresses} &= [0\,0\,0\,0\ \dots\ 0\,0\,1\,0] \end{aligned}\right\} \overrightarrow{\text{speaks}}.\overrightarrow{\text{addresses}} = \vec{0}$$

$$\left.\begin{aligned} \text{illinois} &= [1\,0\,0\,0\ \dots\ 0\,0\,0\,0] \\ \text{chicago} &= [0\,1\,0\,0\ \dots\ 0\,0\,0\,0] \end{aligned}\right\} \overrightarrow{\text{illinois}}.\overrightarrow{\text{chicago}} = \vec{0}$$

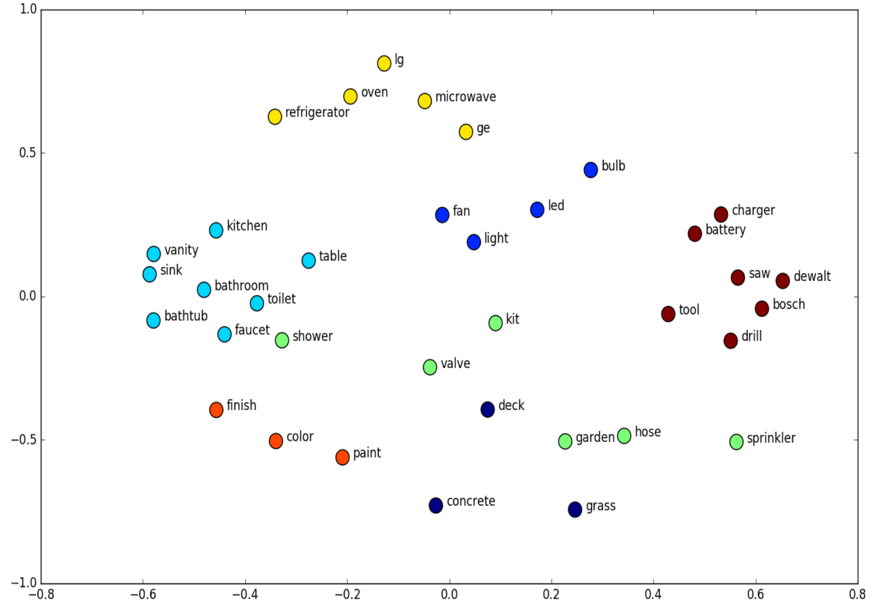
Word embeddings



Representation Learning for Text

- **“a word is defined by “the company it keeps” (Firth, 1957)**

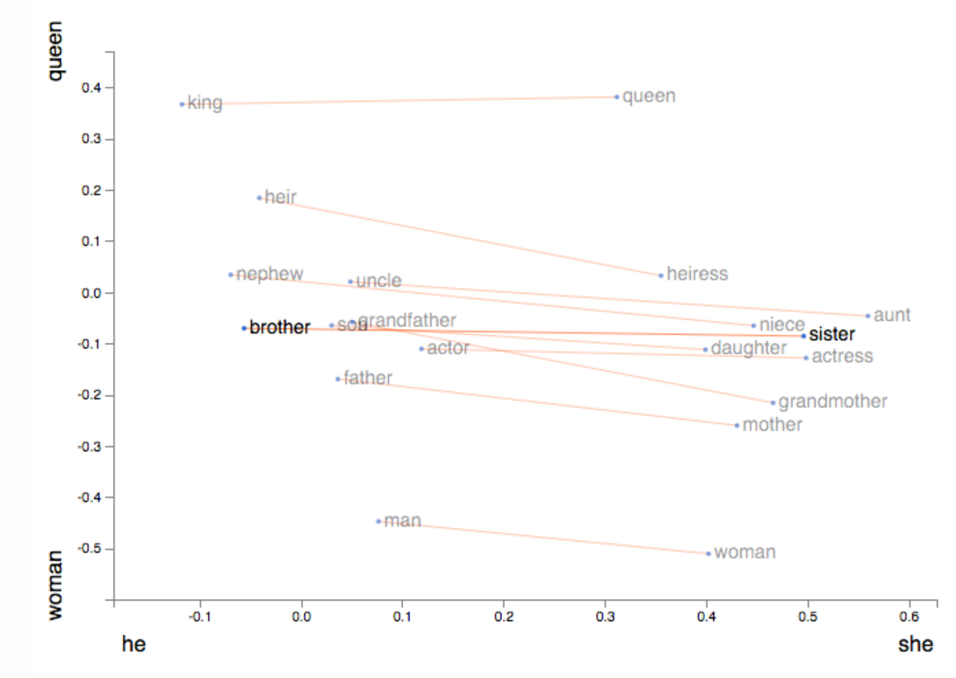
- Word embeddings are a class of algorithms where each word is represented as real-valued vector.
- The learning process of these vectors is either joint with a neural network model on some task or is an unsupervised process.
- Similar words in meaning have similar representation.



Representation Learning for Text

- Words with similar meaning end up close to each other
- Words sharing similar contexts may be analogous
  - Synonyms
  - Antonyms
  - Names
  - Colors
  - Places
  - Interchangeable words

- Vector arithmetics to work with analogies
- i.e. **king - man + woman** = **queen**



https://lamyiowce.github.io/word2viz/

OUTLINE

- Representation Learning for Text
  - **SVD**
  - Word2Vec
  - Fast Text Embeddings – subword information
- Attention based architectures
  - ELMO, BERT. BART
- NLP tasks and evaluation
  - GLUE, FLUE
  - French Linguistics

SVD word embeddings

- Dimensionality reduction on co-occurrence matrix
- Create a $|V|x|V|$ word co-occurrence matrix X
- Apply SVD $X = USV^T$
- Take first k columns of U
- Use the k-dimensional vectors as representations for each word
- Able to capture semantic and syntactic similarity

LSI – an example

LSI application on a term – document matrix

C1: Human machine Interface for Lab ABC computer application

C2: A survey of user opinion of computer system response time

C3: The EPS user interface management system

C4: System and human system engineering testing of EPS

C5: Relation of user-perceived response time to error measurements

M1: The generation of random, binary unordered trees

M2: The intersection graph of path in trees

M3: Graph minors IV: Widths of trees and well-quasi-ordering

M4: Graph minors: A survey

- The dataset consists of 2 classes, 1st: “human – computer interaction” (c1-c5) 2nd: related to graph (m1-m4). After feature extraction the titles are represented as follows.

Indexing by Latent Semantic Analysis (1990) Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman, Journal of the American Society of Information Science

LSI – an example

$A=ULV^T$

A=

1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0
0	1	1	2	0	0	0	0	0
0	1	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	1

LSI – an example

$A=ULV^T$

U=

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41	0	0	0
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11	0	0	0
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49	0	0	0
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01	0	0	0
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27	0	0	0
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05	0	0	0
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05	0	0	0
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17	0	0	0
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58	0	0	0
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23	0	0	0
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23	0	0	0
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18	0	0	0

LSI – an example

$A=ULV^T$

L=

3.34	0	0	0	0	0	0	0	0
0	2.54	0	0	0	0	0	0	0
0	0	2.35	0	0	0	0	0	0
0	0	0	1.64	0	0	0	0	0
0	0	0	0	1.50	0	0	0	0
0	0	0	0	0	1.31	0	0	0
0	0	0	0	0	0	0.85	0	0
0	0	0	0	0	0	0	0.56	0
0	0	0	0	0	0	0	0	0.36
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

LSI – an example

$A=ULV^T$

V=

0.20	-0.06	0.11	-0.95	0.05	-0.08	0.18	-0.01	-0.06
0.61	0.17	-0.50	-0.03	-0.21	-0.26	-0.43	0.05	0.24
0.46	-0.13	0.21	0.04	0.38	0.72	-0.24	0.01	0.02
0.54	-0.23	0.57	0.27	-0.21	-0.37	0.26	-0.02	-0.08
0.28	0.11	-0.51	0.15	0.33	0.03	0.67	-0.06	-0.26
0.00	0.19	0.10	0.02	0.39	-0.30	-0.34	0.45	-0.62
0.01	0.44	0.19	0.02	0.35	-0.21	-0.15	-0.76	0.02
0.02	0.62	0.25	0.01	0.15	0.00	0.25	0.45	0.52
0.08	0.53	0.08	-0.03	-0.60	0.36	0.04	-0.07	-0.45

LSI – an example

Choosing the 2 largest singular values we have

U_k=

0.22	-0.11
0.20	-0.07
0.24	0.04
0.40	0.06
0.64	-0.17
0.27	0.11
0.27	0.11
0.30	-0.14
0.21	0.27
0.01	0.49
0.04	0.62
0.03	0.45

L_k=

3.34	0
0	2.54

V_k^T=

0.20	0.61	0.46	0.54	0.28	0.00	0.02	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53

LSI reconstruction (2 singular values)

$A_k =$

	C1	C2	C3	C4	C5	M1	M2	M3	M4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
Interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
Computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
User	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
System	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
Response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
Time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
Survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
Trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
Graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
Minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

LSI Example

- Query: “human computer interaction” retrieves documents: c_1,c_2, c_4 but *not* c_3 and c_5.
- If we submit the same query (based on the transformation shown before) to the transformed matrix we retrieve (using cosine similarity) all c_1-c_5 even if c_3 and c_5 have no common keyword to the query.
- According to the transformation for the queries we have:

Query transformation

	query
human	1
Interface	0
computer	1
User	0
System	0
Response	0
Time	0
EPS	0
Survey	0
Trees	0
Graph	0
Minors	0

$$q = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Query transformation

$$q^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$U_k = \begin{bmatrix} 0.22 & -0.11 \\ 0.20 & -0.07 \\ 0.24 & 0.04 \\ 0.40 & 0.06 \\ 0.64 & -0.17 \\ 0.27 & 0.11 \\ 0.27 & 0.11 \\ 0.30 & -0.14 \\ 0.21 & 0.27 \\ 0.01 & 0.49 \\ 0.04 & 0.62 \\ 0.03 & 0.45 \end{bmatrix}$$

$$L_k = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.39 \end{bmatrix}$$

$$q_n = q^T U_k L_k = \begin{bmatrix} 0.138 & -0.0273 \end{bmatrix}$$

Query transformation

Map docs to the 2 dim space $V_k L_k =$

0.20	-0.06
0.61	0.17
0.46	-0.13
0.54	-0.23
0.28	0.11
0.00	0.19
0.01	0.44
0.02	0.62
0.08	0.53

3.34	0
0	2.54

=

0.67	-0.15
2.04	0.43
1.54	-0.33
1.80	-0.58
0.94	0.28
0.00	0.48
0.03	1.12
0.07	1.57
0.27	1.35

$q_n L_k =$

0.138	-0.0273

3.34	0
0	2.54

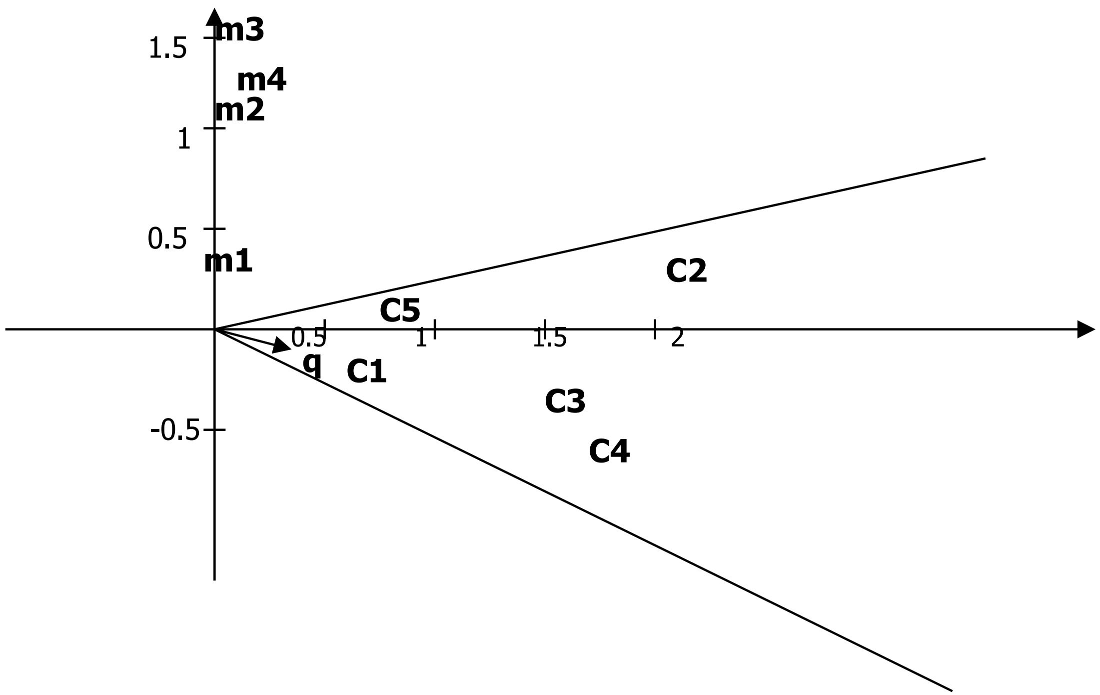
=

0.46	-0.069

Query transformation

- The cosine similarity matrix of query vector to the documents is:

	query
C1	0.99
C2	0.94
C3	0.99
C4	0.99
C5	0.90
M1	-0.14
M2	-0.13
M3	-0.11
M4	0.05



SVD problems

- The dimensions of the matrix change when dictionary changes
- The whole decomposition must be re-calculated when we add a word
- Sensitive to the imbalance of word frequency
- Very high dimensional matrix
- Not suitable for large corpora or dictionaries
- Quadratic cost to perform SVD
- Solution: Directly calculate a low-dimensional representation

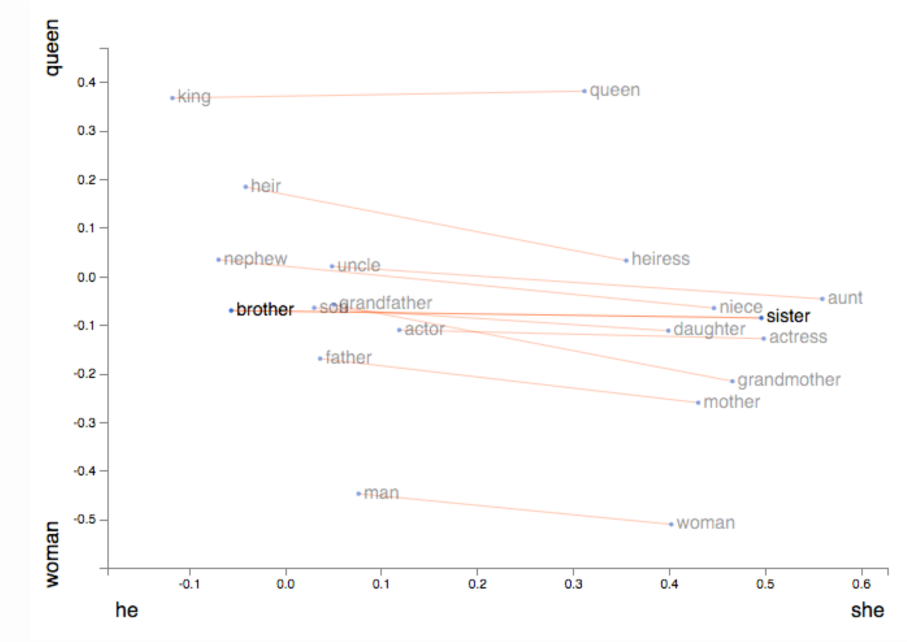
OUTLINE

- Representation Learning for Text
  - SVD
  - **Word2Vec**
  - Fast Text Embeddings – subword information
  - Document representations
- NLP tasks and evaluation
  - GLUE, FLUE
  - French Linguistics

Word analogy

- Words with similar meaning end up close to each other
- Words sharing similar contexts may be analogous
  - Synonyms
  - Antonyms
  - Names
  - Colors
  - Places
  - Interchangeable words

- Vector arithmetics to work with analogies
- i.e. **king - man + woman** = **queen**



https://lamyiowce.github.io/word2viz/

But why?

- what's an analogy?

$$\frac{p(w'|man)}{p(w'|woman)} \approx \frac{p(w'|king)}{p(w'|queen)}$$

Assume PMI is approximated by a low rank approximation of the co-occurrence matrix.

1. $PMI(w', w) \approx v_w v_{w'}$ *inner product*
2. Isotropic: $E_{w'}[(v_{w'} v_u)]^2 = ||v_u||^2$

Then

3. $argmin_w\, E_{w'}[ln \frac{p(w'|w)}{p(w'|queen)} - ln \frac{p(w'|man)}{p(w'|woman)}]^2$
4. $argmin_w\, E_{w'}[\left(PMI(w'|w) - PMI(w'|queen)\right) - (PMI(w'|man) - PMI(w'|woman))]^2$
5. $argmin_w ||\left(v_w - v_{queen}\right) - (v_{man} - v_{woman})||^2$
6. $v_w \approx v_{queen} - v_{woman} + v_{man}$ which is an analogy!

- *Arora et al (ACL 2016)* shows that if (2) holds then (1) holds as well
- So we need to construct vectors from co-occurrence that satisfy (2)
- d<<|V| in order to have isotropic vectors

A Latent Variable Model Approach to PMI-based Word Embeddings, *Arora et al (ACL 2016)*

Learning Word Vectors

- Corpus containing a sequence of T training words
- Objective: $\mathrm{f}(w_t, \ldots, w_{t-n+1}) = \widehat{\mathrm{P}}(\mathrm{w_t} \mid \mathrm{w_{t-n+1}} \ldots \mathrm{w_{t-1}})$
- Decomposed in two parts:

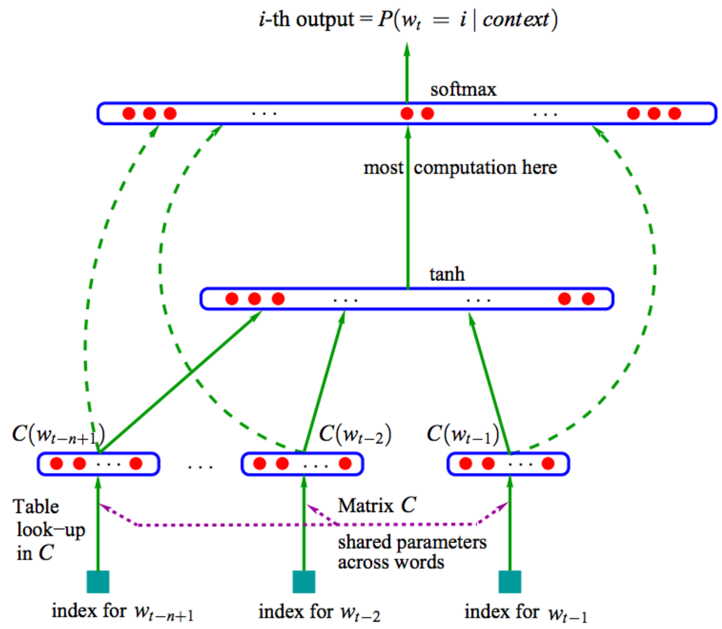
$$\mathrm{w_i} \in \mathrm{V} \xrightarrow{\text{mapping C}} \mathbb{R}^{\mathrm{m}}$$

  - Mapping **C** (1-hotv => lower dimensions)
  - Mapping any **g** s.t. (estimate prob t+1| t previous)
    $\mathrm{f}(w_{t-1},\cdots,w_{t-n+1}) = \mathrm{g}(\mathrm{C}(w_{t-1}),\cdots,\mathrm{C}(w_{t-n+1}))$

• C(i) is the i-th word feature vector

(Word embedding)

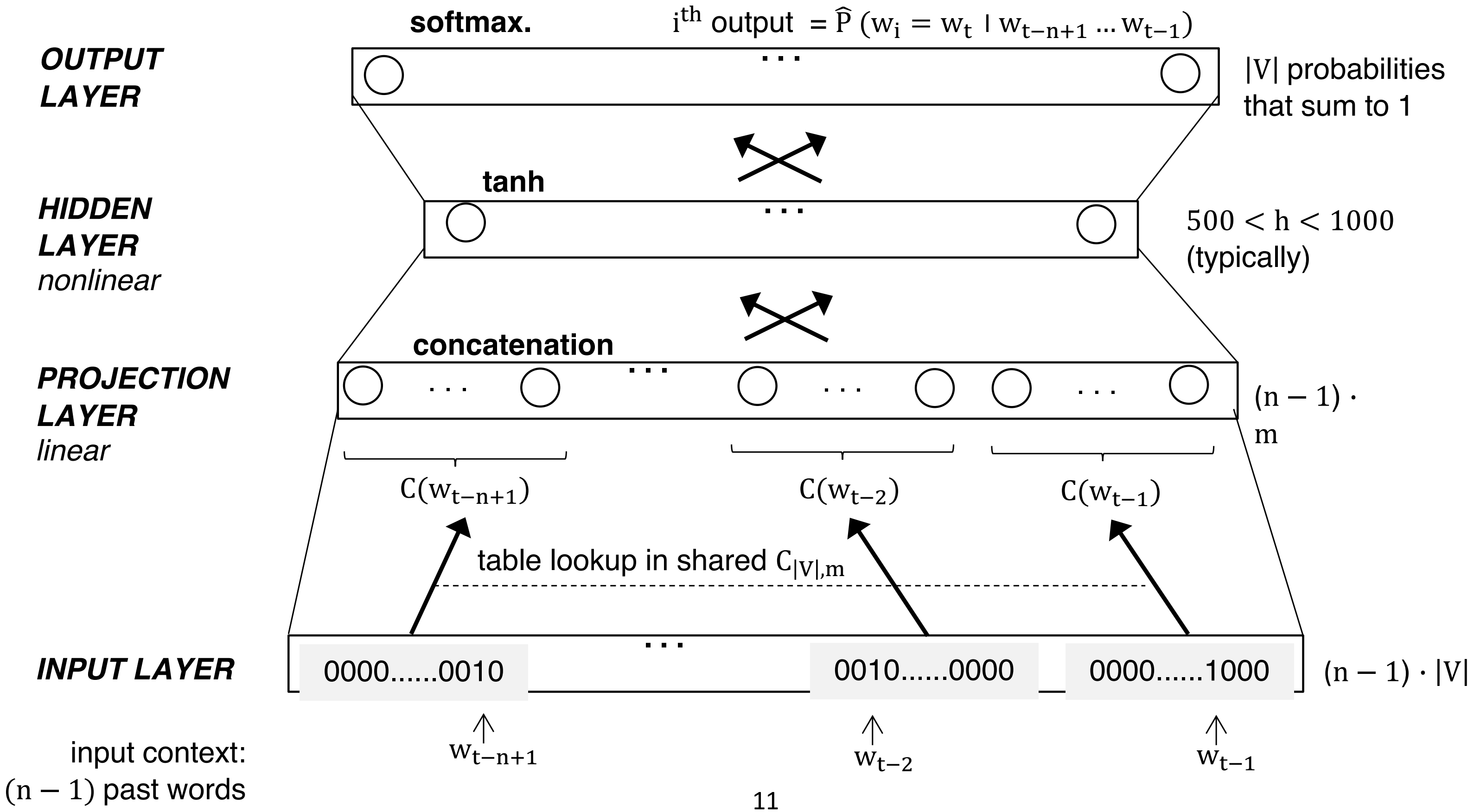
- Objective function: $J = \frac{1}{T}\sum \mathrm{f}(w_t, \ldots, w_{t-n+1})$



Bengio, Yoshua, et al. "A neural probabilistic language model." *The Journal of Machine Learning Research* 3 (2003): 1137-1155.

