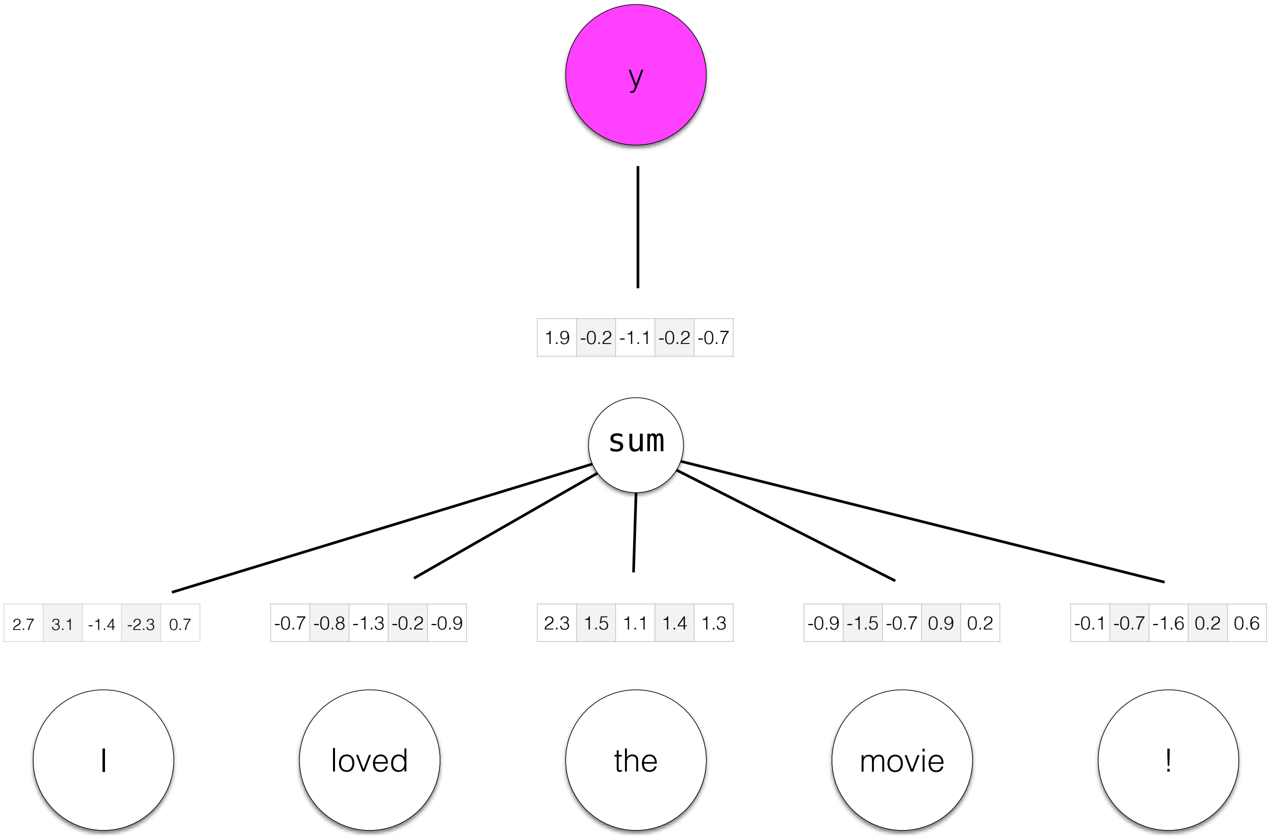