Neural Net Language Model

For each training sequence: input = (context, target) pair: $(w_{t-n+1} \dots w_{t-1}, w_t)$
objective: minimize $E = -\log \hat{P}(w_t \mid w_{t-n+1} \dots w_{t-1})$



Objective function

- $\mathrm{E} = -\log \widehat{\mathrm{P}}(\mathrm{w_t} \mid \mathrm{w_{t-n+1}} \ldots \mathrm{w_{t-1}})$
- a probability between 0 and 1.
- On this support, the log is negative => –log term positive.
- makes sense to try to minimize it. ~

  - Probability of word given the context be as high as possible (1 for a perfect prediction).

    - case the error is equal to 0 (global minimum).

p	**log(p)**	**-log(p)**
0,7	-0,15490196	0,15490196
0,2	-0,698970004	0,698970004

NNLM facts

- tested on Brown (1.2M words, $|\mathrm{V}| \cong$ 16K) and AP News (14M words, $|\mathrm{V}| \cong$ 150K reduced to 18K) corpuses
- Brown: $\mathrm{h} = 100$, $\mathrm{n} = 5$, $\mathrm{m} = 30$
- AP News: $\mathrm{h} = 60$, $\mathrm{n} = 6$, $\mathrm{m} = 100$, **3 week** training using **40 cores**
- 24% and 8% relative improvement (resp.) over traditional smoothed n-gram LMs
- in terms of test *set perplexity*: geometric average

$$1/\widehat{\mathrm{P}}(\mathrm{w_t} \mid \mathrm{w_{t-n+1}} \dots \mathrm{w_{t-1}})$$

- Due to **complexity**, NNLM can't be applied to large data sets → poor performance on rare words
- Bengio et al. (2003) initially thought their main contribution was a more accurate LM. They let the interpretation and use of the word vectors as **future work**

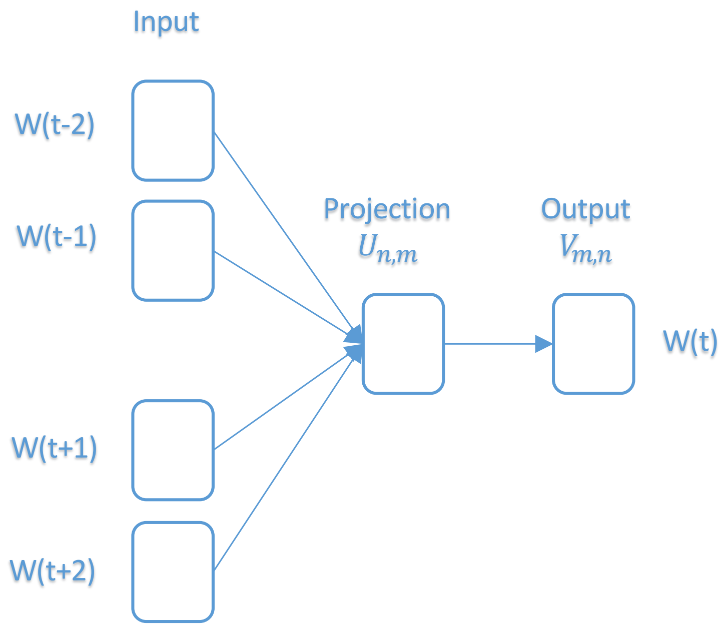
- On the opposite, Mikolov et al. (2013) focus on the **word vectors**

Word2Vec

- Mikolov et al. in 2013
- Word2vec key idea: achieve better performance not by using a more complex model (i.e., with more layers), but by allowing a **simpler (shallower) model** to be trained on **much larger amounts of data**
- no hidden layer (leads to 1000X speedup)
- projection layer is shared (not just the weight matrix) - C
- context: words from both history & future:
- Two algorithms for learning words vectors:
  - **CBOW**: from context predict target
  - **Skip-gram**: from target predict context

W2V: Continuous Bag Of Words – CBOW

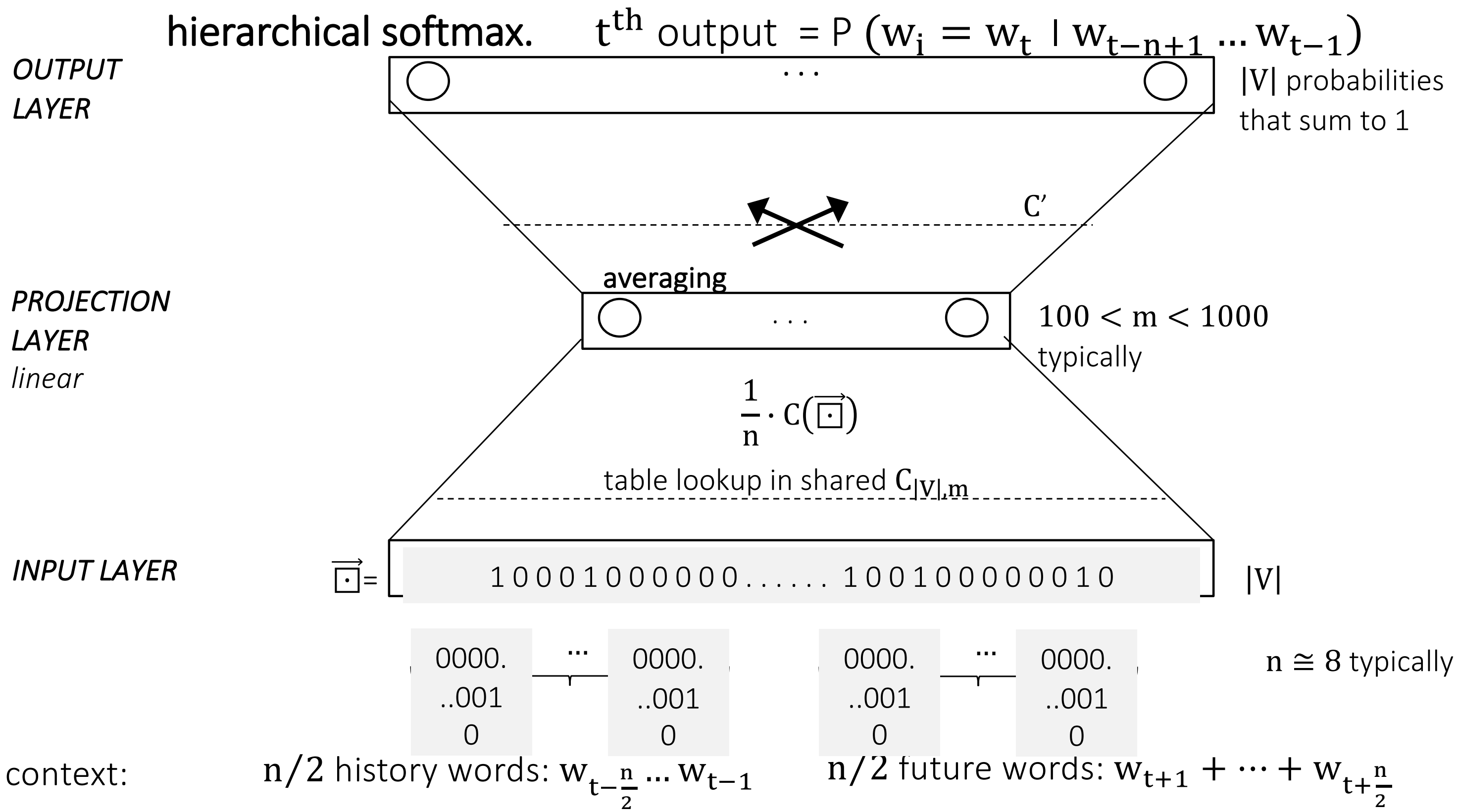
- An unsupervised technique to learn word embeddings.
- CBOW learns the embeddings by predicting the target word (the center one) based on the context words (surrounding words).
- i.e. “Paris is the capital of France located in Ile de France”



Continuous Bag-of-Words (CBOW)

For each training sequence: input = (context, target) pair: $(w_{t-\frac{n}{2}} \dots w_{t-1} w_{t+1} \dots w_{t+\frac{n}{2}}, w_t)$

objective: minimize $-\log\widehat{P}(w_t \mid w_{t-n+1} \dots w_{t-1})$



W2V: Continuous Bag Of Words – CBOW : Forward

- Each word **W(t)** is represented by one-hot vector of size **n** (vocabulary size).
- The **w** context words are averaged and forwarded to the projection layer to produce an embedded vector **z** of size **m:**

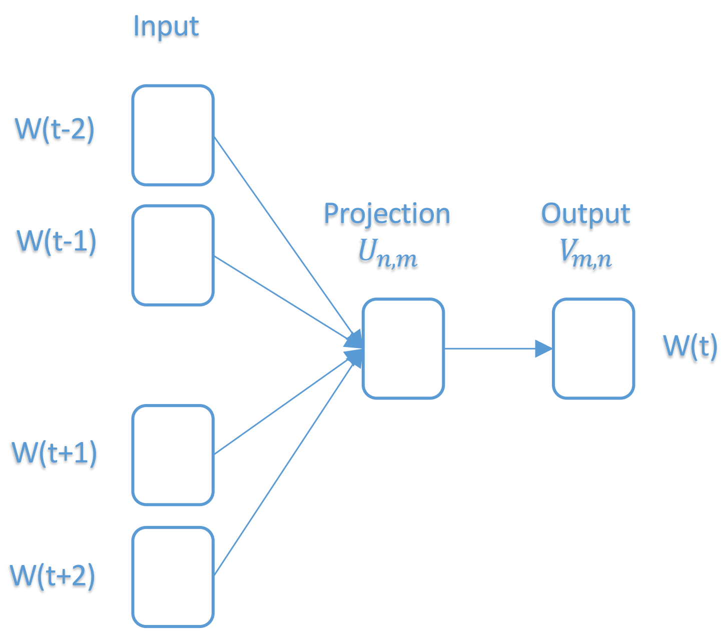
$$z_{1,m} = \frac{1}{w} \sum_{W \epsilon context} W_{1,n} U_{n,m}$$

- The vector **z** is forwarded to an output projection layer that produce the out vector **y** of size **n**.

$$y_{1,n} = z_{1,m} V_{m,n}$$

- Finally, a soft-max activation function is applied to the output to find a vector of probabilities for each word:

$$\sigma(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

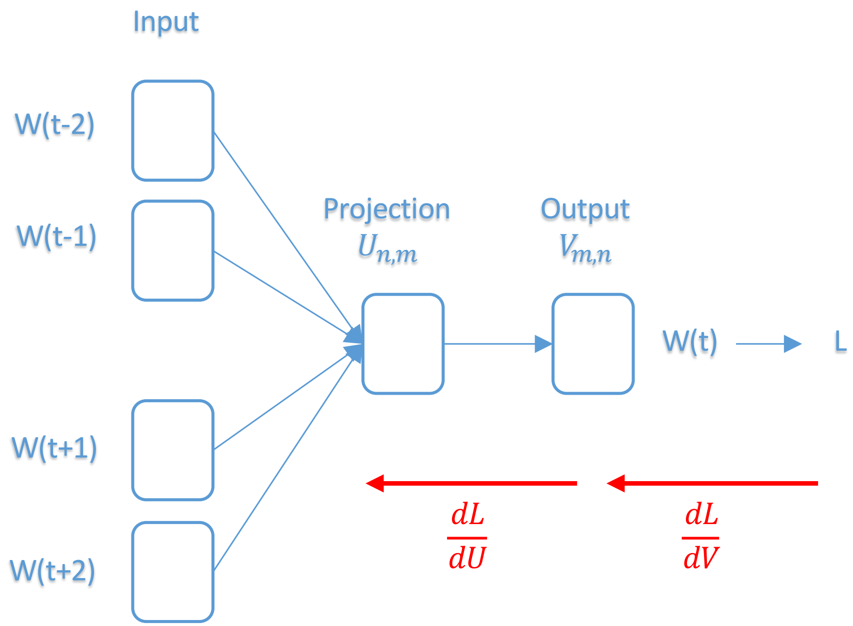


W2V: Continuous Bag Of Words – CBOW : Backward

- To update the weights, first we compute the log loss function (cross-entropy):

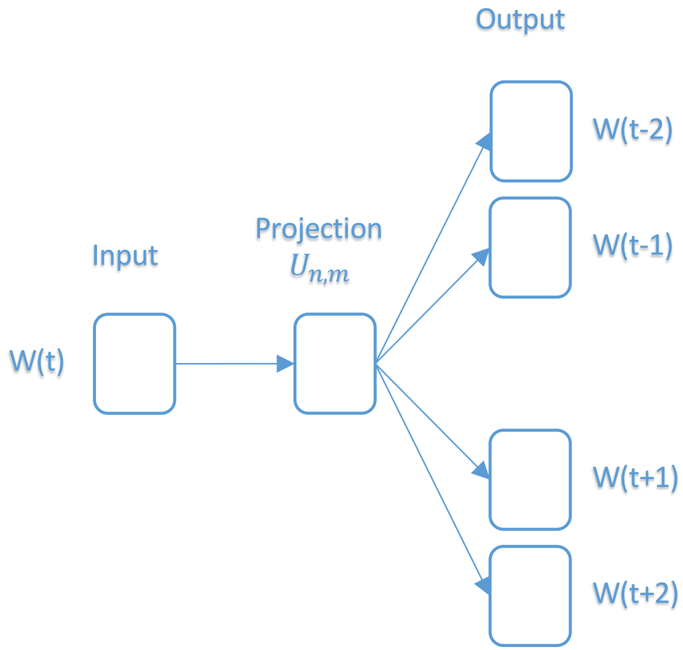
$$L = -\frac{1}{n}\sum_{i=1}^{n} W(t)_i \log(\sigma(y_i))$$

- The weights (matrices U and V) are now updated using gradient descent with learning rate α.
- $V = V - \alpha \frac{dL}{dV}$
- $U = U - \alpha \frac{dL}{dU}$
- Finally, after multiple passes through the corpus, **U** is the final **Word Embeddings Matrix** where each row represent a vector of size **m** for a specific word.



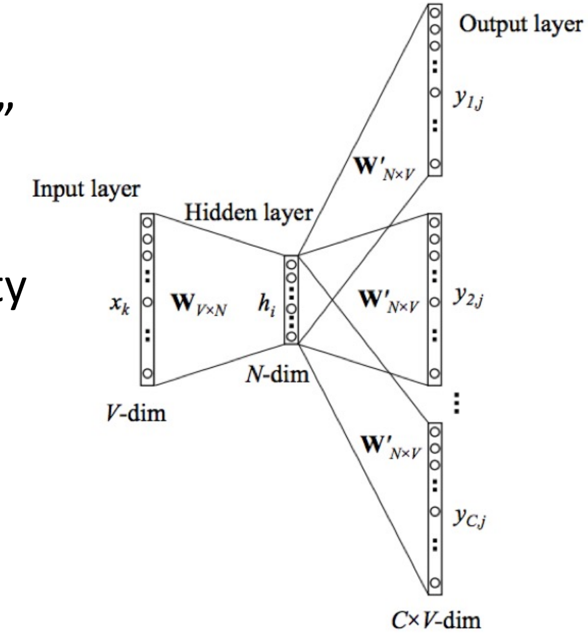
W2V: Skip-Gram

- An unsupervised technique to learn word embeddings.
- Skip-gram learn the embeddings by predicting the context of the word.
- The used loss function is cross-entropy as in CBOW.



W2V: Skip-gram

- skip-gram uses the context’s center word to predict the surrounding words
- i.e. “Paris is the capital of France located in Ile de France”
- instead of computing the probability of the target word w_t given its previous words, we calculate the probability of the surrounding word w_{t+j} given w_t
- $p(w_{t+j}|w_t) = \frac{\exp(\boldsymbol{v}_{\boldsymbol{w_t}}^{\boldsymbol{T}} \boldsymbol{v}'_{\boldsymbol{w_{t+j}}})}{\sum_{w \in V} \exp(\boldsymbol{v}_{\boldsymbol{w_t}}^{\boldsymbol{T}} \boldsymbol{v}'_{\boldsymbol{w_{t+j}}})}$
- $v^T{}_{wt}$ is a column of W_{VxN} and $v'_{w_{t+j}}$ is a column of $W'{}_{NxV}$
- Objective function $J = \frac{1}{T}\sum_{t=1}^{T} \sum_{-n \le j \le n} \log p(w_{t+j}|w_t)$



Efficient Estimation of Word Representations in Vector Space- Mikolov et al. 2013

Word2vec facts

- Complexity is $\mathbf{n} * \mathbf{m} + \mathbf{m} * \mathbf{log}|\mathbf{V}|$ (Mikolov et al. 2013a)
- **n**:size of context window (~10) **nxm**: dimensions of the projection layer, **|V|** size of the vocabulary
- On Google news 6B words training corpus, with $|\mathbf{V}| \sim 10^6$:
  - CBOW with $\mathrm{m} = 1000$ took **2 days** to train on **140 cores**
  - Skip-gram with $\mathrm{m} = 1000$ took **2.5 days** on **125 cores**
  - NNLM (Bengio et al. 2003) took **14 days** on **180 cores**, for $\mathrm{m} = 100$ only!

word2vec training speed ≅ 100K-5M words/s

- Quality of the word vectors:
  - ↗ significantly with **amount of training data** and **dimension of the word vectors** (m)
  - measured in terms of accuracy on 20K semantic and syntactic association tasks.

    e.g., words in **bold** have to be returned:

Capital-Country	Past tense	Superlative	Male-Female	Opposite
Athens: **Greece**	walking: **walked**	easy: **easiest**	brother: **sister**	ethical: **unethical**

- Best NNLM: 12.3% overall accuracy. Word2vec (with Skip-gram): 53.3%
- References: http://www.scribd.com/doc/285890694/NIPS-DeepLearningWorkshop-NNforText#scribd ---- https://code.google.com/p/word2vec/

GloVe

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

- Ratio of co-occurrence probabilities best distinguishes relevant words

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$$

- *Cast this into a lease square problem:*

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}.$$

- X co-occurrence matrix
- f weighting function,
- b bias terms
- $w_i = word\ vector$
- $\widetilde{w_j} = context\ vector$

model that utilizes
- count data
- bilinear prediction-based methods like word2vec

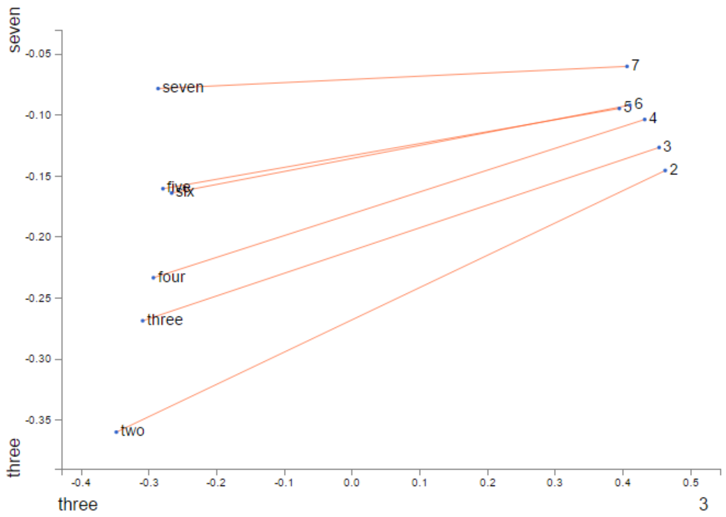
https://nlp.stanford.edu/pubs/glove.pdf

Which is better?

- Open question
- SVD vs word2vec vs GloVe
- All based on co-occurrence
- *Levy, O., Goldberg, Y., & Dagan, I. (2015)*
    - SVD performs best on similarity tasks
    - Word2vec performs best on analogy tasks
    - *No single algorithm consistently outperforms the other methods*
    - *Hyperparameter tuning is important*
    - 3 out of 6 cases, tuning hyperparameters is more beneficial than increasing corpus size
    - word2vec outperforms GloVe on all tasks
    - *CBOW is worse than skip-gram on all tasks*

Applications

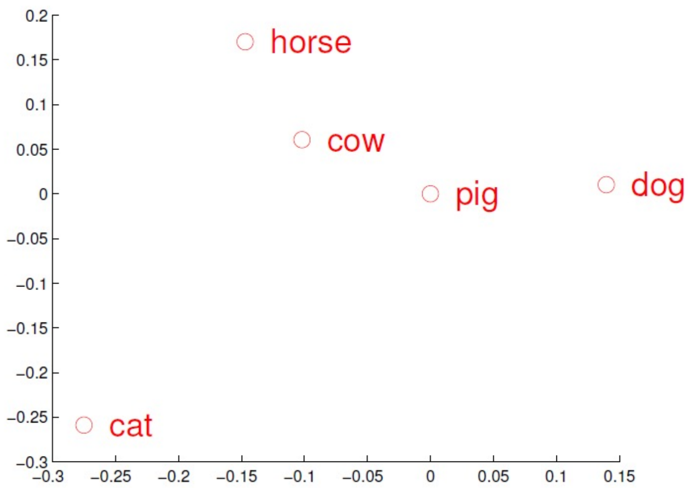
- Word analogies
- Find similar words
  - Semantic similarity
  - Syntactic similarity
- POS tagging
- Similar analogies for different languages
- Document classification

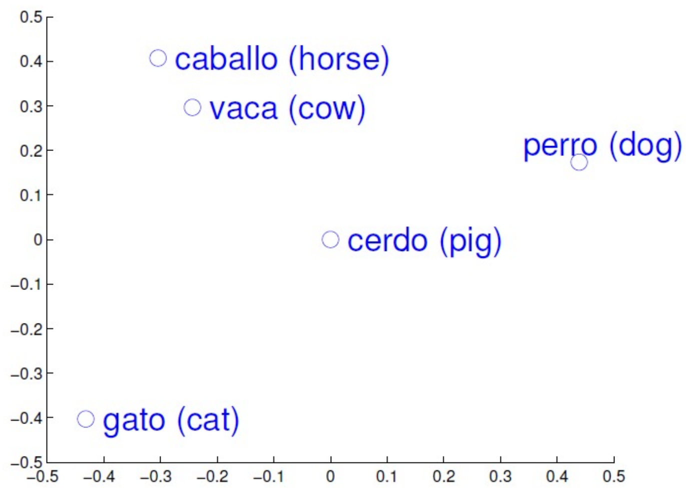


https://lamyiowce.github.io/word2viz/

Applications

- High quality word vectors boost performance of all NLP tasks, including document classification, machine translation, information retrieval…
- Example for English to Spanish machine translation:

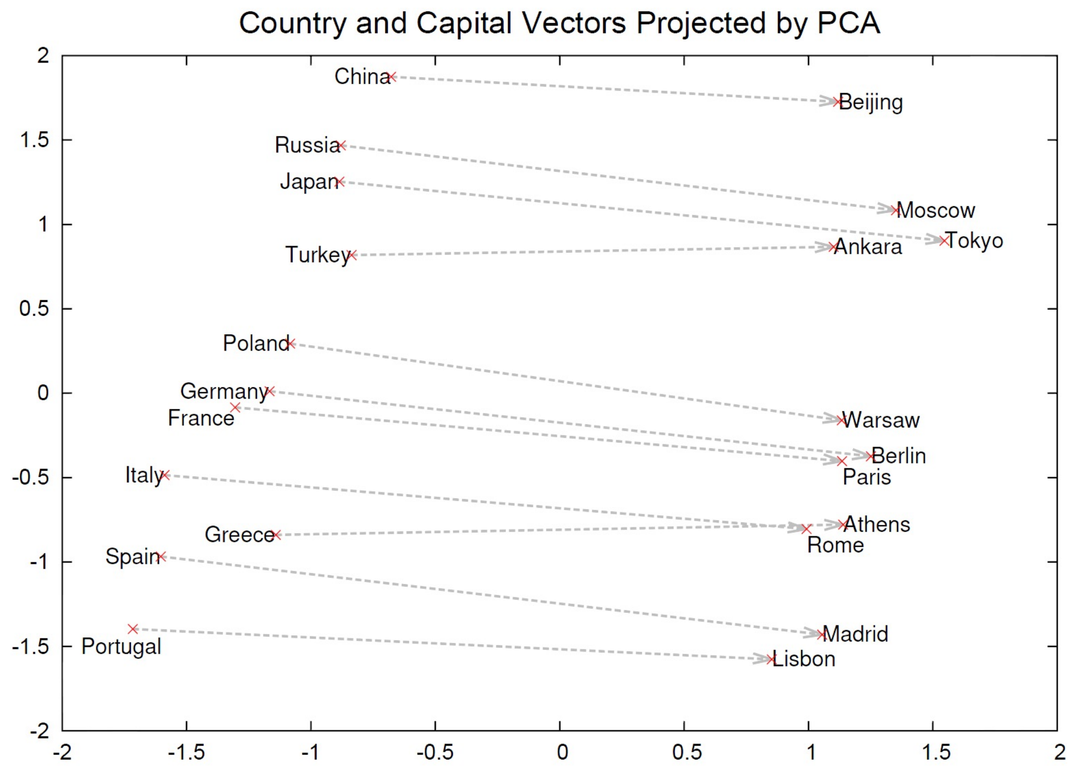




About 90% reported accuracy (Mikolov et al. 2013c)

[Mikolov, T., Le, Q. V., & Sutskever, I. (2013). Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168.*]

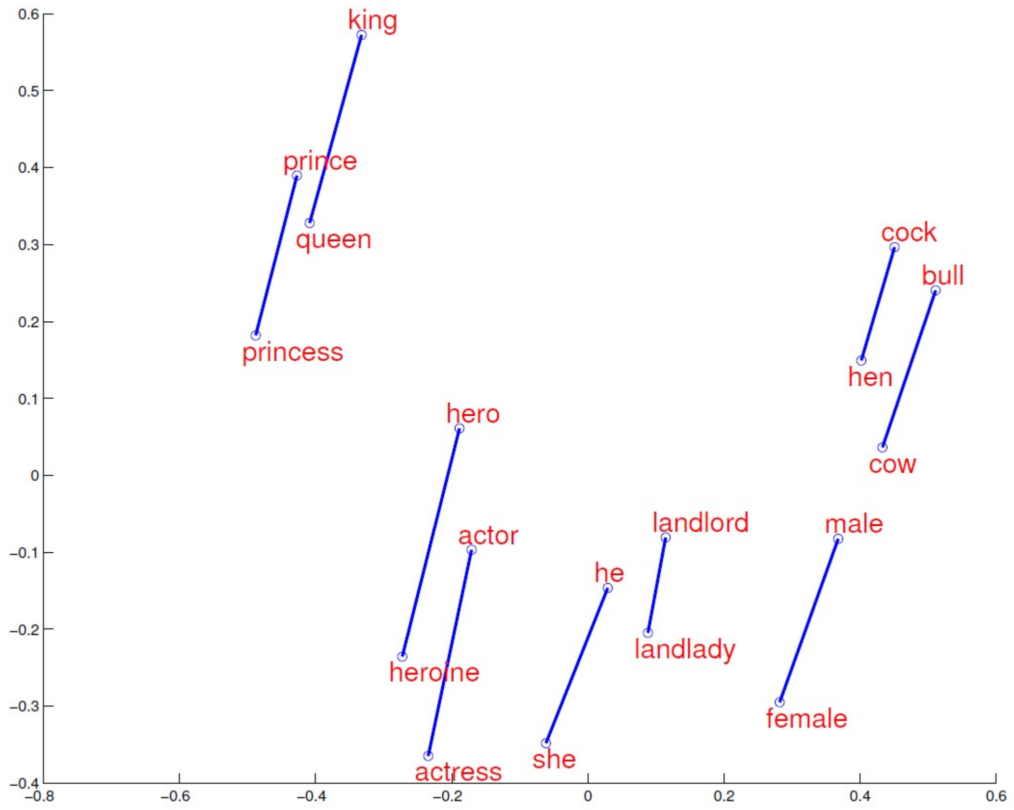
Remarkable properties of word vectors



regularities between words are encoded in the difference vectors
e.g., there is a constant **country-capital** difference vector

Mikolov et al. (2013b)
Distributed representations of words and phrases and their compositionality

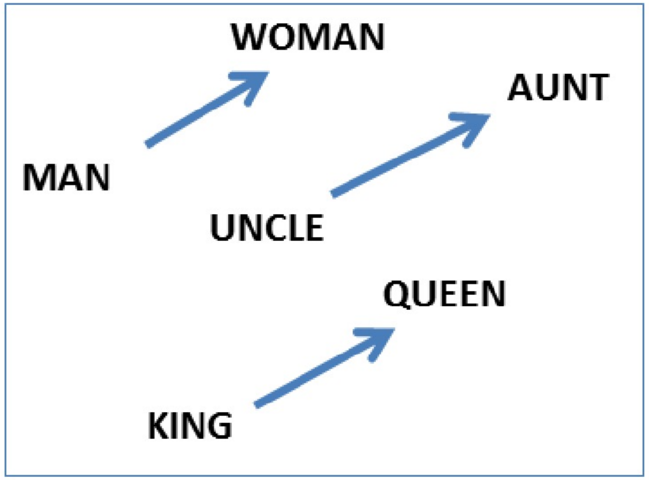
Remarkable properties of word vectors



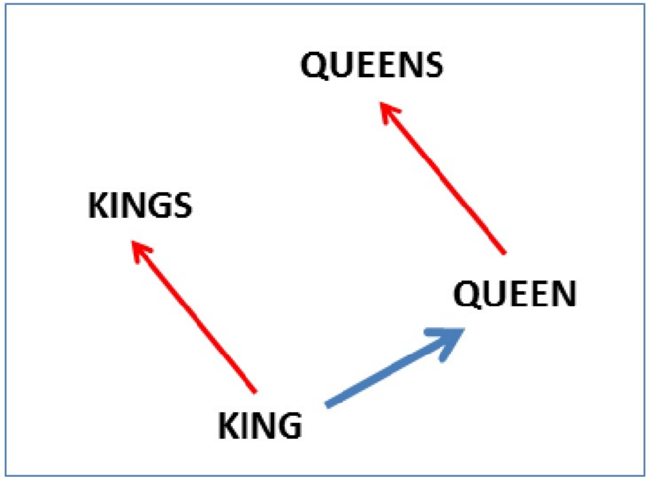
constant **female-male** difference vector

http://www.scribd.com/doc/285890694/NIPS-DeepLearningWorkshop-NNforText#scribd

Remarkable properties of word vectors



constant **male-female** difference vector



constant **singular-plural** difference vector

- ➢ Vector operations are supported and make intuitive sense:

$$w_{king} - w_{man} + w_{woman} \cong w_{queen}$$

$$w_{einstein} - w_{scientist} + w_{painter} \cong w_{picasso}$$

$$w_{paris} - w_{france} + w_{italy} \cong w_{rome}$$

$$w_{his} - w_{he} + w_{she} \cong w_{her}$$

$$w_{windows} - w_{microsoft} + w_{google} \cong w_{android}$$

$$w_{cu} - w_{copper} + w_{gold} \cong w_{au}$$

- ➢ Online demo (scroll down to end of tutorial)

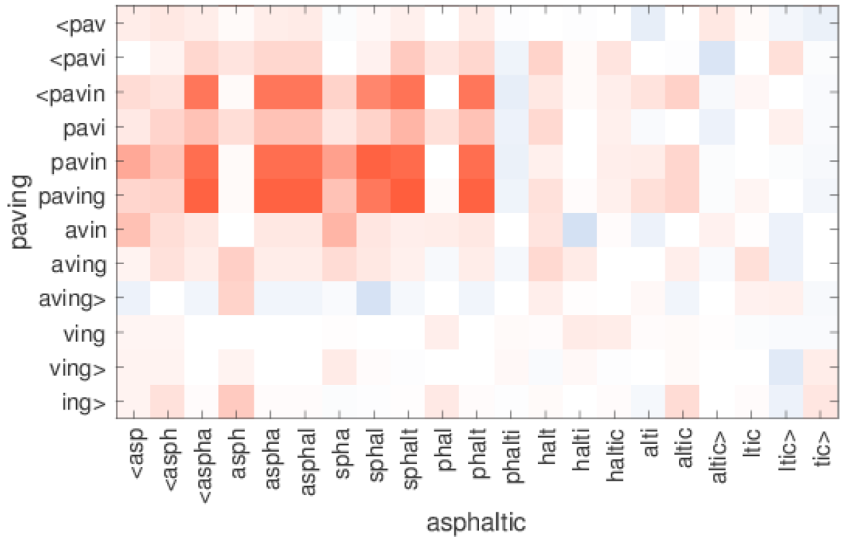
http://rare-technologies.com/word2vec-tutorial/

OUTLINE

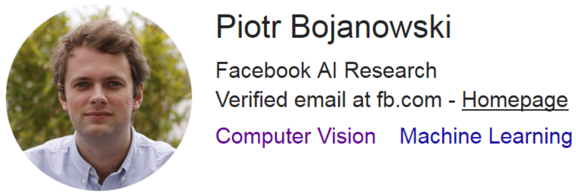
- Representation Learning for Text
  - SVD
  - Word2Vec
  - **Fast Text Embeddings – subword information**
  - Document representations
- NLP tasks and evaluation
  - GLUE, FLUE
  - French Linguistics

Enriching Word Vectors with Subword Information

FAIR 2017, 860 cites

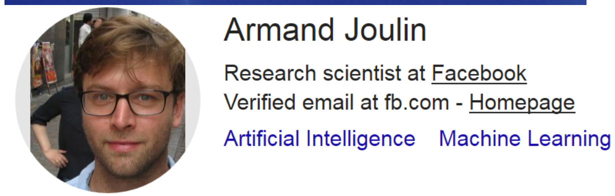


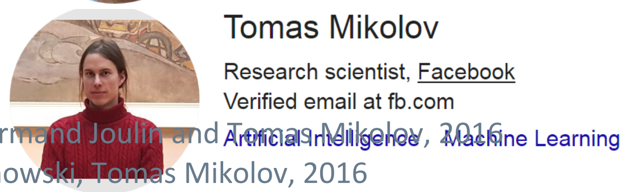
X'11-MVA'12



X'9-MVA'10

X'8-MVA'9





- *Enriching Word Vectors with Subword Information,* Piotr Bojanowski, Edouard Grave, Armand Joulin and Tomas Mikolov, 2016
- *Bag of Tricks for Efficient Text Classification,* Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, 2016

Enriching Word Vectors with Subword Information

Simple problem: word2vec/glove etc. ignore the internal structure of words

E.g., knowledge about *luck* is not used when learning a representation for *unlucky or luckily*

=> parameters are not shared => difficult to learn good vectors for rare words, and impossible for out-of-vocabulary words

Simple solution: learn vectors for character n-grams. Compose word vectors from their n-gram vectors.

Enriching Word Vectors with Subword Information

Quick recap: in skip-gram (Mikolov et al. 2013), the objective is to maximize:

$$\sum_{t=1}^{T} \sum_{c \in \mathcal{C}_t} \log p(w_c \mid w_t)$$

$w_1, \ldots, w_T$: the training corpus
C_t : set of indices of the context words around w_t
In English: the objective is to *predict well the context of a word given this word*

$p(w_c \mid w_t)$ is parameterized by the word vectors through a scoring function ***s***

$$s(w_t, w_c) = \mathbf{u}_{w_t}^{\top} \mathbf{v}_{w_c}$$

u and **v** above are taken from the input and output embedding matrices, resp.

Enriching Word Vectors with Subword Information

Proposed approach:

Each word is represented as a bag of **character n-grams**. E.g., for the word *where* and n=3:

<wh, whe, her, ere, re>

The < and > characters are added at the beginning and end of the word to keep prefix/suffix information.

New scoring function: $$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c$$

G_w is the set of character n-grams in word w. z_g is the vector of the g^{th} n-gram
V_c is the vector of the context word c
=> w is represented as the sum of its n-gram vectors

Enriching Word Vectors with Subword Information

Quantitative results: word similarity and word analogy tasks in 7 languages

- similarity: better than original skipgram and CBOW on 6/7 datasets
- analogy: - improves on original skipgram and CBOW for syntactic tasks
  - no improvement for semantic tasks

Qualitative results:

query	tiling	tech-rich	english-born	micromanaging	eateries	dendritic
`sisg`	tile flooring	tech-dominated tech-heavy	british-born polish-born	micromanage micromanaged	restaurants eaterie	dendrite dendrites
`sg`	bookcases built-ins	technology-heavy .ixic	most-capped ex-scotland	defang internalise	restaurants delis	epithelial p53

Nearest neighbors of rare words using subword (sisg) and original (sg) skipgram

Enriching Word Vectors with Subword Information

	anarchy	chy	<anar	narchy
	monarchy	monarc	chy	<monar
	kindness	ness>	ness	kind
	politeness	polite	ness>	eness>
EN	unlucky	<un	cky>	nlucky
	lifetime	life	<life	time
	starfish	fish	fish>	star
	submarine	marine	sub	marin
	transform	trans	<trans	form
	finirais	ais>	nir	fini
FR	finissent	ent>	finiss	<finis
	finissions	ions>	finiss	sions>

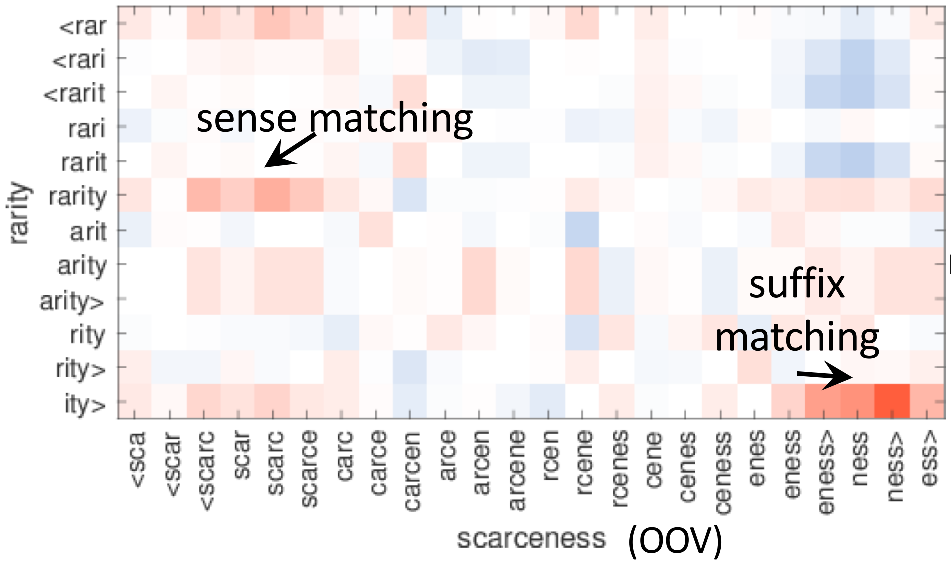
Most important character n-grams for selected words

Observations:

the most important n-grams tend to make sense and match:

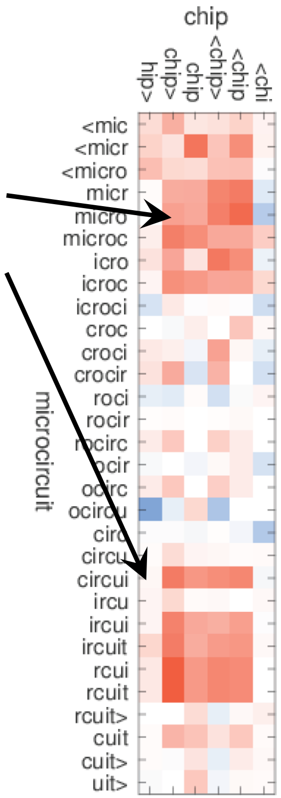
- prefixes & suffixes
- morphemes
- verb inflections

Enriching Word Vectors with Subword Information



(OOV)

The areas corresponding to *micro* and *circuit* match *chip* the most



(OOV)

Similarity between n-grams of in and out of vocabulary words

Observation: matches between n-grams are meaningful.

=> high quality vectors can be constructed for the OOV words (by summing the vectors of the n-grams)

OUTLINE

- Representation Learning for Text
  - SVD
  - Word2Vec
  - Fast Text Embeddings – subword information
- **CNNs for text classification**
- Attention based architectures
- NLP tasks and evaluation
  - GLUE, FLUE
  - French Linguistics

Convolutional Neural Networks

- In 1995, Yann LeCun and Yoshua Bengio introduced the concept of convolutional neural networks.
- Neuro-biologically motivated by the findings of locally sensitive and orientation-selective nerve cells in the visual cortex.
- They designed a network structure that implicitly extracts relevant features.
- Convolutional Neural Networks are a special kind of multi-layer neural networks.

[2] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. 3, 4

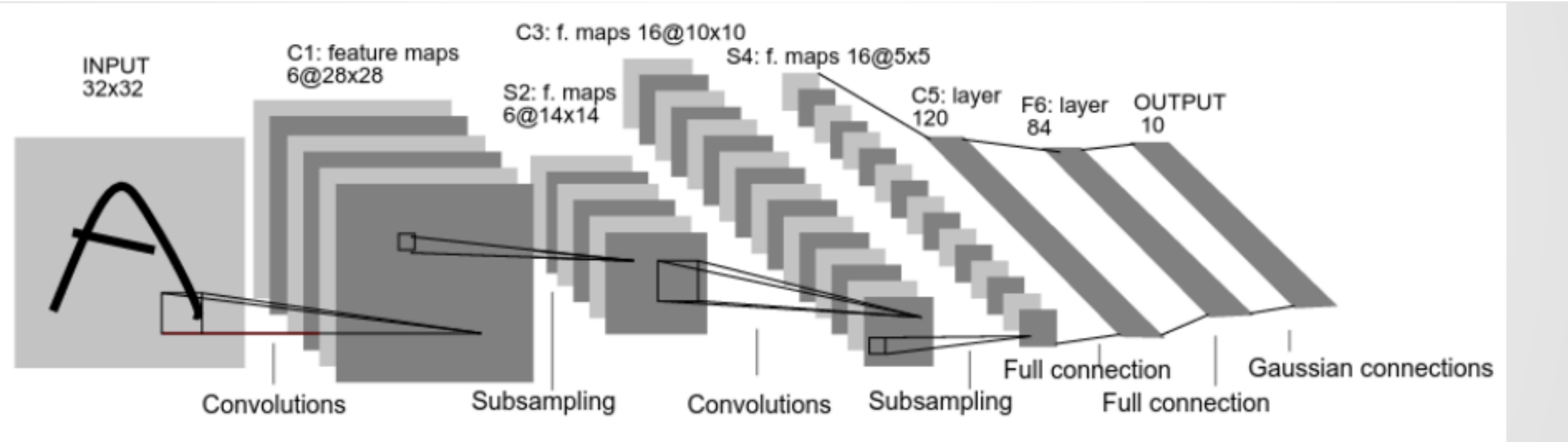
Convolutional Neural Networks

- inspired by studies of the cat's visual cortex [1], developed in computer vision to work on regular grids such as images [2].
- Feed-forward NNs , each neuron receives input from a neighborhood of the neurons (receptive fields) in the previous layer.
- Receptive fields, allow CNNs to recognize complex patterns in a hierarchical way, by combining lower-level, elementary features into higher-level features *compositionality*.
  - raw pixels=> edges =>shapes =>objects.
- absolute positions of features in the image are not important – only useful respective positions is useful composing higher-level patterns.
- Model detect a feature regardless of its position in the image - local invariance.
- Compositionality, local invariance two key concepts of CNNs.

[1] Hubel, David H., and Torsten N. Wiesel (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology 160.1:106-154. 4

[2] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. 3, 4

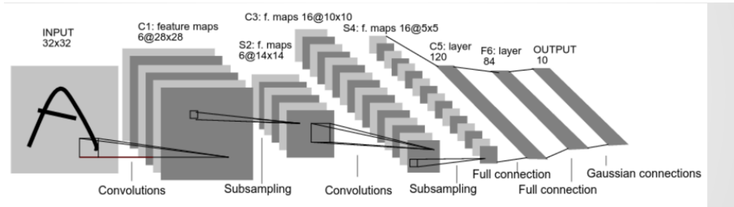
CNN architecture



Lenet-5 (Lecun-98), Convolutional Neural Network for digits recognition

Feature maps

- Feature Map - Obtained by convolution of the feature matrix with a linear filter, adding a bias term and applying a non-linear function
- Non-linear functions:
  - Sigmoid $$\frac{1}{1+e^{-x}}$$
  - Tanh $$\frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Lenet-5 (Lecun-98), Convolutional Neural Network for digits recognition

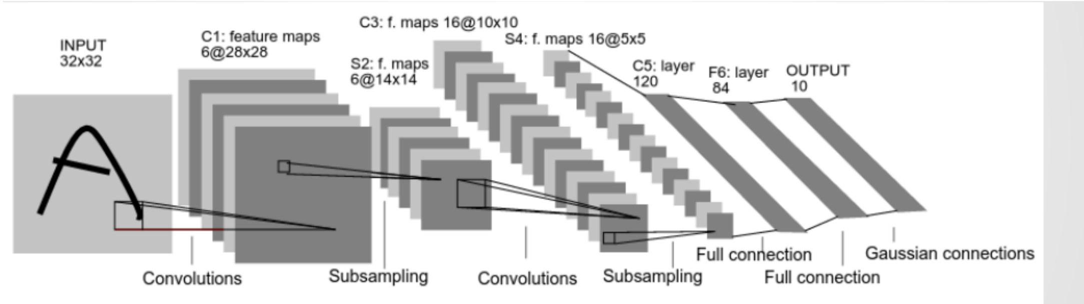
  - Rectified Linear Unit (ReLU) -> Most popular choice avoids saturation issues, makes learning faster

$$f(x) = \max(0, x)$$

- Require a number of such feature maps at each layer to capture sufficient features

Pooling

- Sub-sampling layer
- Variants:
  - Max pooling
  - Weighted average
  - L2 norm of neighborhood
- Provides translation invariance
- Reduces computation



Lenet-5 (Lecun-98), Convolutional Neural Network for digits recognition

CNN for Text Classification

- Use the word embeddings of the document terms as input for Convolutional Neural Network
- Input must be fixed size
- Applies multiple filters to concatenated word vectors
- Produces new features for every filter
- picks the max as a feature for the CNN

CNN architecture for document classification

- Use the high quality embeddings as input for Convolutional Neural Network
- Applies multiple filters to concatenated word vector

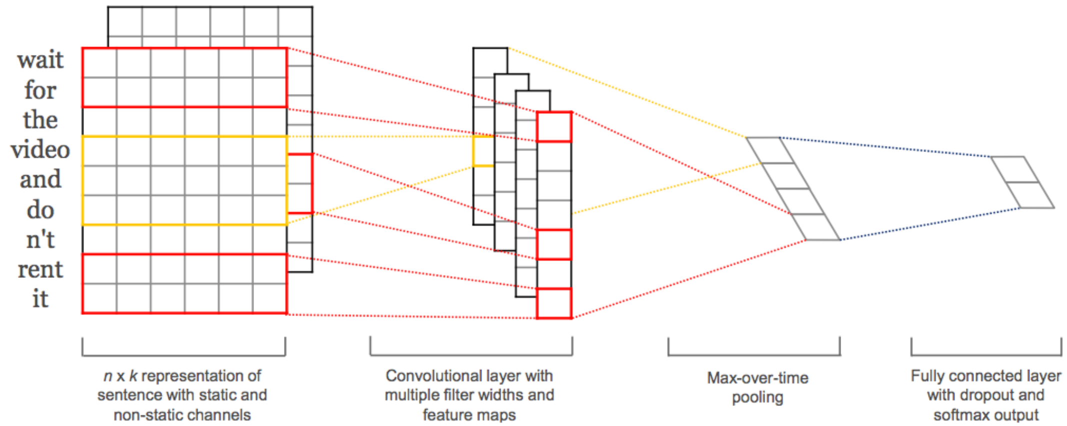
$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$$