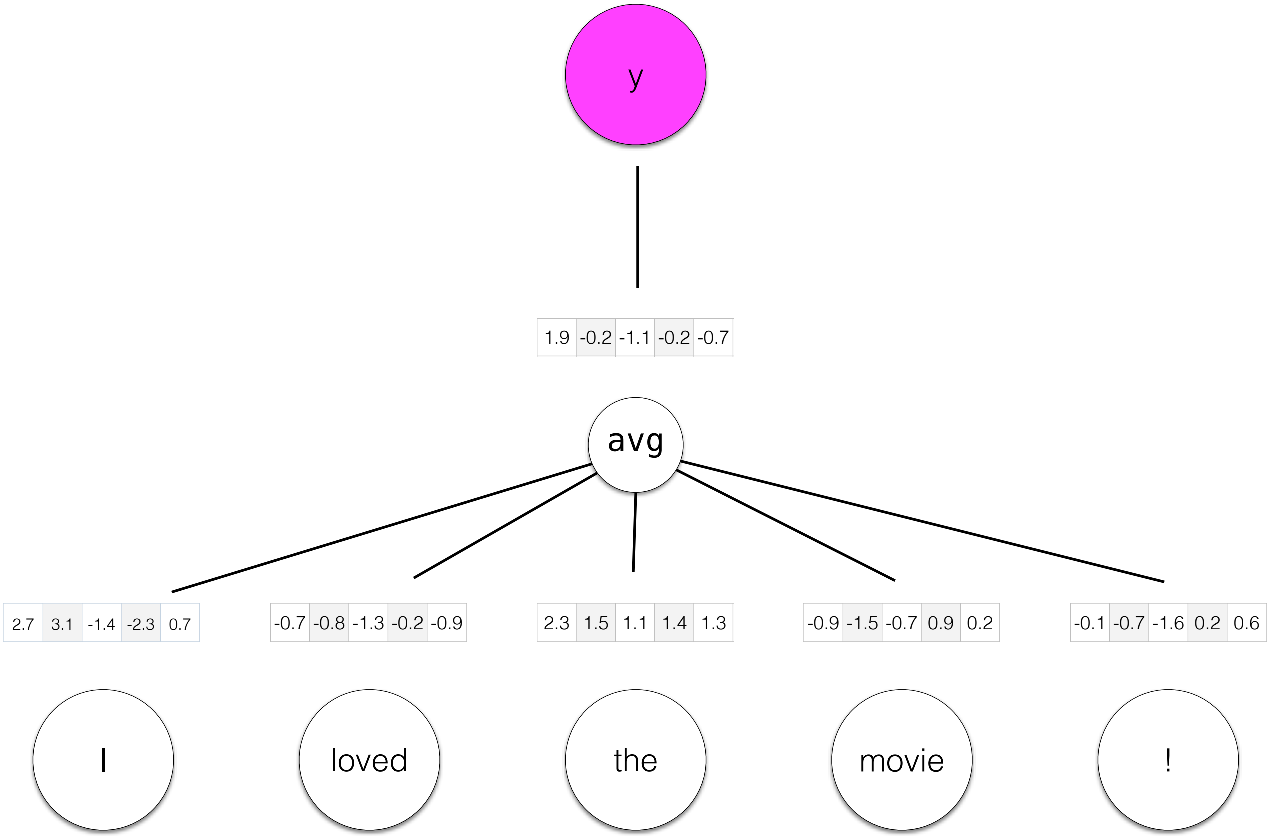
# Applied Natural Language Processing