- Produces new features for every filter

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$$

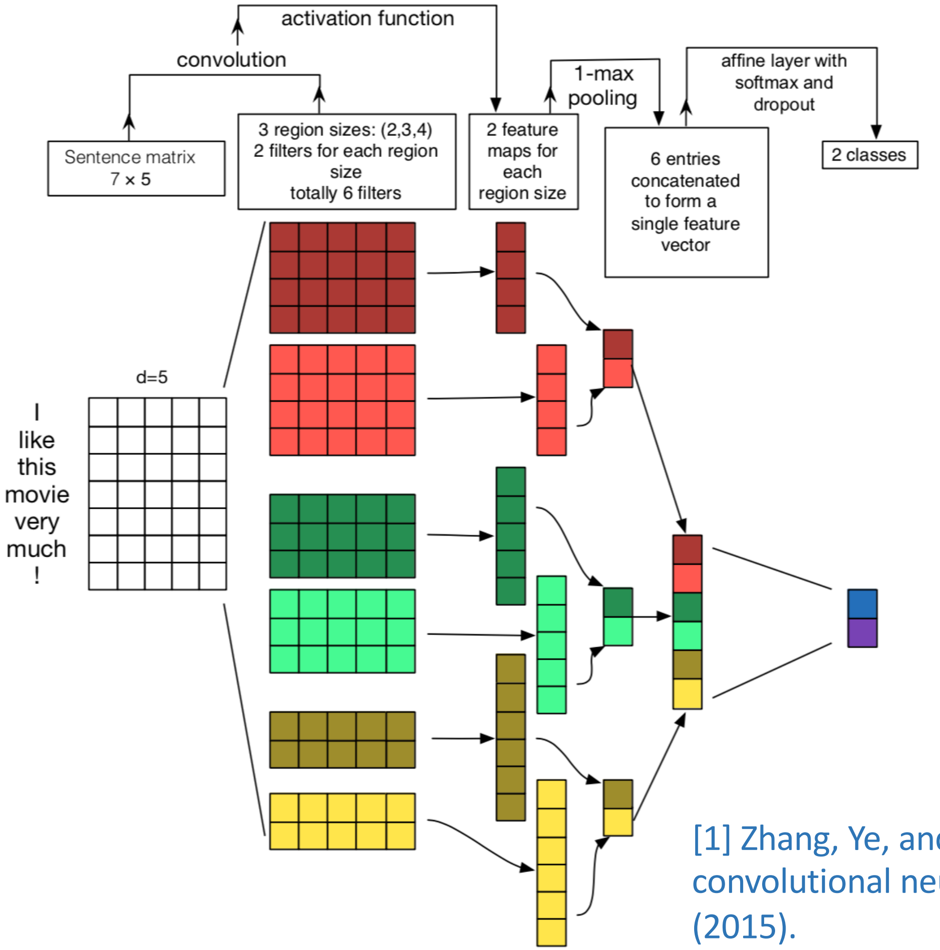
- And picks the max as a feature for the CNN

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \qquad \hat{c} = \max\{\mathbf{c}\}$$



Yoon Kim - Convolutional Neural Networks for Sentence Classification

CNN architecture for document classification [1]



- Data (text) only 1st column of input
- Rest of each row: embedding (in images 2D+RGB dimension)
- Filters of different sizes (4x5, 3x5 etc.)
  - Each size captures different features (need ~10^2 filters/size)
- Feature maps:
  - As many as the times filter fits on data matrix
- Max pooling maintains the “best features”
- Global feature map => classification via softmax

[1] Zhang, Ye, and Byron Wallace. "A sensitivity analysis of (and practitioners’ guide to) convolutional neural networks for sentence classification." arXiv preprint arXiv:1510.03820 (2015).

CNN for text classification

Many variations of the model [1]

- use existing vectors as input (CNN-static)
- learn vectors for the specific classification task through backpropagation (CNN-rand)
- Modify existing vectors for the specific task through backpropagation(CNN-non-static)

[1] Y. Kim, Convolutional Neural Networks for Sentence Classification, EMNLP 2014

CNN for text classification

- Combine multiple word embeddings
- Each set of vectors is treated as a ‘channel’
- Filters applied to all channels
- Gradients are back-propagated only through one of the channels
- Fine-tunes one set of vectors while keeping the other static

CNN for text classification

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Accuracy scores (Kim et al vs others)

CNN architecture for (short) document classification – T-SNE visualization (see Lab notes)

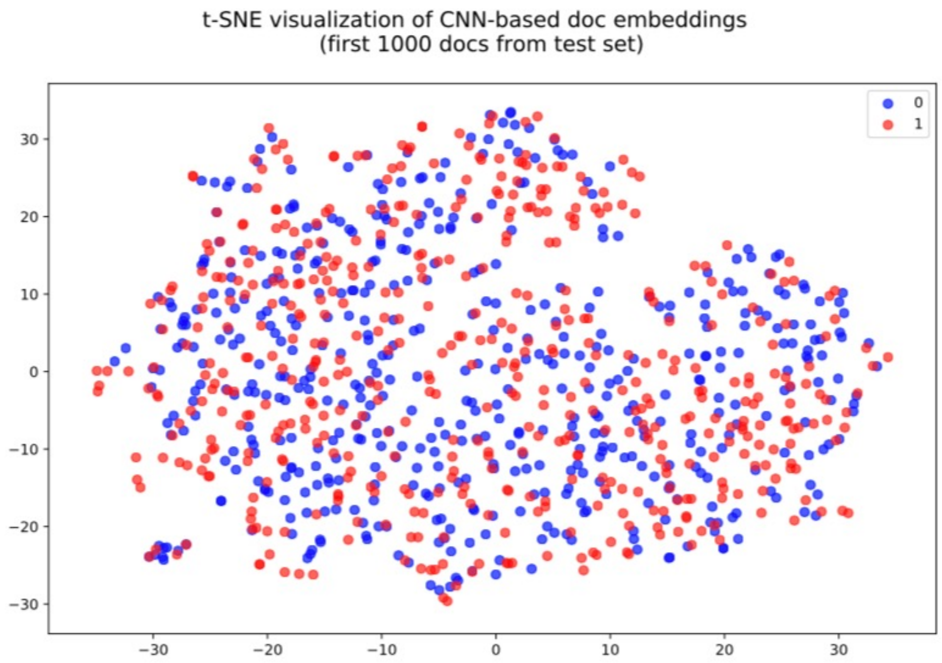


Figure 2: Doc embeddings before training.

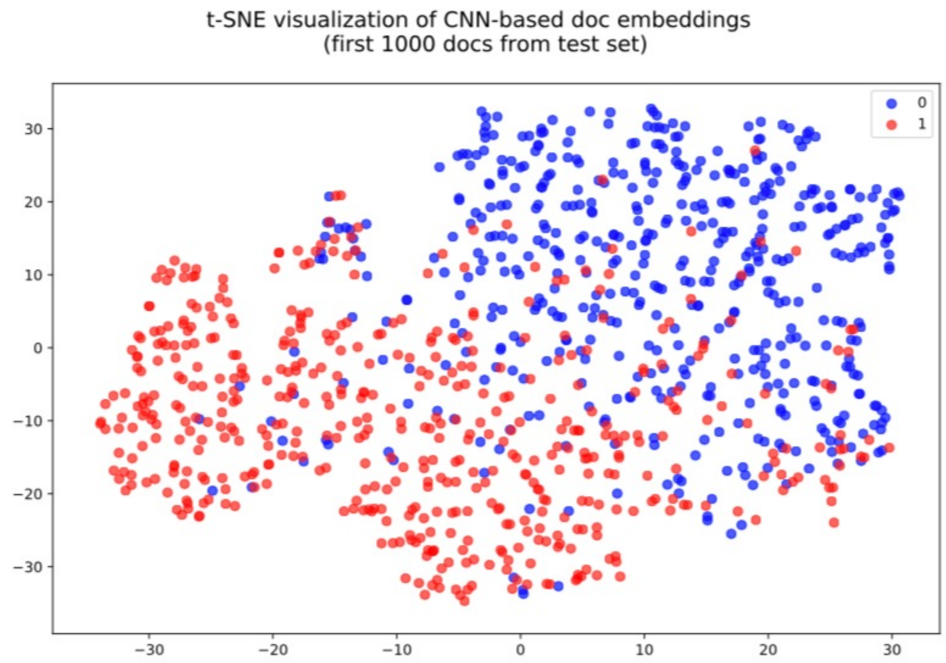


Figure 3: Doc embeddings after 2 epochs.

CNN architecture for document classification - Saliency maps (see Lab notes)

- words are most related to changing the doc classification
- A in R^{sxd}, s:# sentence words, d:size of embeddings

$$\text{saliency}(a) = \left| \frac{\partial(\text{CNN})}{\partial a} \right|_a$$

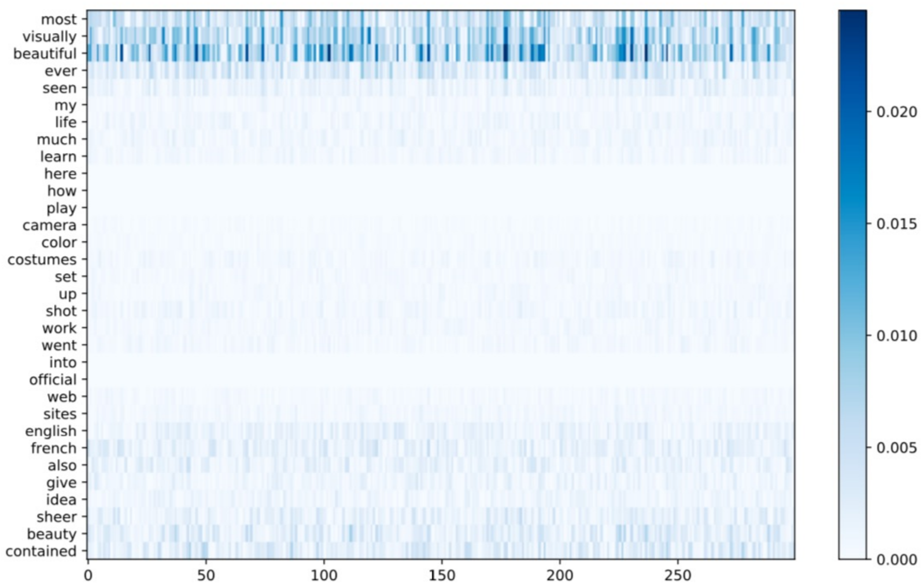


Figure 4: Saliency map for document 1 of the IMDB test set (true label: positive)

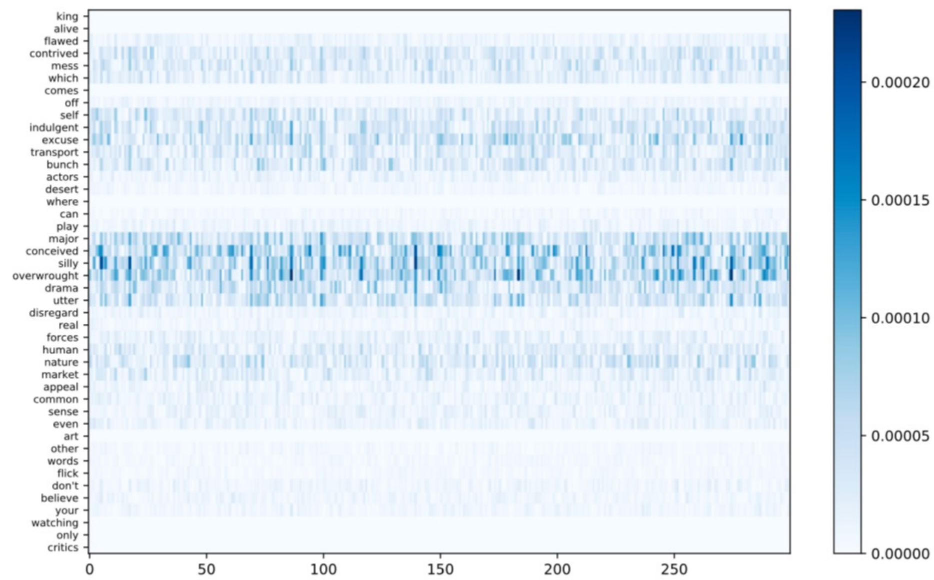
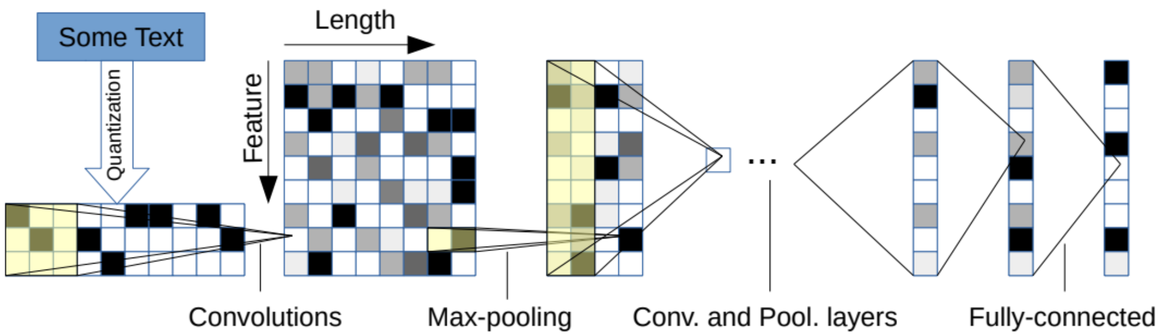


Figure 5: Saliency map for document 15 of the IMDB test set (true label: negative)

Character-level CNN for Text Classification

- Input: sequence of encoded characters
- quantize each character using “one-hot” encoding
- input feature length is 1014 characters
- 1014 characters able capture most of the texts of interest
- Also perform Data Augmentation using Thesaurus as preprocessing step

Model Architecture



- 9 layers deep
- 6 convolutional layers
- 3 fully-connected layers
- 2 dropout modules in between the fully-connected layers for regularization

Model Comparison

Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46
Bag-of-means	**16.91**	**10.79**	**9.55**	**12.67**	**47.46**	**39.45**	**55.87**	**18.39**
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Lg. w2v Conv.	9.92	4.39	1.42	4.60	40.16	31.97	44.40	5.88
Sm. w2v Conv.	11.35	4.54	1.71	5.56	42.13	31.50	42.59	6.00
Lg. w2v Conv. Th.	9.91	-	1.37	4.63	39.58	31.23	43.75	5.80
Sm. w2v Conv. Th.	10.88	-	1.53	5.36	41.09	29.86	42.50	5.63
Lg. Lk. Conv.	8.55	4.95	1.72	4.89	40.52	29.06	45.95	5.84
Sm. Lk. Conv.	10.87	4.93	1.85	5.54	41.41	30.02	43.66	5.85
Lg. Lk. Conv. Th.	8.93	-	1.58	5.03	40.52	28.84	42.39	5.52
Sm. Lk. Conv. Th.	9.12	-	1.77	5.37	41.17	28.92	43.19	5.51
Lg. Full Conv.	9.85	8.80	1.66	5.25	38.40	29.90	40.89	5.78
Sm. Full Conv.	11.59	8.95	1.89	5.67	38.82	30.01	40.88	5.78
Lg. Full Conv. Th.	9.51	-	1.55	4.88	38.04	29.58	40.54	5.51
Sm. Full Conv. Th.	10.89	-	1.69	5.42	**37.95**	29.90	40.53	5.66
Lg. Conv.	12.82	4.88	1.73	5.89	39.62	29.55	41.31	5.51
Sm. Conv.	15.65	8.65	1.98	6.53	40.84	29.84	40.53	5.50
Lg. Conv. Th.	13.39	-	1.60	5.82	39.30	**28.80**	40.45	**4.93**
Sm. Conv. Th.	14.80	-	1.85	6.49	40.16	29.84	**40.43**	5.67

Testing errors for all models
Blue->best, Red->worst

links

- http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
- https://arxiv.org/pdf/1509.01626.pdf
- http://www.aclweb.org/anthology/D14-1181
- http://cs231n.github.io/convolutional-networks/
- http://ufldl.stanford.edu/tutorial/supervised/Pooling/

OUTLINE

- Representation Learning for Text
  - SVD
  - Word2Vec
  - Fast Text Embeddings – subword information
  - Document Representations
- CNNs for text classification
- **Attention based architectures**
- Transformer Architecture
- NLP tasks and evaluation
  - GLUE, FLUE
  - French Linguistics

Context & Senses

- **“a word is defined by “the company it keeps” (Firth, 1957)**
- Words are *ambiguous*:
  - The amount of deposits in the banks decreased by 3.5% in March
  - The trees on bank of the river were offering their shade to the visitors
- We need different vectors to represent all word/token senses

Deep Contextualized Word Representations

Best paper NAACL 2018

transfer learning has been used in vision since 2012 (ImageNet)

it is used in NLP since 2018! [1] [2]

ELMo: Embeddings from Language Models

Initial problem: traditional word vectors map each word to a single, context-independent vector

But some words have more than one sense! (polysemy)

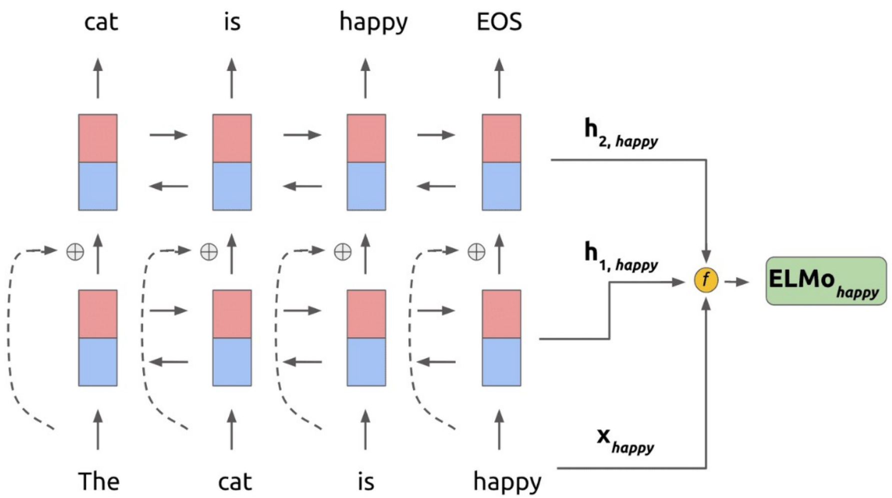
e.g., bank, get, wood, play, mean…

Solution:

- token assigned a representation based on entire input sentence
- use the internal representations of a RNN language model pre-trained on a large dataset

Deep Contextualized Word Representations

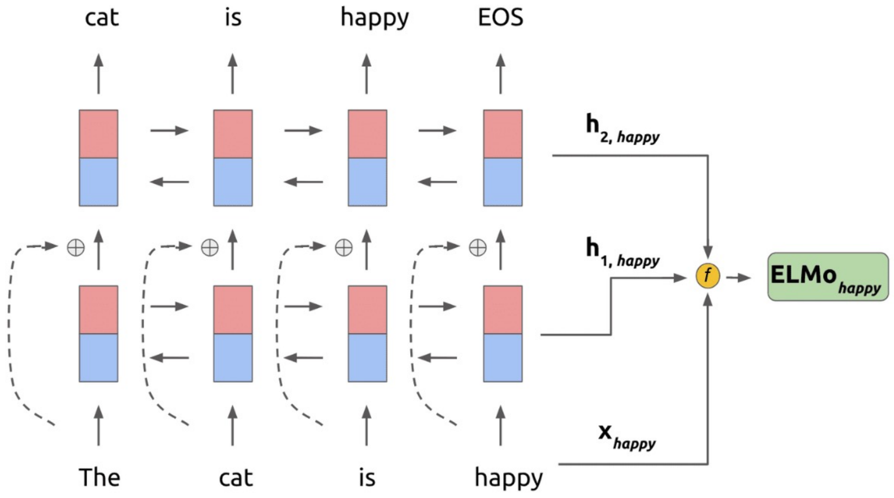
- bidirectional LSTM is trained with a coupled language model (LM) on a large text corpus - ELMo (Embeddings from Language Models) representations.
  - Forward $p(t_1, t_2..., t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2..., t_N)$
  - Bacward $p(t_1, t_2..., t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}..., t_{k+N})$



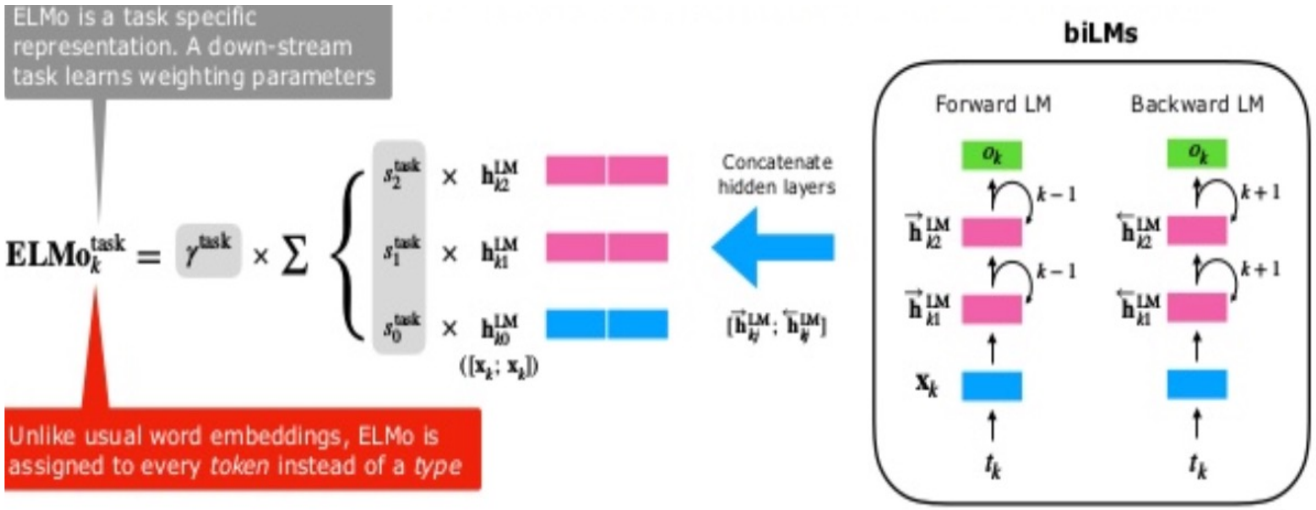
Graphic from: https://www.mihaileric.com/posts/deep-contextualized-word-representations-elmo/

Deep Contextualized Word Representations

- ELMo representations are deep: function of all of the internal layers of the biLM.
  - learn a linear combination of the vectors stacked above each input word for each end task,
  - Combining the internal states in this manner allows for very rich word representations.
  - higher-level LSTM states ′context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised word sense disambiguation tasks)
  - lower level states model aspects of syntax (e.g., they can be used to do part-of-speech tagging).
  - Simultaneously exposing all of these signals is highly beneficial, allowing the learned models select the types of semi-supervision that are most useful for each end task.



Deep Contextualized Word Representations



Semi-supervised approach:

- 1) a deep bidirectional RNN language model is pretrained on a large dataset
- 2) the vector of each word in a given input sentence is computed as a weighted sum of the RNN hidden states
- 1) is *unsupervised*, 2) weights are learned in a *supervised* way on some task-specific dataset

Graphic from: https://www.mihaileric.com/posts/deep-contextualized-word-representations-elmo/

$$\mathbf{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

$h_{k,j}^{LM}$ is the k^th^ hidden representation of the j^th^ layer of the bi-RNN LM

s^{task} is the softmax weight vector, γ^{task} is a scaling parameter (for optimization)

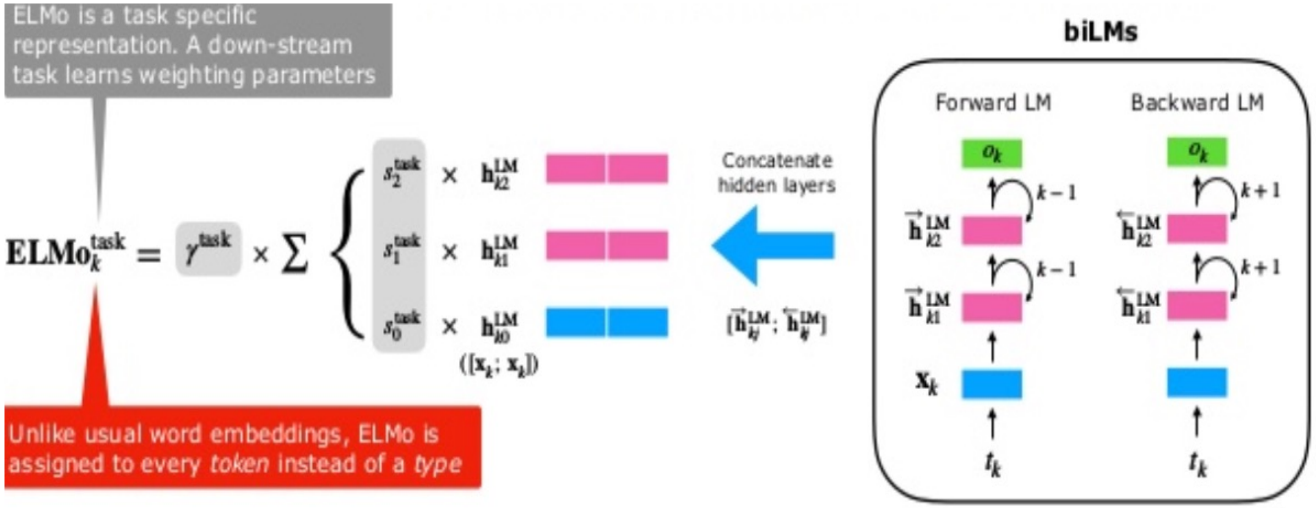
The authors use L=2 in all experiments

Deep Contextualized Word Representations

Use of ELMo in practice, on a task-specific dataset:

- 1) use the pretrained language model in prediction mode and store $h_{k,j}^{LM}$ for each word (each k) and each layer (each j)
- 2) concatenate $ELMo_k^{task}$ with the corresponding input vector* of whatever supervised model is used to solve the task (e.g., RNN, CNN, feed-forward…)
- 3) update s^{task} and γ^{task} with the other parameters of the supervised model during training

*given by word2vec, glove, etc.



Results:

ELMo improves over the baselines on 8 tasks, ranging from question answering to co-reference resolution, sentiment analysis, POS-tagging, and disambiguation

Deep Contextualized Word Representations

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	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%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

SQuAD: Question answering, SNLI: Entailment, SRL: Semantic Role Labelling ("Who did what to whom"), NER: Name entity recognition, SST: Sentiment Analysis, Coref: task of clustering mentions in text that refer to the same underlying real world entities

Deep Contextualized Word Representations

Model	$\mathbf{F_1}$
WordNet 1st Sense Baseline	65.9
Raganato et al. (2017a)	69.9
Iacobacci et al. (2016)	**70.1**
CoVe, First Layer	59.4
CoVe, Second Layer	*64.7*
biLM, First layer	67.4
biLM, Second layer	*69.0*

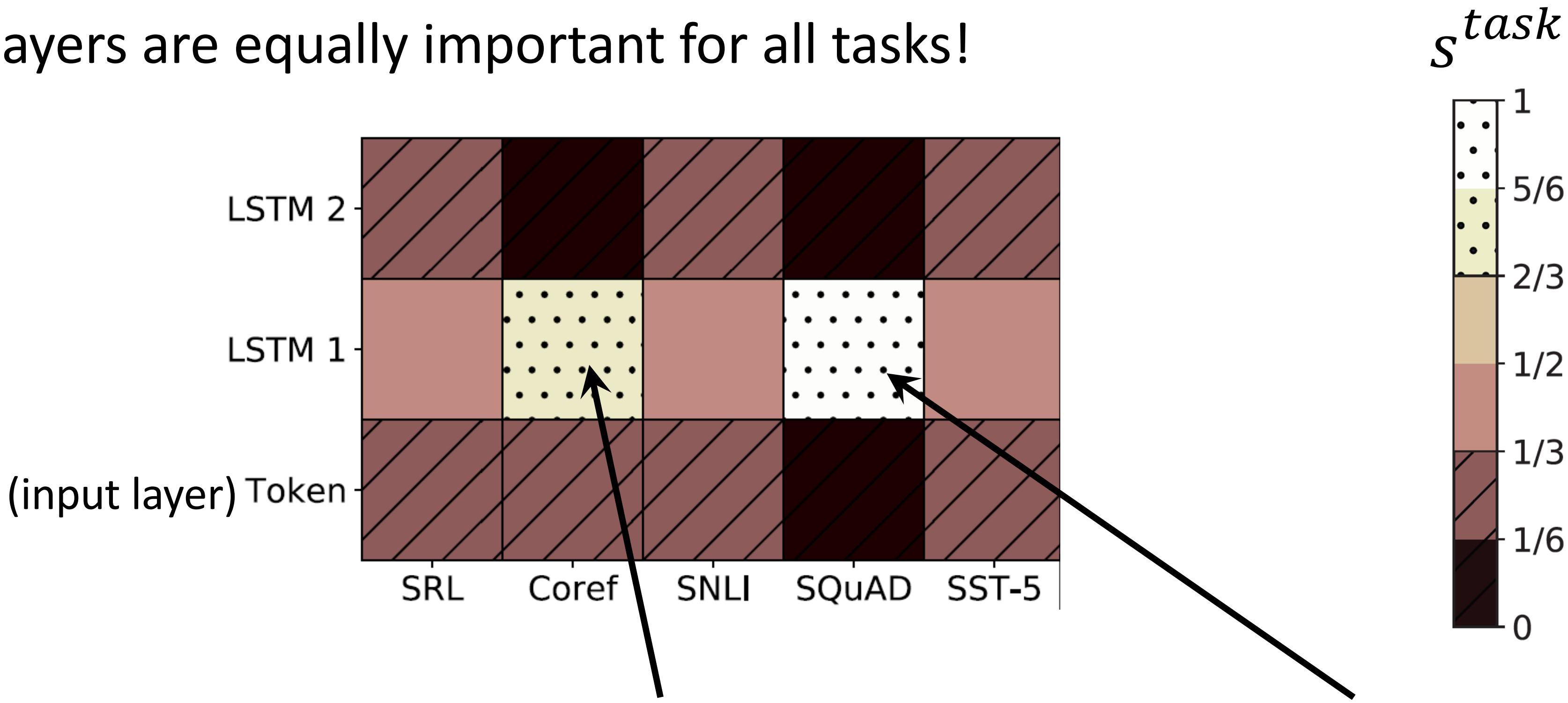
WSD

Model	**Acc.**
Collobert et al. (2011)	97.3
Ma and Hovy (2016)	97.6
Ling et al. (2015)	**97.8**
CoVe, First Layer	*93.3*
CoVe, Second Layer	92.8
biLM, First Layer	*97.3*
biLM, Second Layer	96.8

POS tagging

Deep Contextualized Word Representations

Not all layers are equally important for all tasks!



The 1st layer clearly dominates for co-reference resolution and question answering
For the other tasks, the weights are more evenly distributed among layers

OUTLINE

- RNNs, LSTMs
- **Attention/Context based architectures**
  - ELMO, **BERT,** BART
- Energy Based architectures

ATTENTION BASED ARCHITECTURES

Attention is ubiquitous in DL today

Image captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

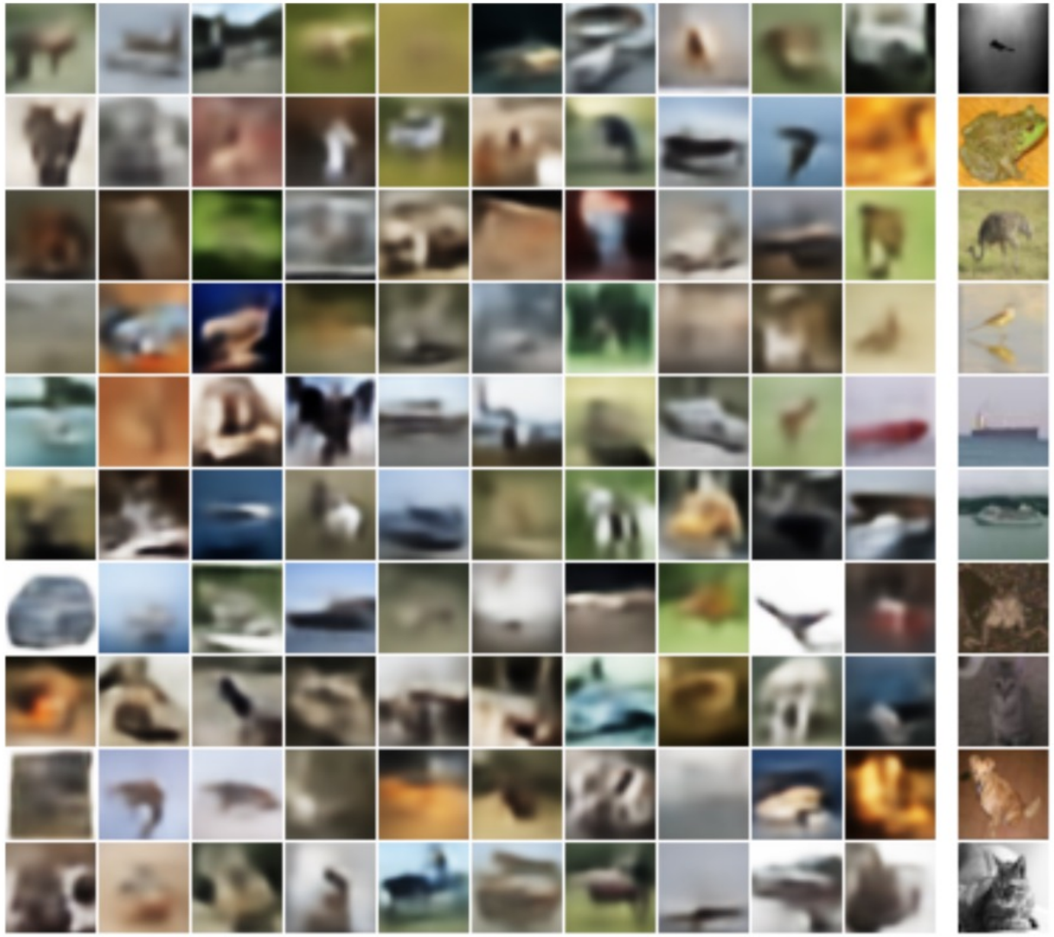
A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Show, attend and tell: Neural image caption generation with visual attention (Xu et al. 2015)

Attention is ubiquitous in DL today

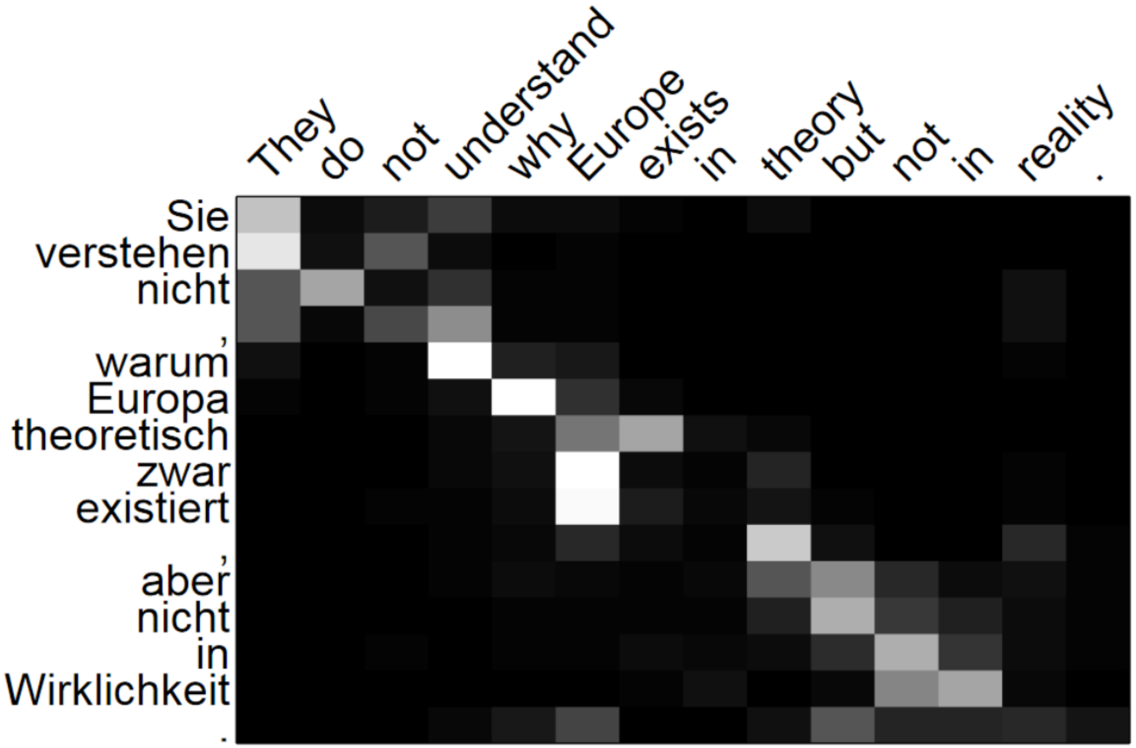
Image generation



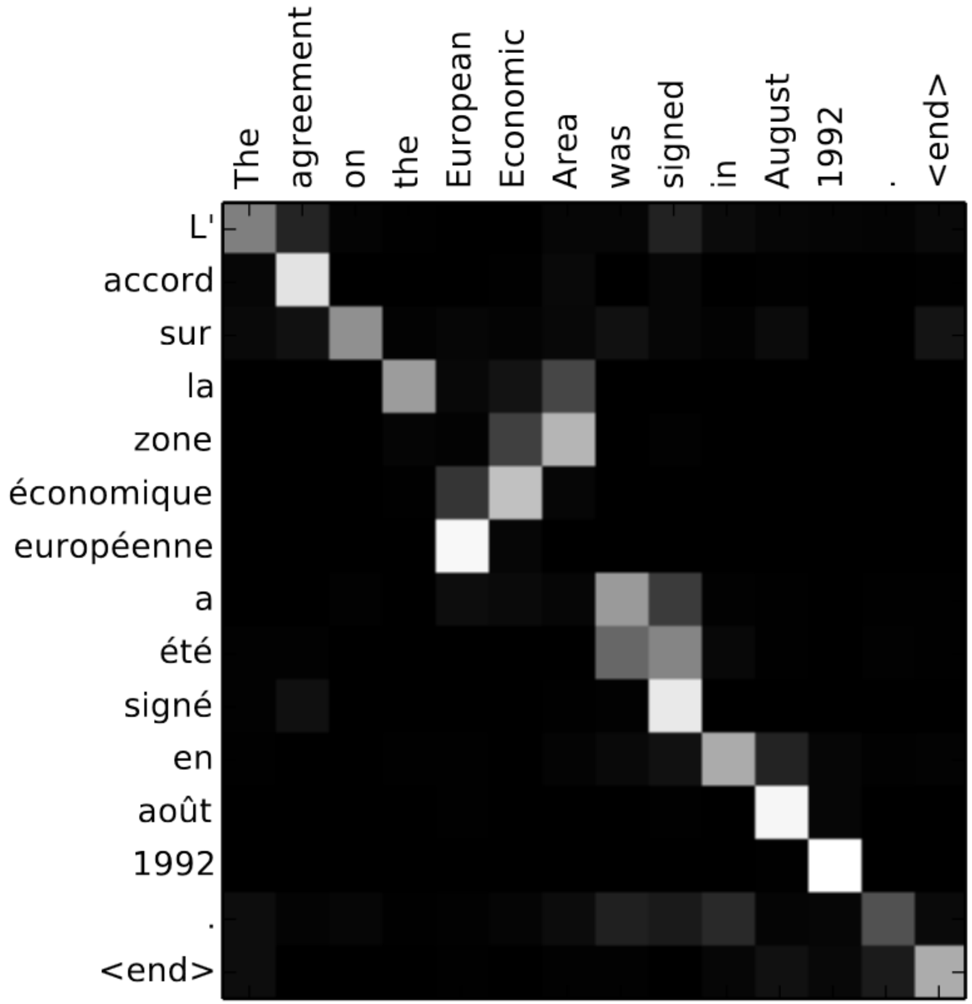
DRAW: A recurrent neural network for image generation (Gregor et al. 2015)

Attention is ubiquitous in DL today

Neural Machine Translation



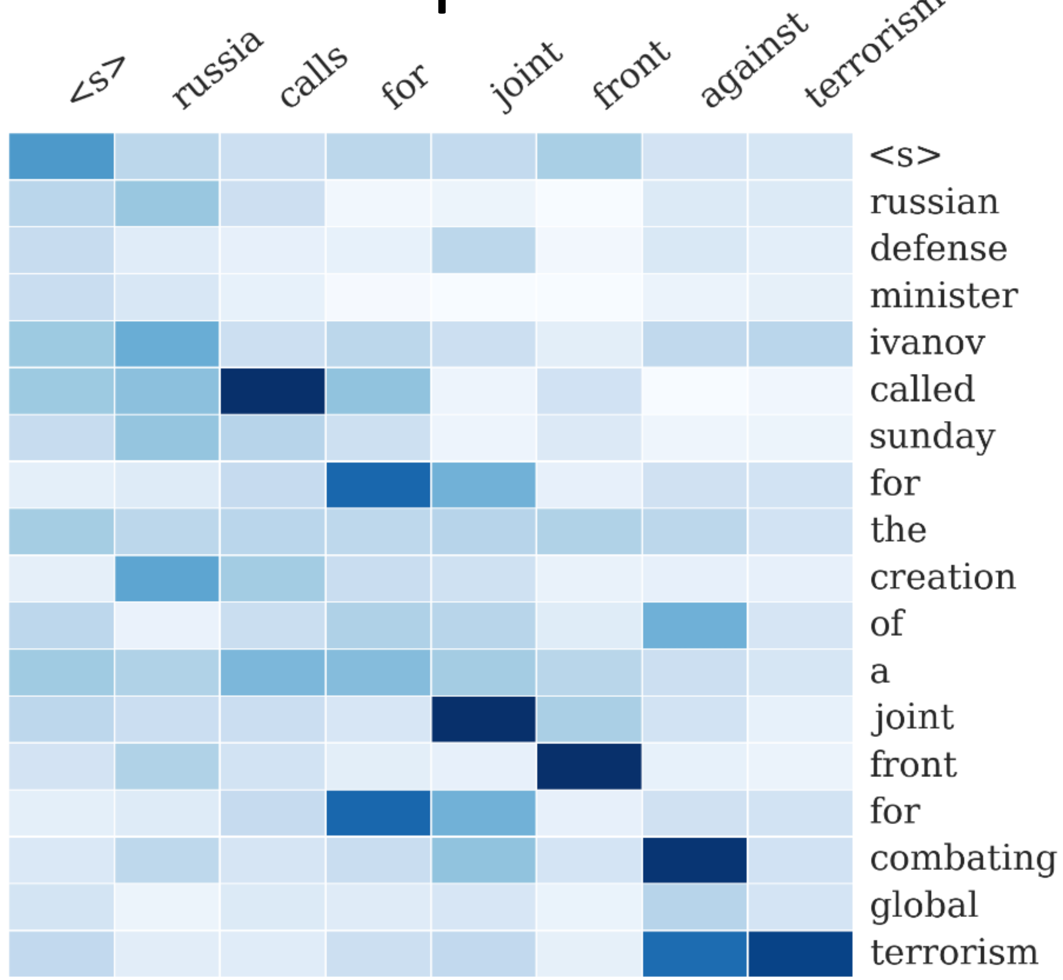
Effective approaches to attention-based neural machine translation (Luong et al. 2015)



Neural machine translation by jointly learning to align and translate (Bahdanau et al. 2014)

Attention is ubiquitous in DL today

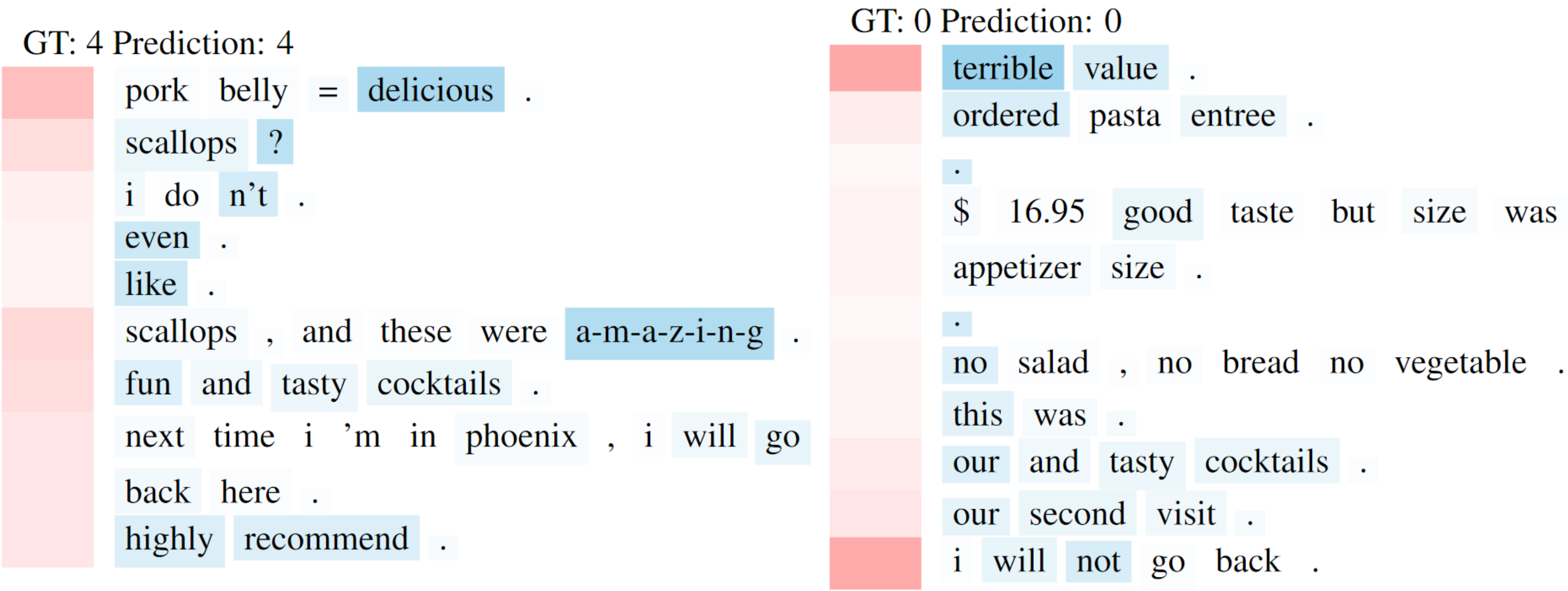
<u>Abstractive summarization</u>



A neural attention model for abstractive sentence summarization (Rush et al. 2015)

Attention is ubiquitous in DL today

Sentiment analysis



Hierarchical attention networks for document classification (Yang et al. 2016)