Info 256
Lecture 15: Attention/BERT (Oct. 14, 2021)

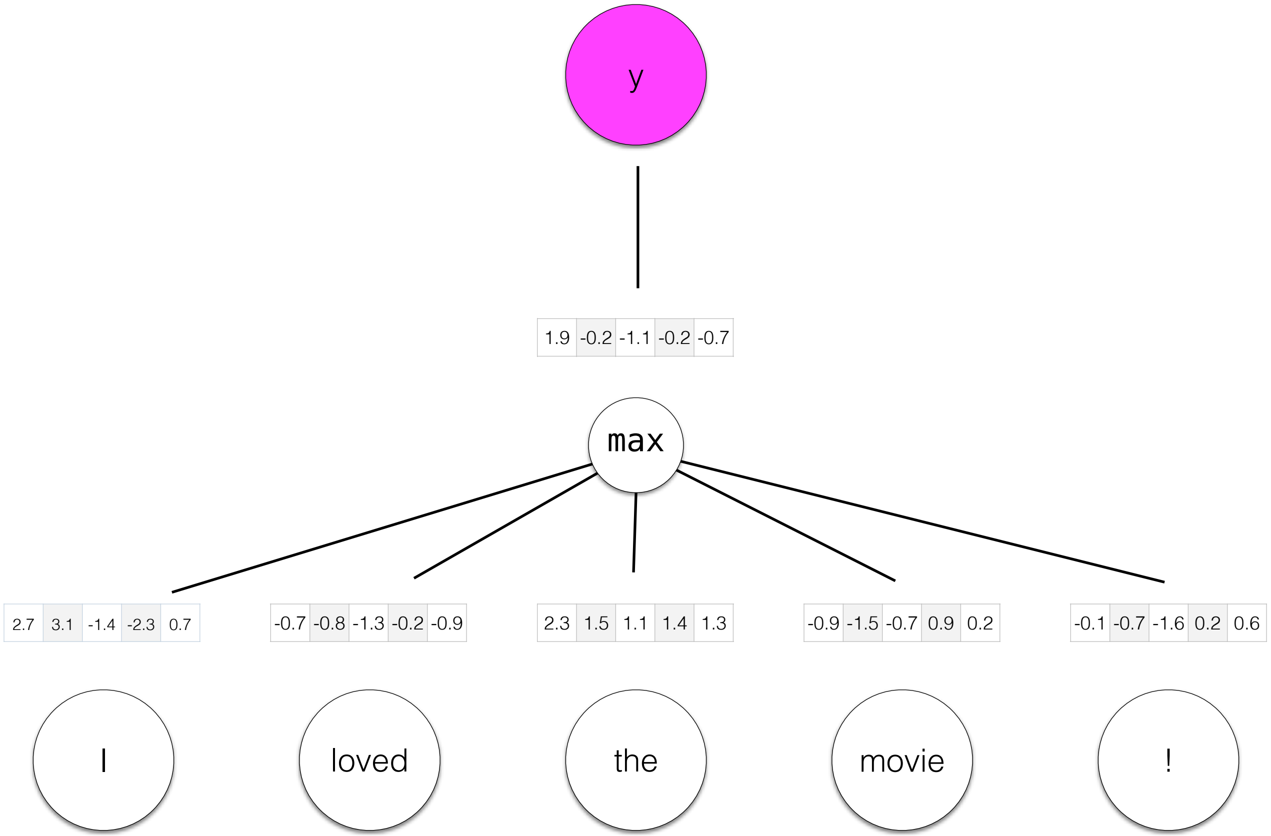
David Bamman, UC Berkeley

How do we use word embeddings for document classification?