Encoder-Decoder Architectures

<u>General idea:</u>

encoder **decoder**

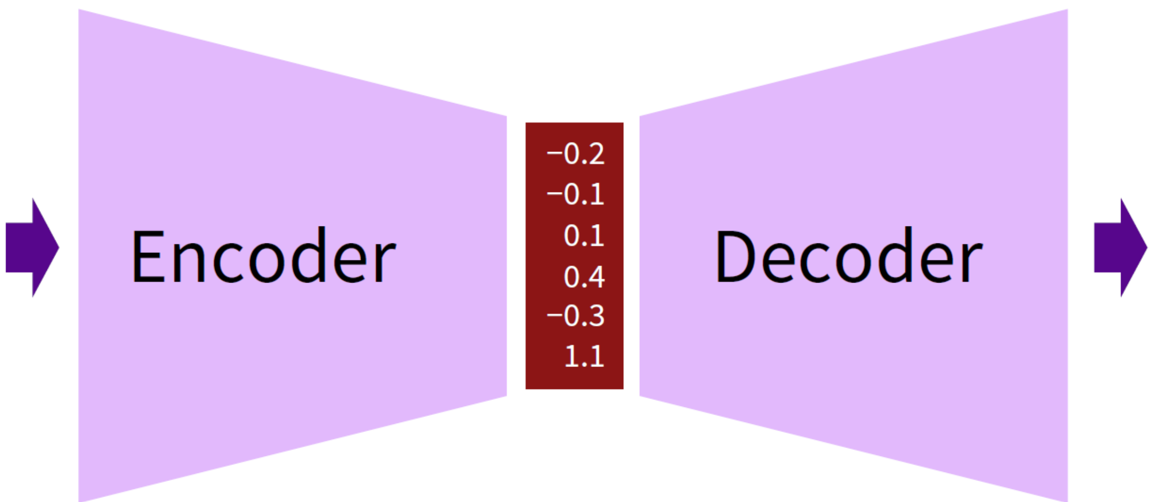
Input -> embedding -> Output

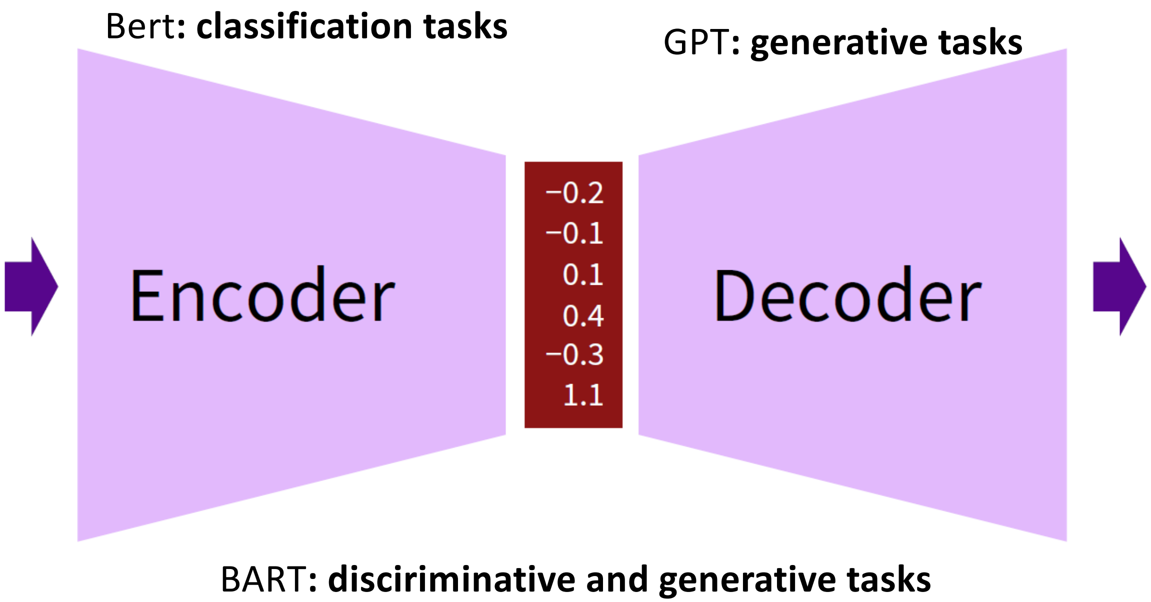
- image
- document
- source language
- ...

- caption
- summary
- target language
- ...

Known as *sequence-to-sequence* (seq2seq) when input and output are sequences (e.g., NLP applications)

The full architecture is differentiable => end-to-end training




Bert: **classification tasks**
GPT: **generative tasks**
Encoder
−0.2
−0.1
0.1
0.4
−0.3
1.1
Decoder
BART: **disciriminative and generative tasks**

What is attention?

Objective

- Traditional models (e.g., encoder) having to embed the input into a single fixed-length vector (lossy).
- Alternatively information can be kept and stored into multiple vectors.
- information can be retrieved later on (e.g., by the decoder). (Bahdanau et al. 2014)

Quick history:

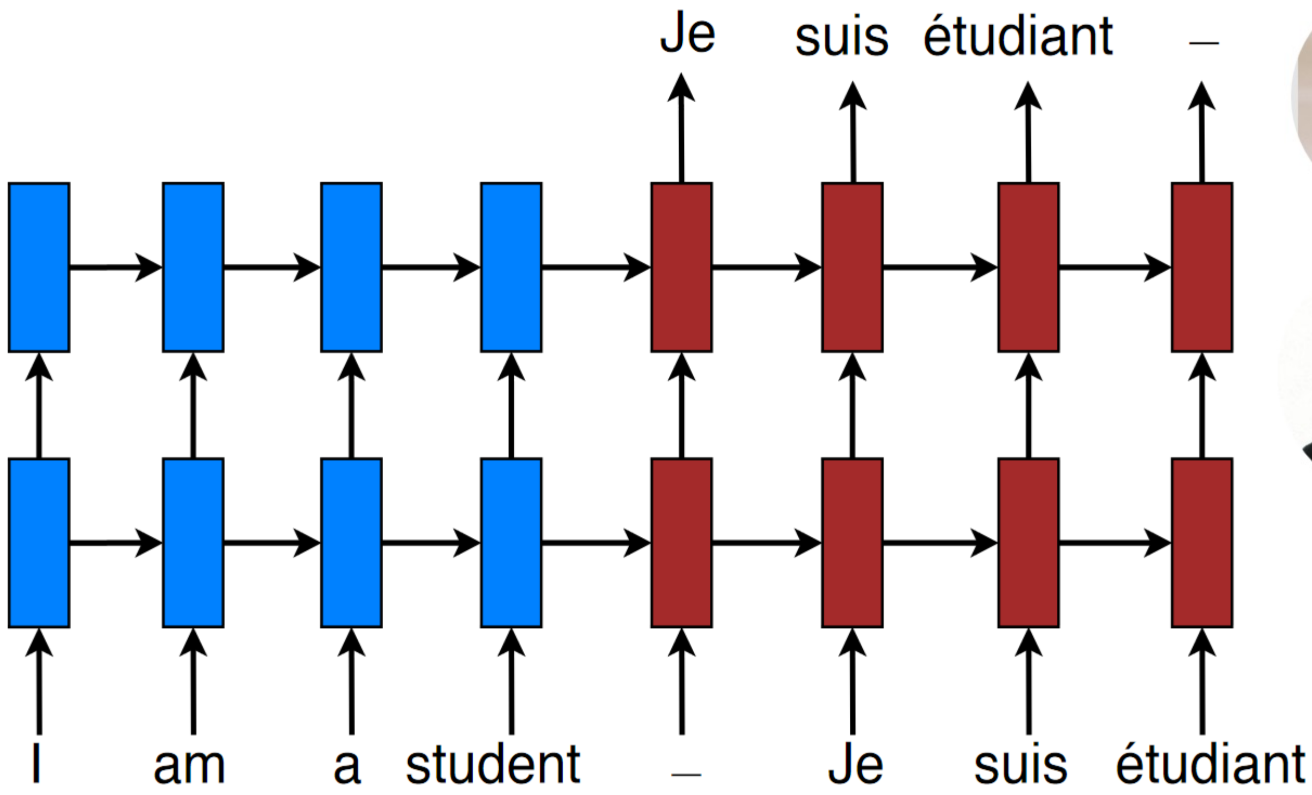
- developed in the context of **encoder-decoder** architectures for neural machine translation (Bahdanau et al. 2014)
- rapidly applied to naturally related tasks like image captioning (Xu et al. 2015) and summarization (*Luong et al. 2015*)
- also proposed for encoders only, e.g. for text classification (Yang et al. 2015) and representation learning (Conneau et al. 2017).

  Known as <u>*self or inner attention*</u> in such cases.

Encoder-Decoder for Neural Machine Translation

Encoder **Decoder**

Target *sentence (generated)*



Source *sentence (input)*

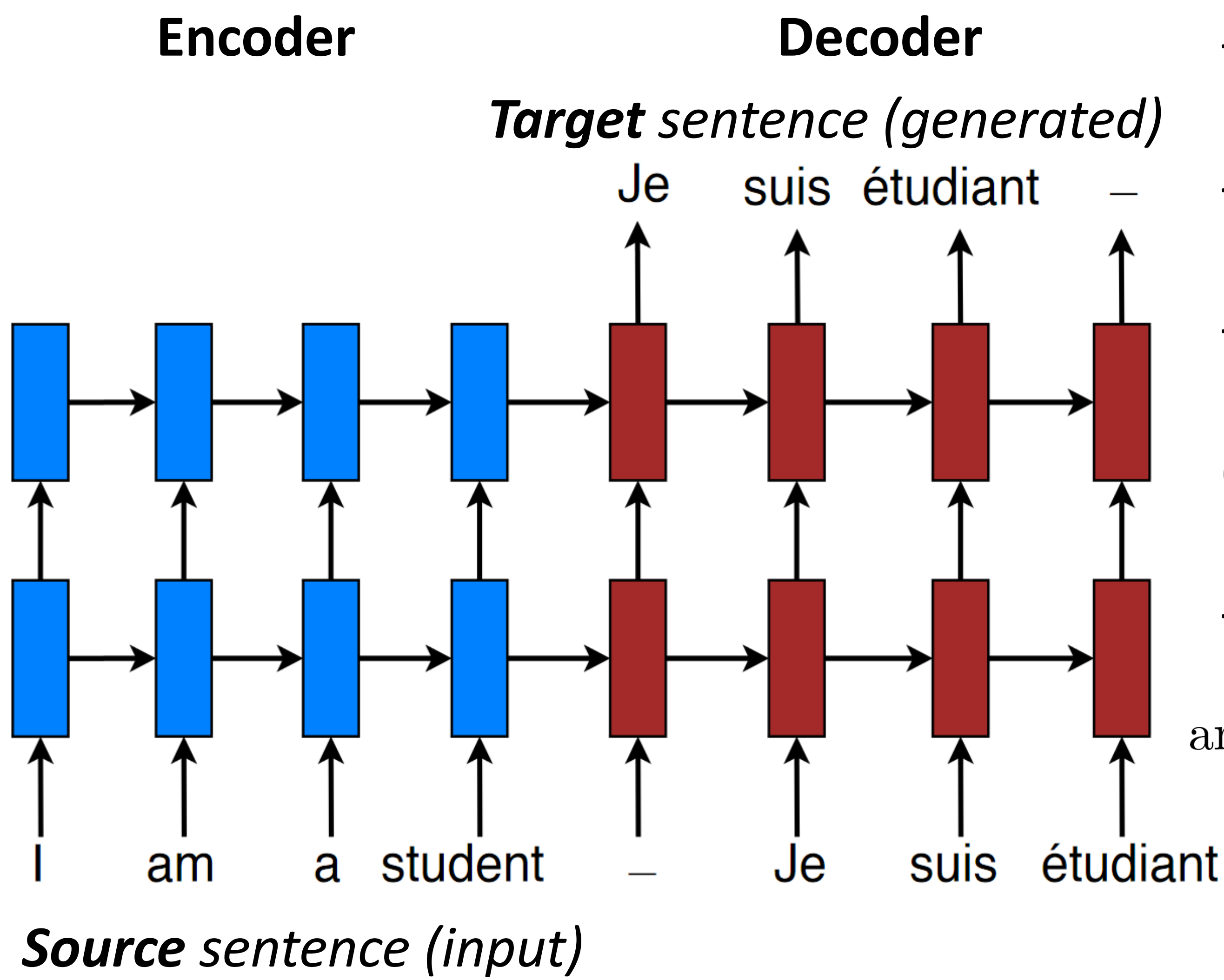
Effective approaches to Attention-Based Neural Machine Translation (2015)

Minh-Thang Luong
Research Scientist at Google
Verified email at google.com - Homepage
Deep Learning Natural Language Processing

Hieu Pham
Carnegie Mellon University, Google Brain
Verified email at google.com
Machine Learning

Christopher D Manning
Professor of Computer Science and Linguistics
Verified email at stanford.edu - Homepage
Natural Language Processing

Encoder-Decoder for Neural Machine Translation



- source: $\left(x_1, \ldots, x_{T_x}\right)$

- target: $\left(y_1, \ldots, y_{T_y}\right)$

- Both encoder & decoder are unidirectional deep RNNs (a.k.a. *stacking* RNNs)

- Training objective:

$$\operatorname{argmax}_\theta \left\{ \sum_{(x,y) \in \text{corpus}} \log p(y|x;\theta) \right\}$$

Luong et al. (2015)

Encoder-Decoder for Neural Machine Translation

Encoder: usually: CNN, stacking RNN* with LSTM or GRU units…
* unidirectional (Luong et al. 2015) or
bidirectional (Bahdanau et al. 2014).

Decoder: unidirectional *RNN* (well suited to text generation), best if deep.

Generates the target sentence $(y_1, \cdots, y_{T_y})$ one word at a time:

$$P\left[y_t \middle| \{y_1, ..., y_{t-1}\}, c_t\right] = \mathrm{softmax}\left(W_s \tilde{h}_t\right)$$

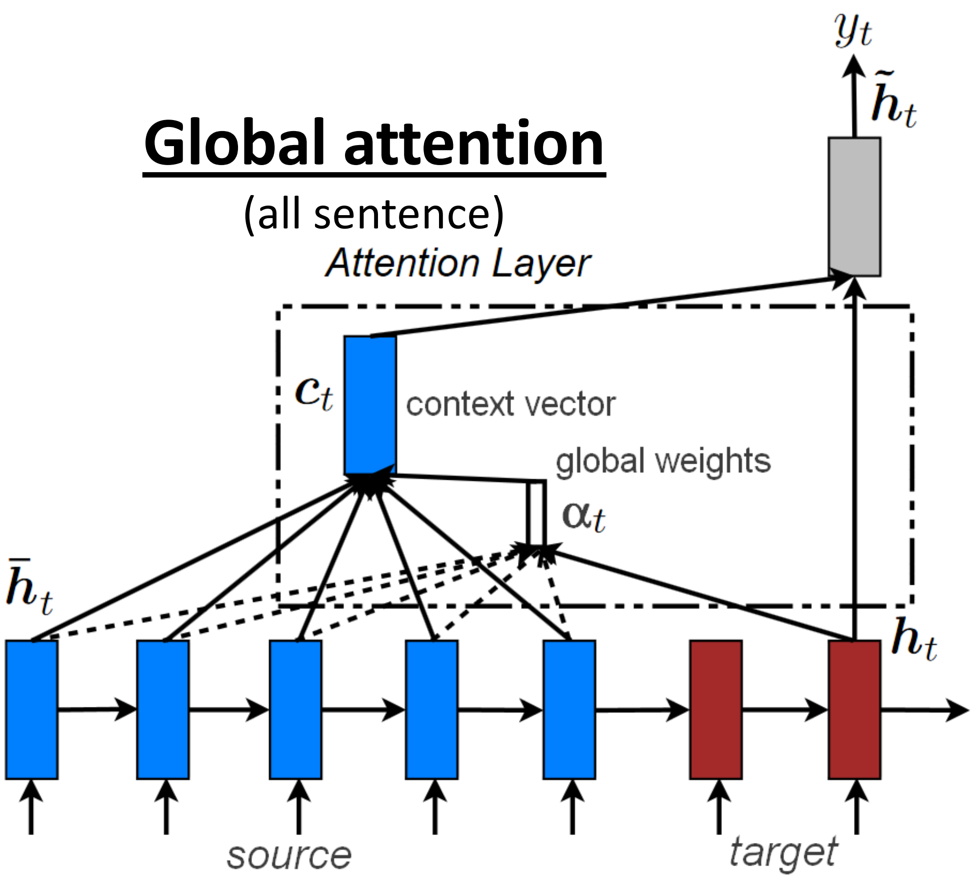
Luong et al. (2015)

Encoder-**Decoder** for Neural Machine Translation

$$P\left[y_t \middle| \{y_1, \ldots, y_{t-1}\}, c_t\right] = \mathrm{softmax}\left(W_s \tilde{h}_t\right)$$

Global attention

(all sentence)



attentional hidden state

$$\tilde{h}_t = \tanh\left(W_c \left[c_t; h_t\right]\right)$$

context vector

decoder hidden state

$$c_t = \sum_{i=1}^{T_x} \alpha_{t,i} \bar{h}_i$$

i^th encoder hidden state

alignment vector

$$\alpha_{t,i} = \frac{\exp\left(\mathrm{score}(h_t, \bar{h}_i)\right)}{\sum_{i'=1}^{T_x} \exp\left(\mathrm{score}(h_t, \bar{h}_{i'})\right)}$$

$$\mathrm{score}(h_t, \bar{h}_i) = h_t^\top \bar{h}_i$$

Luong et al. (2015)

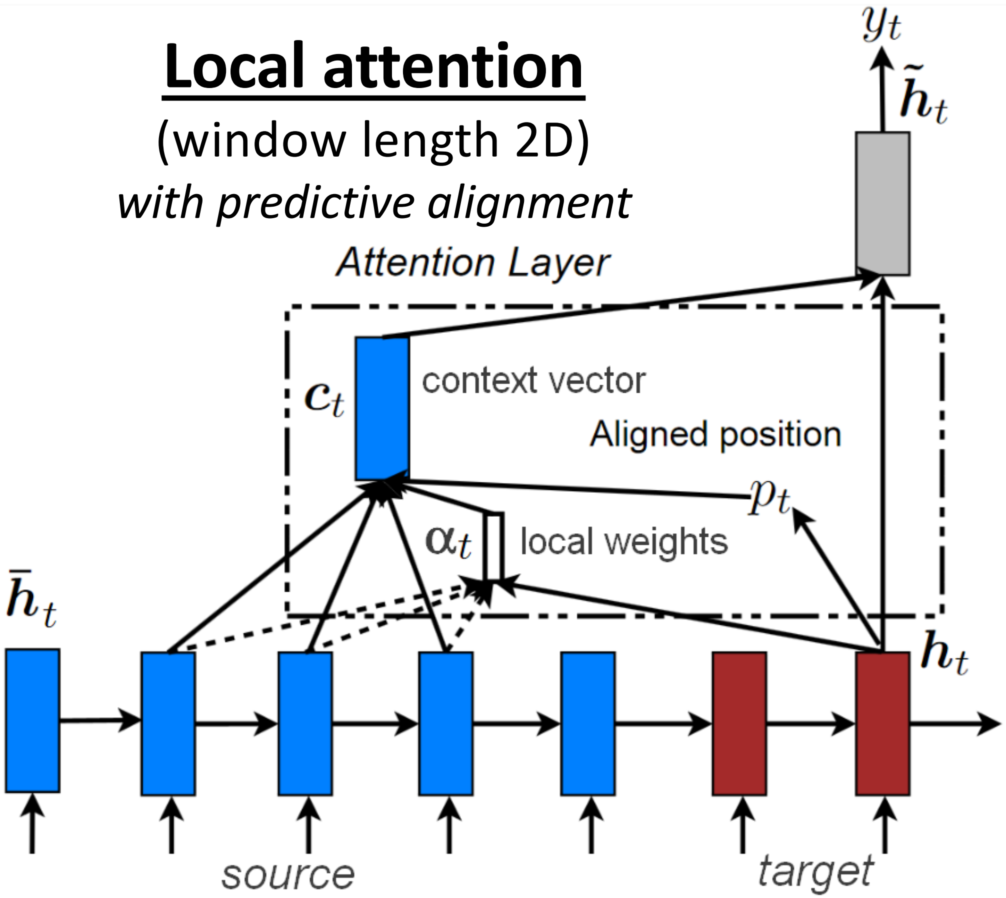
Encoder-**Decoder** for Neural Machine Translation

$$P\left[y_t \middle| \{y_1, \ldots, y_{t-1}\}, c_t\right] = \operatorname{softmax}\left(W_s \tilde{h}_t\right)$$

Local attention

(window length 2D)

with predictive alignment



attentional hidden state

$$\tilde{h}_t = \tanh\left(W_c\left[c_t; h_t\right]\right)$$

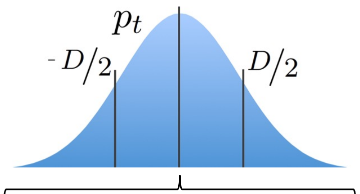
decoder hidden state

context vector

$$p_t = T_x \cdot \sigma\left(v_p^\top \tanh(W_p h_t)\right)$$

i^{th} encoder hidden state

$$c_t = \sum_{i=p_t-D}^{p_t+D} \alpha_{t,i} \bar{h}_i$$



alignment vector

$$\alpha_{t,i} = \frac{\exp\left(\operatorname{score}(h_t, \bar{h}_i)\right)}{\sum_{i'=p_t-D}^{p_t+D} \exp\left(\operatorname{score}(h_t, \bar{h}_{i'})\right)} \exp\left(-\frac{(i-p_t)^2}{2(D/2)^2}\right)$$

$$\operatorname{score}(h_t, \bar{h}_i) = h_t^\top W_\alpha \bar{h}_i$$

Luong et al. (2015)

Encoder-Decoder for Neural Machine Translation

Facts:

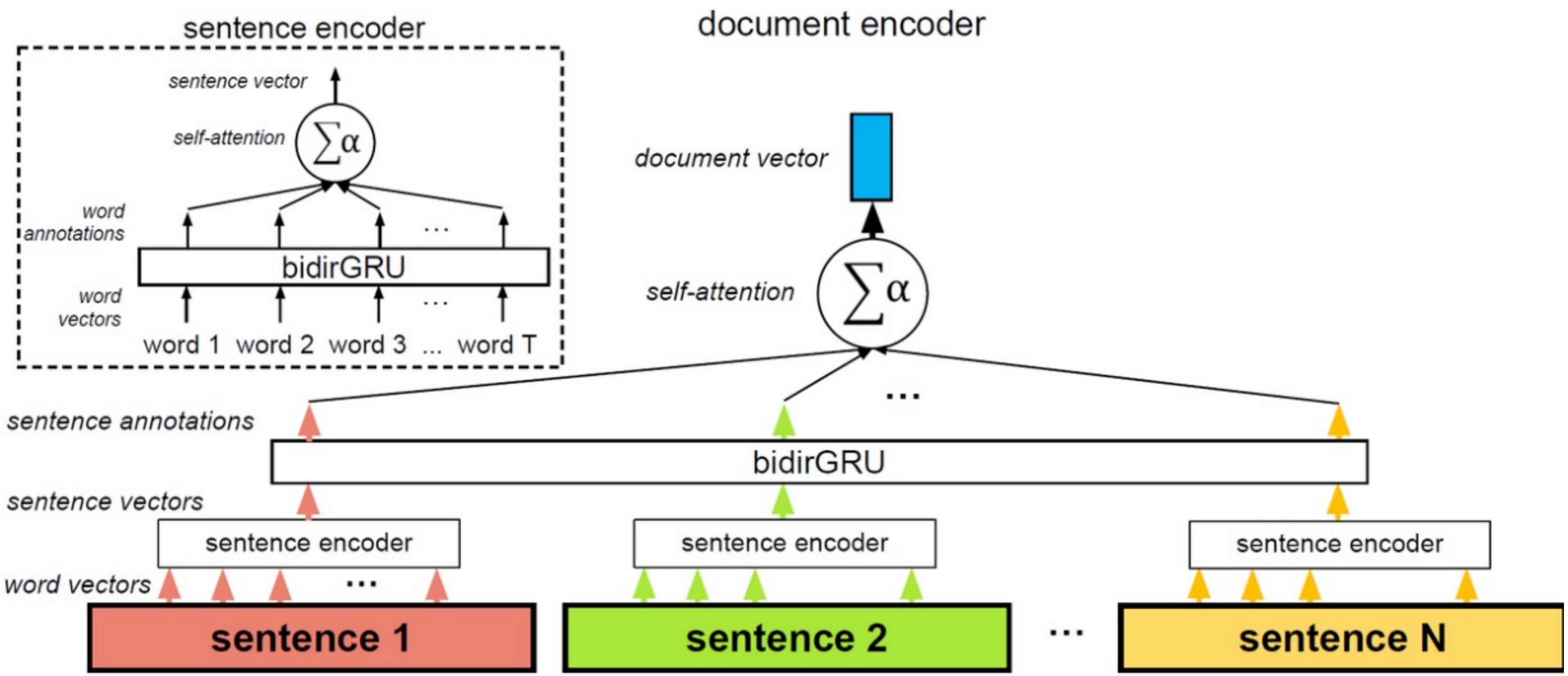
- Tested on the English <-> German task (WMT'14 dataset)
- 4.5M sentence pairs
- Encoder and decoder RNNs feature 4 layers of stacking and 1000-dimensional hidden states
- Window of size *D=10* for local attention
- Trained for 12 epochs (total of 7-10 days on a single GPU, at 1K target words/s)
- New state-of-the-art performance

Lessons learned:

- Local attention with predictive alignment gives better results than global attention
- Dot product ($\mathrm{score}(h_t, \bar{h}_i) = h_t^\top \bar{h}_i$) works well for global attention
- The general formulation ($\mathrm{score}(h_t, \bar{h}_i) = h_t^\top W_\alpha \bar{h}_i$) is better for local attention

Luong et al. (2015)

Self-attention for RNN encoders



Self-attention for RNN encoders

- Input is a sentence $(x_1, \ldots, x_T)$
- We're only interested in getting an embedding s of the sentence for some downstream task (e.g., classification)

encoder hidden state

context vector

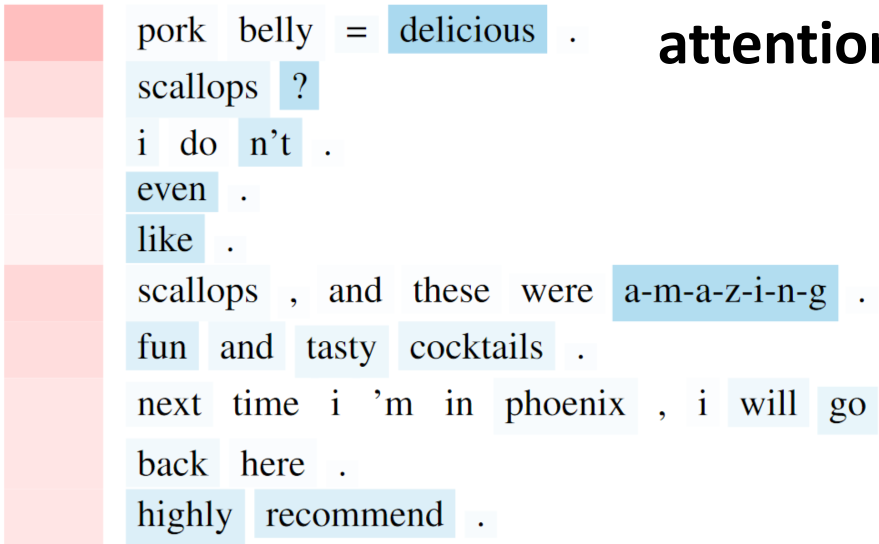
$$u_t = \tanh(W h_t)$$

$$\alpha_t = \frac{\exp(\text{score}(u_t, u))}{\sum_{t'=1}^{T} \exp(\text{score}(u_{t'}, u))}$$

$$s = \sum_{t=1}^{T} \alpha_t h_t$$

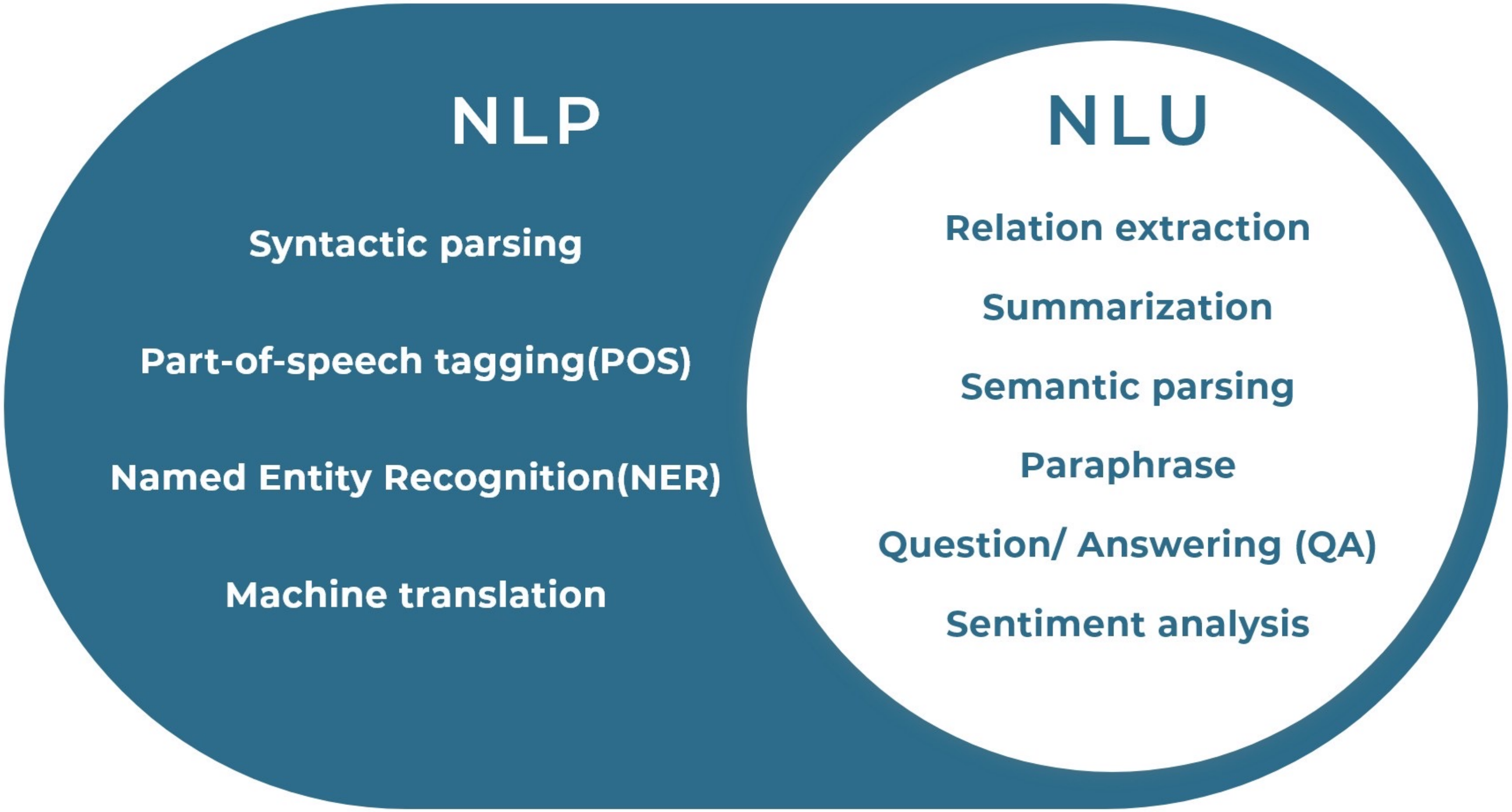
Where $\text{score}(u_t, u) = u_t^\top u$

The same process can be repeated over the sentence vectors s -> **hierarchical attention**



OUTLINE

- Representation Learning for Text
  - SVD
  - Word2Vec
  - Fast Text Embeddings – subword information
  - Document Representations
- CNNs for text classification
- Attention Based architectures
- **NLP tasks and evaluation**
  - GLUE, FLUE
  - French Linguistics


NLP
Syntactic parsing
Part-of-speech tagging(POS)
Named Entity Recognition(NER)
Machine translation
NLU
Relation extraction
Summarization
Semantic parsing
Paraphrase
Question/ Answering (QA)
Sentiment analysis

Motivation

Nowadays, it's critical to develop NLU models with understanding beyond the detection of superficial correspondences between inputs and outputs, and to facilitate this development we need an evaluation test or a **benchmark** to evaluate language models.

GLUE [1] **Flaubert** [2] papers…

[1] https://arxiv.org/abs/1804.07461 [2] https://arxiv.org/abs/1912.05372

G.L.U.E

General Language Understanding Evaluation

- Collection of NLU tasks (Q/A, sentiment analysis, etc…)
- An online platform for model evaluation
- GLUE only considers data sets in English language
- GLUE only considers the ability to make predictions
- Based on 9 data sets that cover different sizes, text genres and difficulties
- In addition it has diagnostic evaluation data set

GLUE: tasks and datasets

Dataset	Description	Data example		Metric
CoLA	Is the sentence grammatical or ungrammatical?	"This building is than that one." = **Ungrammatical**	**acceptability (2 classes)**	Matthews
SST-2	Is the movie review positive, negative, or neutral?	"The movie is funny , smart , visually inventive , and most of all , alive ." = **.93056 (Very Positive)**	**only positive/negative sentiments**	Accuracy
MRPC	Is the sentence B a paraphrase of sentence A?	A) "Yesterday , Taiwan reported 35 new infections , bringing the total number of cases to 418 ." B) "The island reported another 35 probable cases yesterday , taking its total to 418 ." = **A Paraphrase**	**Similarity (2 classes)**	Accuracy / F1
STS-B	How similar are sentences A and B?	A) "Elephants are walking down a trail." B) "A herd of elephants are walking along a trail." = **4.6 (Very Similar)**	**Similarity (regression)**	Pearson / Spearman
QQP	Are the two questions similar?	A) "How can I increase the speed of my internet connection while using a VPN?" B) "How can Internet speed be increased by hacking through DNS?" = **Not Similar**	**Q/A task (2 classes)**	Accuracy / F1
MNLI-mm	Does sentence A entail or contradict sentence B?	A) "Tourist Information offices can be very helpful." B) "Tourist Information offices are never of any help." = **Contradiction**	**Entailment task (3 classes)**	Accuracy
QNLI	Does sentence B contain the answer to the question in sentence A?	A) "What is essential for the mating of the elements that create radio waves?" B) "Antennas are required by any radio receiver or transmitter to couple its electrical connection to the electromagnetic field." = **Answerable**	**Q/A task (3 classes)**	Accuracy
RTE	Does sentence A entail sentence B?	A) "In 2003, Yunus brought the microcredit revolution to the streets of Bangladesh to support more than 50,000 beggars, whom the Grameen Bank respectfully calls Struggling Members." B) "Yunus supported more than 50,000 Struggling Members." = **Entailed**	**Entailement task (2 classes)**	Accuracy
WNLI	Sentence B replaces sentence A's ambiguous pronoun with one of the nouns - is this the correct noun?	A) "Lily spoke to Donna, breaking her concentration." B) "Lily spoke to Donna, breaking Lily's concentration." = **Incorrect Referent**	**Inference task (2 classes)**	Accuracy
Diagnostic	This dataset do not envolve in evaluation but only to address certain phenomena in the language such as	A) I have never seen a hummingbird not flying. B) I have never seen a hummingbird. A-->B : **no entailement.** B-->A : **entailement.**	Inference task (3 classes)	R3 score

	Rank	Name	Model	Score
+	1	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS	90.6
	2	ERNIE Team - Baidu	ERNIE	90.4
+	3	Alibaba DAMO NLP	StructBERT	90.3
	4	T5 Team - Google	T5	90.3
	5	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART	89.9
+	6	ELECTRA Team	ELECTRA-Large + Standard Tricks	89.4
+	7	Huawei Noah's Ark Lab	NEZHA-Large	88.7
+	8	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)	88.4
	9	Junjie Yang	HIRE-RoBERTa	88.3
	10	Facebook AI	RoBERTa	88.1

F.L.U.E

French Language Understanding Evaluation

- Represents the same idea as GLUE but for the french language
- Contains 6 different datasets with different sizes, tasks and difficulties
- 3 from these datasets are from cross-lingual datasets
- The idea of FLUE came with Flaubert model to compare different french language models such as CamemBert

FLUE: tasks and datasets

Dataset	Domain	Task
CLS	Books/DVD/Music product reviews	Sentiment Analysis (2 classes)
PAWS-X	General domain	Paraphrase/Similarity (2 classes)
XNLI	Diverse Genres	Inference, NLI (3 classes)
French TreeBank	Daily newspaper	POS tag
French SemEval	Diverse Genres	Verb Sense
Noun Sense Disambiguation	Diverse Genres	Noun Sense

Model	Books	DVD	Music
MultiFiT†	91.25	89.55	93.40
mBERT†	86.15	86.90	86.65
CamemBERT	92.30	93.00	94.85
FlauBERT$_{\text{BASE}}$	93.10	92.45	94.10
FlauBERT$_{\text{LARGE}}$	**95.00**	**94.10**	**95.85**

† Results reported in (Eisenschlos et al., 2019).

OUTLINE

- Representation Learning for Text
  - SVD
  - Word2Vec
  - Fast Text Embeddings – subword information
- **NLP tasks and evaluation**
  - GLUE, FLUE
  - **French Linguistics (DaSciM)**

Large Scale French Linguistics Resources (DaSciM)

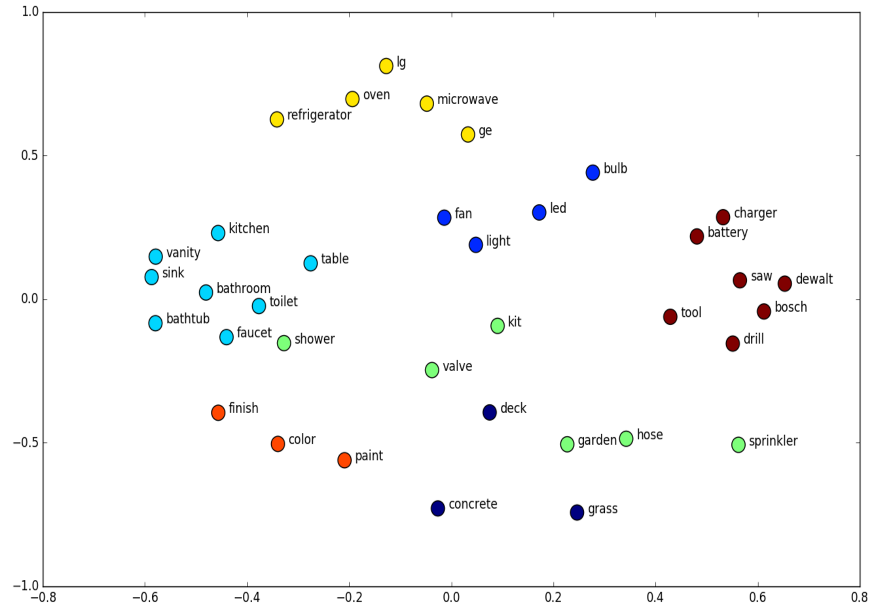
Data Collection

- Crawling more than 1M web pages using Heritrix for **45 days** to get **500GB** of text.
- Using FastText Language detection tool to extract **330GB** of French text.
- Applying deduplication to eliminate redundant data from the corpus which gives us **33GB** of deduplicated French raw text.
- Using Stanford NLP French tokenizer to tokenize the deduplicated text.

Large Scale French Linguistics Resources (DaSciM)

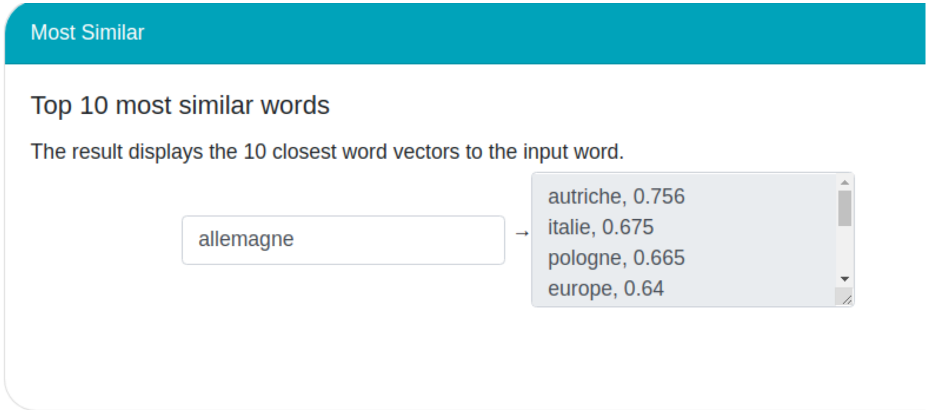
Word Embeddings

- Word embeddings are a class of algorithms where each word is represented as real-valued vector.
- The learning process of these vectors is either joint with a neural network model on some task or is an unsupervised process.
- Similar words in meaning have similar representation.



Training French Word Vectors

Training on:
- 33GB French raw text crawled from the French web
- multiple pre-processing: French language detection, deduplication and tokenization.

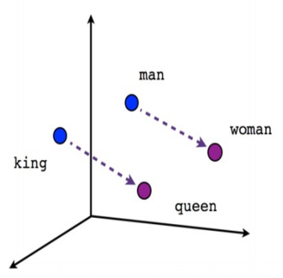


Embeddings	VOC	Tool	Method	Corpus	Window
Fr_web_w5	0.8M	word2vec	CBOW	Fr_web	5
Fr_web_w20	4.4M	word2vec	CBOW	Fr_web	20
Fr_fl_w5	1M	word2vec	CBOW	Flaubert_data	5
Fr_fl_w20	6M	word2vec	CBOW	Flaubert_data	20

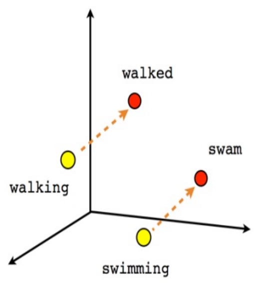
http://master2-bigdata.polytechnique.fr/

Word Analogy

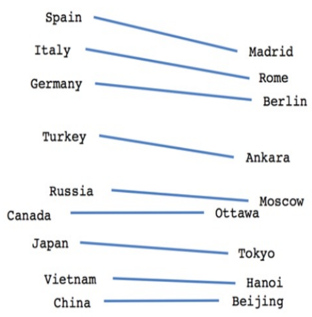
- Word embeddings evaluation method.
- Based on the assumption that a linear relation between word pairs indicates the quality of word embeddings.
- French word analogies dataset that contains **31 688** questions.



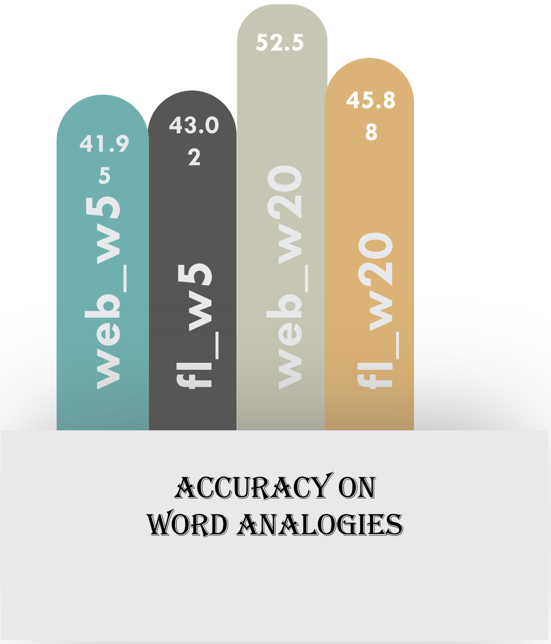
Male-Female



Verb tense

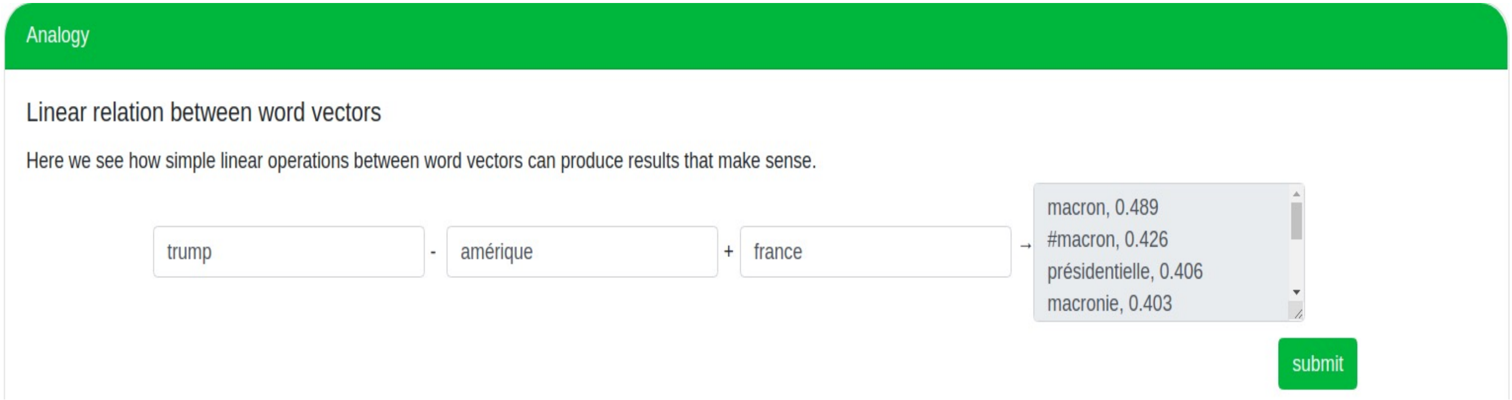


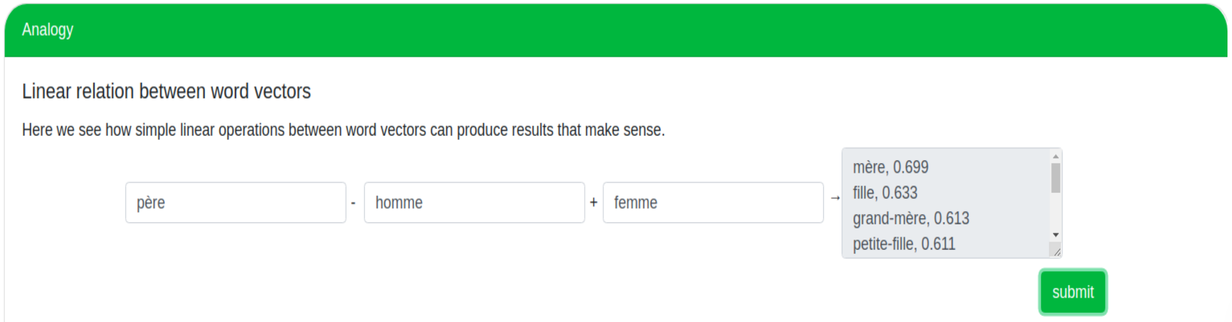
Country-Capital

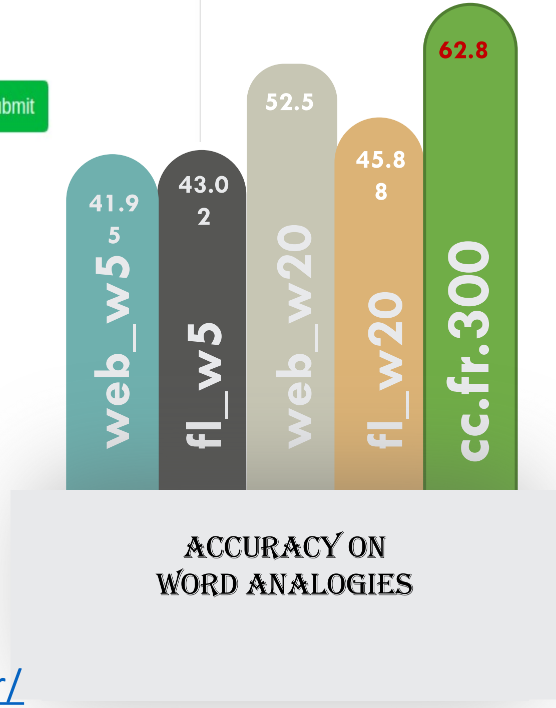


http://master2-bigdata.polytechnique.fr/

Word Analogy







http://master2-bigdata.polytechnique.fr/

French Language Understating Evaluation - FLUE

- A French language understanding evaluation benchmark.
- It contains many datasets that varies in subject, level of difficulty, size and degree of formality.

Dataset	Domain	Task
CLS	Product reviews	Sentiment analysis (Binary classification)
PAWS-X	General domain	Paraphrasing (Binary classification)
XNLI	Diverse genre	Natural language inference (3 classes)
NSD	Diverse domain	Noun sense

Cross Lingual Sentiment - CLS

- Amazon reviews with rating from 1 to 5 divided into three subsets: books, DVD◙ and music♫.

- Each subset contains a balanced train and test set that contains around 1000 positive and 1000 negative samples.

PAWS-X - Paraphrasing (Binary classification)

- Binary classification task that aims to identify if there is a semantic relation between a pair of sentences or not.
- Wikipedia

  high lexical overlap

  Quora

  Pair-sentence sample $\xrightarrow{\text{Translation+human judge}}$ PAWS-FR

- Dataset contains 49 401 training samples, 1 992 validation samples and 1 985 test samples.

XNLI - Natural language inference (3 classes)

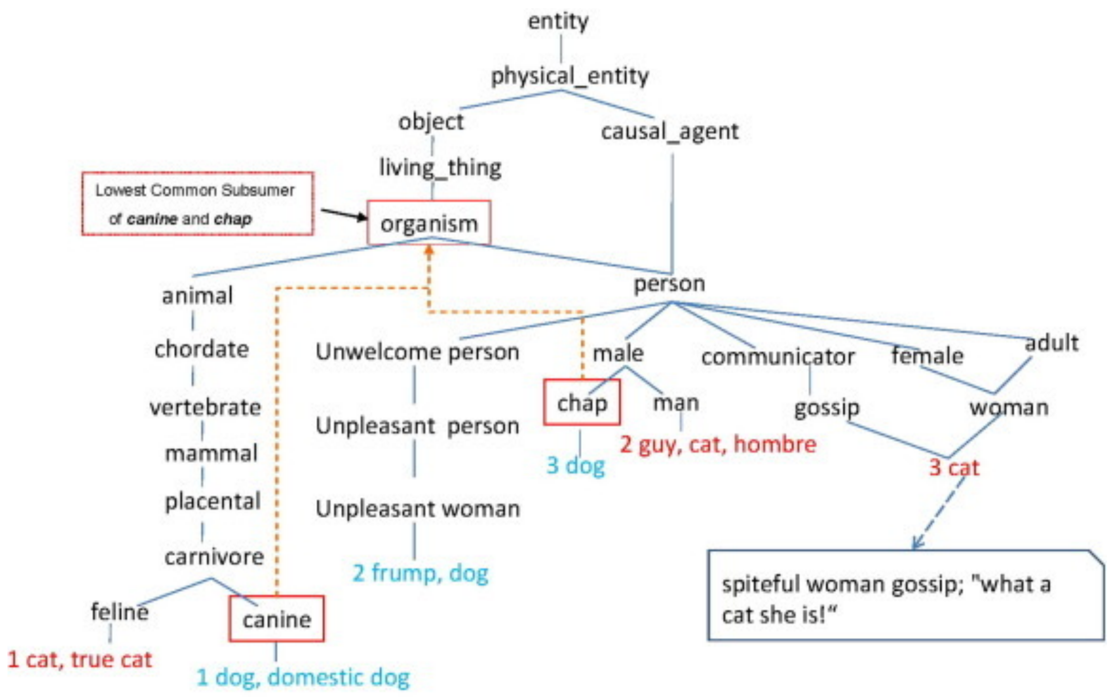
- Task of determining whether a “hypothesis” is true (entailment), false (contradiction), or undetermined (neutral) given a “premise”.
- **premise:** un vaste programme de rénovation devrait être achevé d’ici la fin de 2001.

  hypothesis: le programme de rénovation prendra fin avant le début de l’année 2001.

  label: contradiction.
- The dataset consists of **392 702** training samples, **2 491** validation samples and **5 010** test samples.

NSD - Noun sense

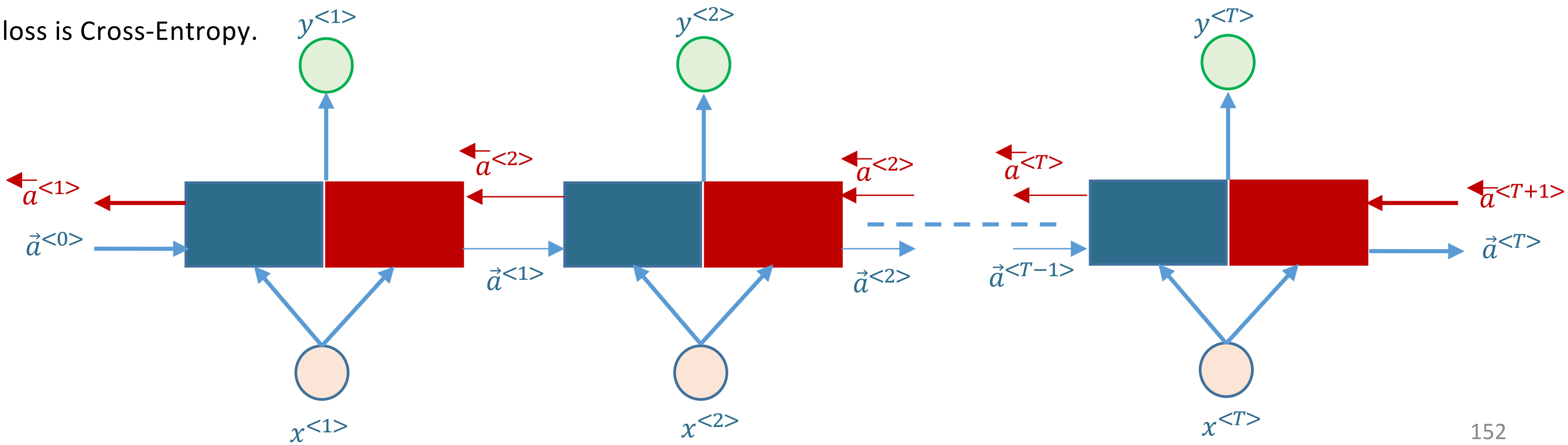
- The evaluation set that is composed of 306 sentences and 1 445 French annotated nouns translated from WordNet.
- The training set is resulted from translating **WordNet Gloss Corpus** to French.



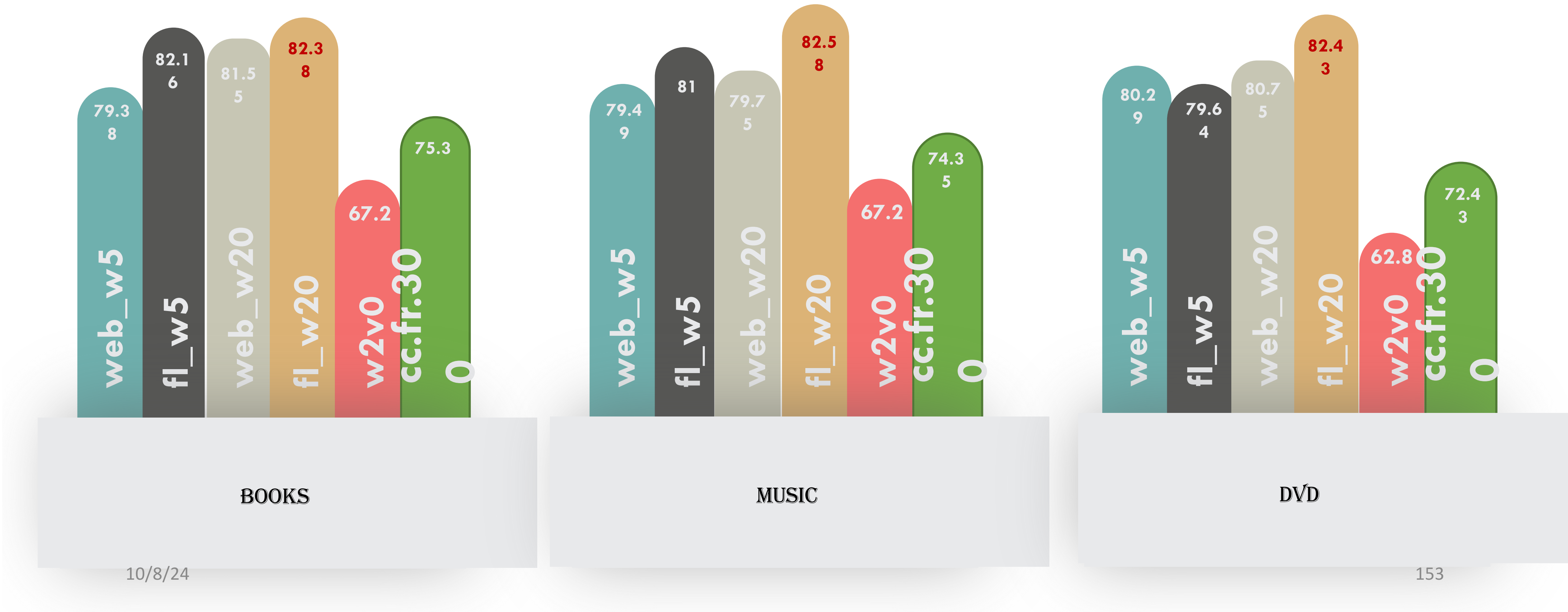
CLS using BiLSTM

BiLSTM/RNN for sentiment classification

- The output of both the previous and next steps are used as input to the current step.
- $\vec{a}^{<t>} = g_1(W_{\vec{a}\vec{a}}\,\vec{a}^{<t-1>} + W_{\vec{a}x}x^{<t>} + b_{\vec{a}}\,)$
- $\overleftarrow{a}^{<t>} = g_1(W_{\overleftarrow{a}\overleftarrow{a}}\,\overleftarrow{a}^{<t+1>} + W_{\overleftarrow{a}x}x^{<t>} + b_{\overleftarrow{a}}\,)$
- $y^{<t>} = g_2(W_{y\overleftarrow{a}}\,\overleftarrow{a}^{<t>} + W_{y\vec{a}}\vec{a}^{<t>} + b_y)$
- The used classification head is formed of: 0.1 dropout, 3000D projection layer, hyperbolic tangent activation, 0.1 dropout, 2D projection layer (classes number) and finally soft-max to find the probability of each class.
- The used loss is Cross-Entropy.

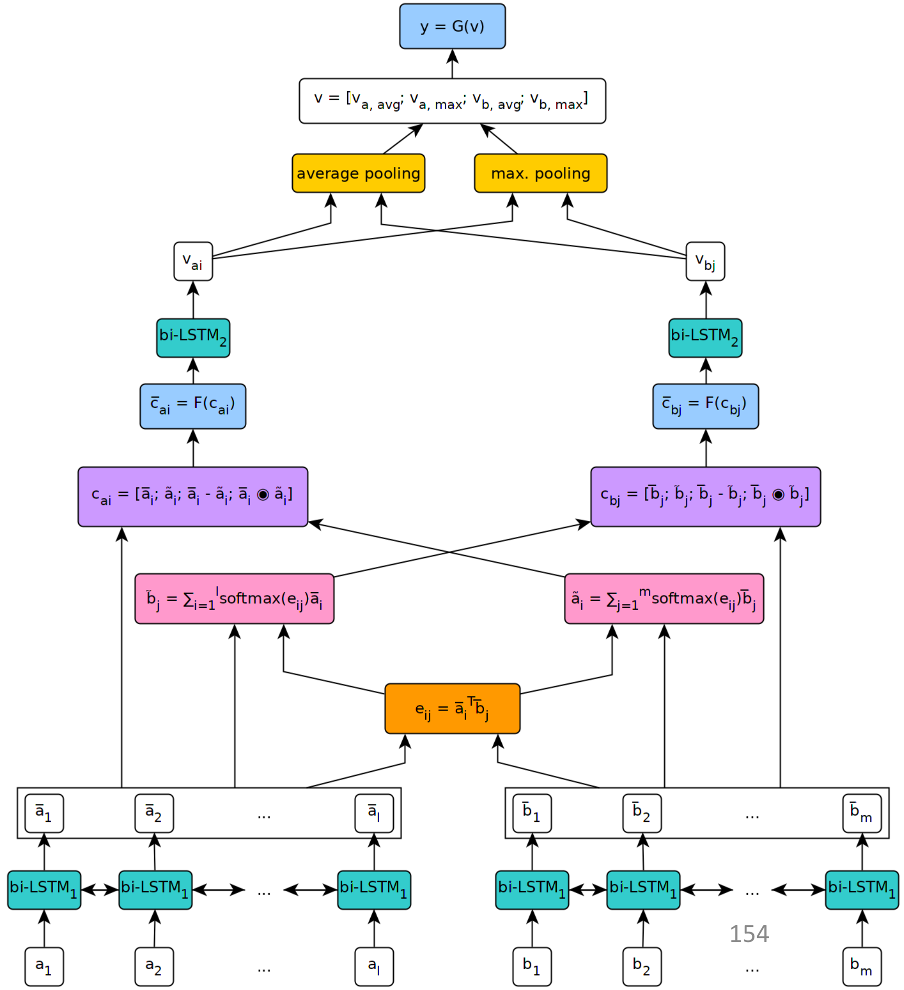


Accuracy On CLS using BiLSTM

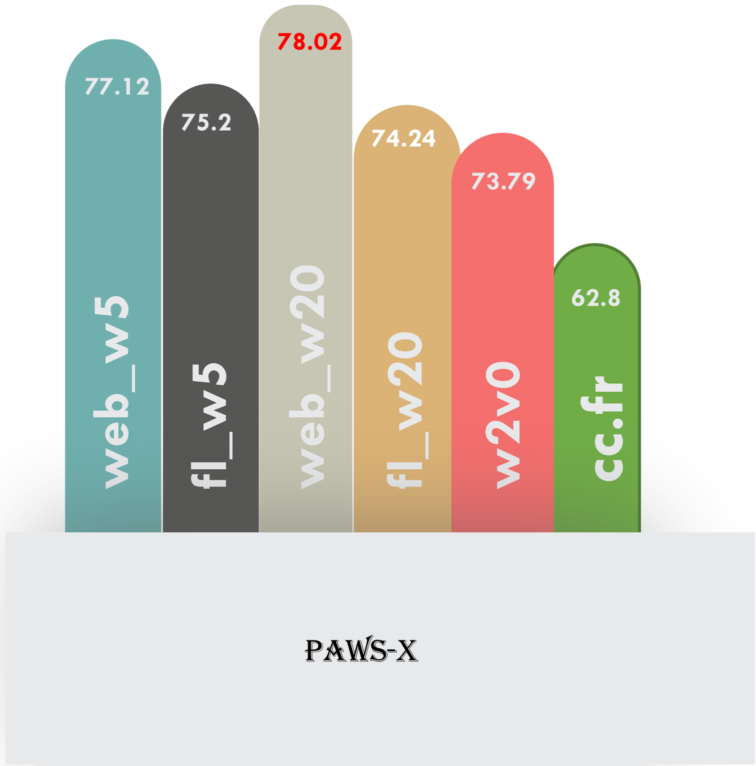


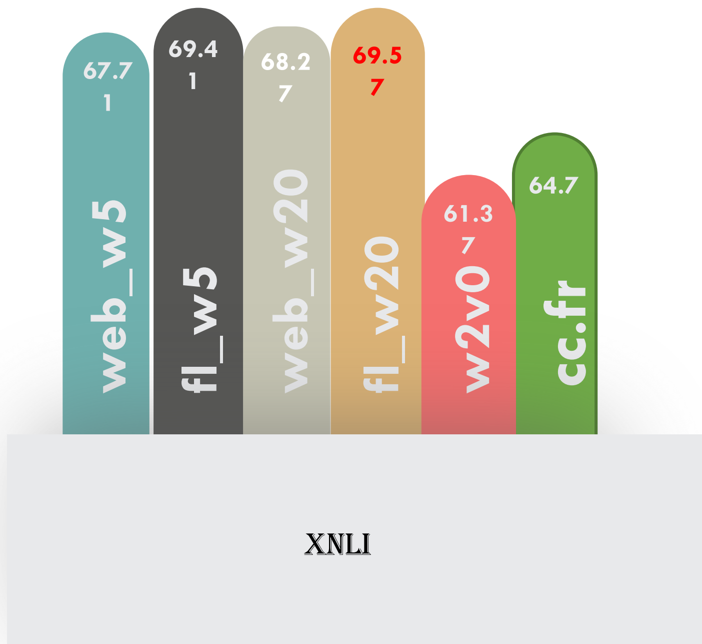
ESIM - Enhanced Sequential Inference Model

- Based on representing each word by its meaning in the first sentence using BiLSTM and by its relation with each word in the second sentence.
- The classification head is the same as sentiment prediction used in CLS dataset.
- The used loss to finetune the model is Cross-Entropy.



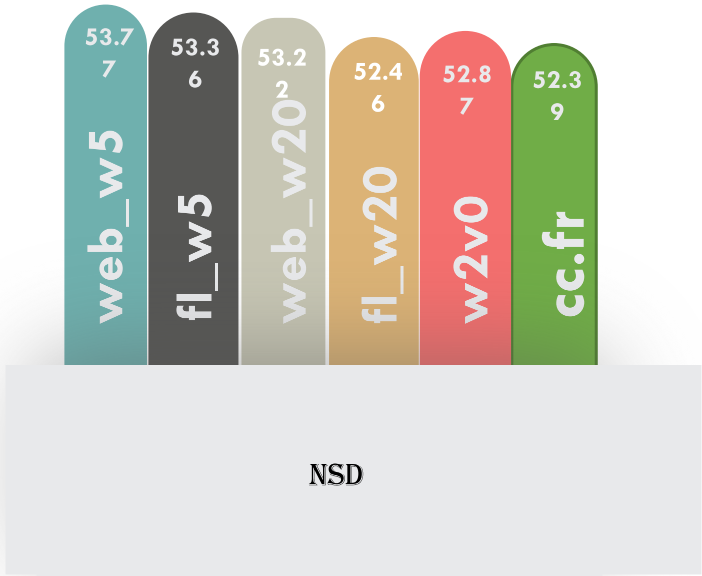
Accuracy On PAWS-X ανδ XNLI using ESIM





F1 Score (%) on NSD

- To finetune The embeddings on NSD task, we used stack of 6 transformer encoder layers.
- The output is then forwarded to a soft-max layer to choose the meaning of the word with the highest probability.



THANK YOU

Ack: H. Abdine, G. Shang, P. Meladianos, A. Tixier

References (1)

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. arXiv preprint arXiv:1705.02364.

Chopra, Sumit, Raia Hadsell, and Yann LeCun. "Learning a similarity metric discriminatively, with application to face verification." *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, 2005.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., & Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*.

Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." *Computer vision and pattern recognition, 2006 IEEE computer society conference on*. Vol. 2. IEEE, 2006.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015).

Mueller, J., & Thyagarajan, A. (2016, February). Siamese Recurrent Architectures for Learning Sentence Similarity. In *AAAI* (pp. 2786-2792).

References (2)

Neculoiu, Paul, Maarten Versteegh, and Mihai Rotaru. "Learning text similarity with siamese recurrent networks." *Proceedings of the 1st Workshop on Representation Learning for NLP*. 2016.

Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." arXiv preprint arXiv:1509.00685 (2015).

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning (pp. 2048-2057).

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. NAACL 2016 (pp. 1480-1489).

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., & Huang, F. (2006). A tutorial on energy-based learning. *Predicting structured data*, *1*(0).

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, *2*(1), 1-127.

References (3)

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1994). Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems* (pp. 737-744).

Neculoiu, P., Versteegh, M., & Rotaru, M. (2016). Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP* (pp. 148-157).

Hoffer, E., & Ailon, N. (2015, October). Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition* (pp. 84-92). Springer, Cham.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

Dor, L. E., Mass, Y., Halfon, A., Venezian, E., Shnayderman, I., Aharonov, R., & Slonim, N. (2018). Learning Thematic Similarity Metric from Article Sections Using Triplet Networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Vol. 2, pp. 49-54).

Bellet, A., Habrard, A., & Sebban, M. (2013). A survey on metric learning for feature vectors and structured data.

Chen, W., Chen, X., Zhang, J., & Huang, K. (2017, July). Beyond triplet loss: a deep quadruplet network for person re-identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Vol. 2, No. 8).

References (4)

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015).

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

Peters, Matthew E., et al. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

References

- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

References

- Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. The Journal of Machine Learning Research, 3, 1137–1155. http://doi.org/10.1162/153244303322533223
- Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), 1–12.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS, 1–9.
- Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing. Proceedings of the 25th International Conference on Machine Learning - ICML '08, 20(1), 160–167. http://doi.org/10.1145/1390156.1390177
- Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016). Character-Aware Neural Language Models. AAAI. Retrieved from http://arxiv.org/abs/1508.06615
- Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., & Wu, Y. (2016). Exploring the Limits of Language Modeling. Retrieved from http://arxiv.org/abs/1602.02410
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (almost) from Scratch. Journal of Machine Learning Research, 12 (Aug), 2493–2537. Retrieved from http://arxiv.org/abs/1103.0398
- Chen, W., Grangier, D., & Auli, M. (2015). Strategies for Training Large Vocabulary Neural Language Models, 12. Retrieved from http://arxiv.org/abs/1512.04906

References

- Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving Distributional Similarity with Lessons Learned from Word Embeddings. Transactions of the Association for Computational Linguistics, 3, 211–225. Retrieved from https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/570
- Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 1532–1543. http://doi.org/10.3115/v1/D14-1162
- Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. ACL, 238–247. http://doi.org/10.3115/v1/P14-1023
- Levy, O., & Goldberg, Y. (2014). Neural Word Embedding as Implicit Matrix Factorization. Advances in Neural Information Processing Systems (NIPS), 2177–2185. Retrieved from http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization
- Hamilton, W. L., Clark, K., Leskovec, J., & Jurafsky, D. (2016). Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Retrieved from http://arxiv.org/abs/1606.02820
- Hamilton, W. L., Leskovec, J., & Jurafsky, D. (2016). Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. arXiv Preprint arXiv:1605.09096.

References- blogs

- Sebastian Ruder blog series on Word Embeddings, http://sebastianruder.com/
- Andy Jones blog on word2vec, http://andyljones.tumblr.com/post/111299309808/why-word2vec-works
- Arora et al, https://arxiv.org/pdf/1502.03520v7.pdf
- Piotr Migdał , http://p.migdal.pl/2017/01/06/king-man-woman-queen-why.html