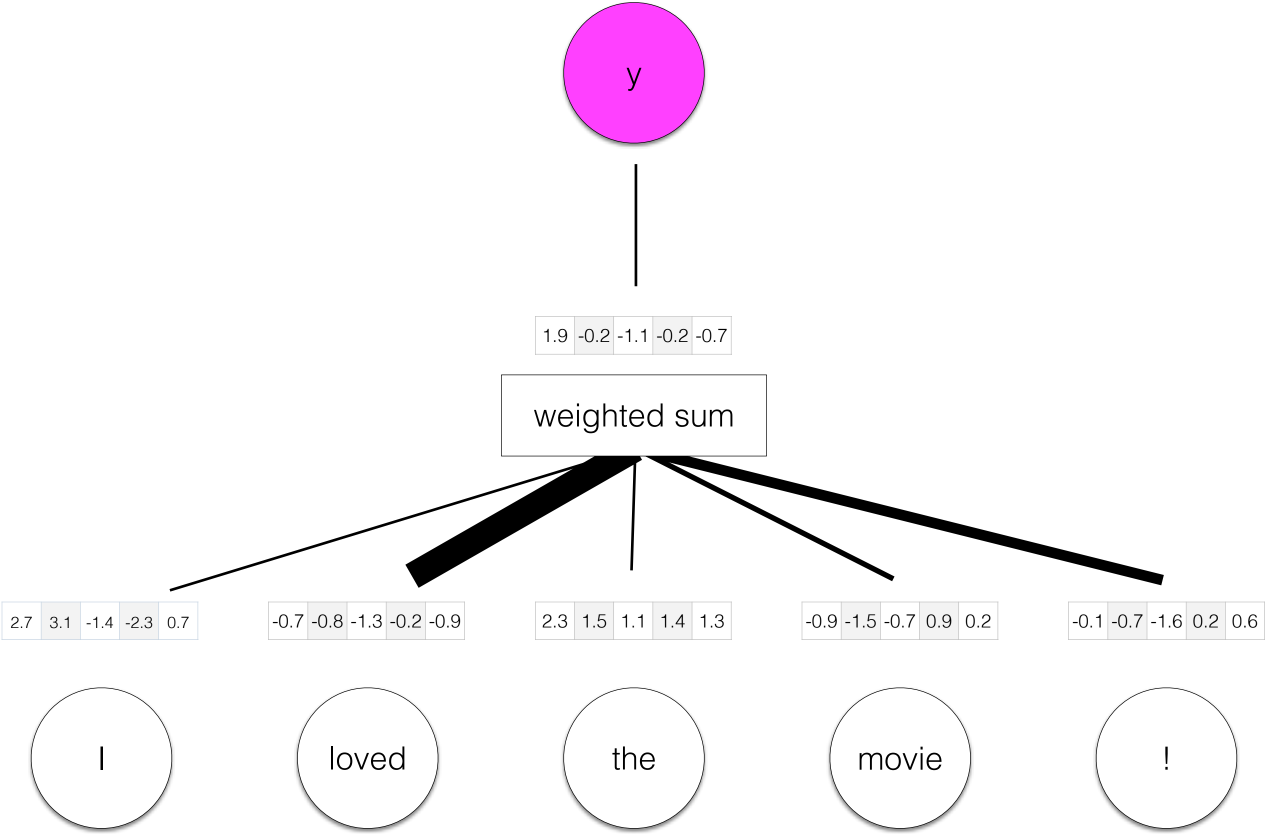

y
1.9 -0.2 -1.1 -0.2 -0.7
sum
2.7 3.1 -1.4 -2.3 0.7
-0.7 -0.8 -1.3 -0.2 -0.9
2.3 1.5 1.1 1.4 1.3
-0.9 -1.5 -0.7 0.9 0.2
-0.1 -0.7 -1.6 0.2 0.6
I
loved
the
movie
!




Iyyer et al. (2015), “Deep Unordered Composition Rivals Syntactic Methods for Text Classification” (ACL)



y
1.9 -0.2 -1.1 -0.2 -0.7
max
2.7 3.1 -1.4 -2.3 0.7
-0.7 -0.8 -1.3 -0.2 -0.9
2.3 1.5 1.1 1.4 1.3
-0.9 -1.5 -0.7 0.9 0.2
-0.1 -0.7 -1.6 0.2 0.6
I
loved
the
movie
!


y
1.9 -0.2 -1.1 -0.2 -0.7
weighted sum
2.7 3.1 -1.4 -2.3 0.7
-0.7 -0.8 -1.3 -0.2 -0.9
2.3 1.5 1.1 1.4 1.3
-0.9 -1.5 -0.7 0.9 0.2
-0.1 -0.7 -1.6 0.2 0.6
I
loved
the
movie
!

# Attention

- Let's incorporate structure (and parameters) into a network that captures which elements in the input we should be attending to (and which we can ignore).

$$v \in \mathscr{R}^H$$

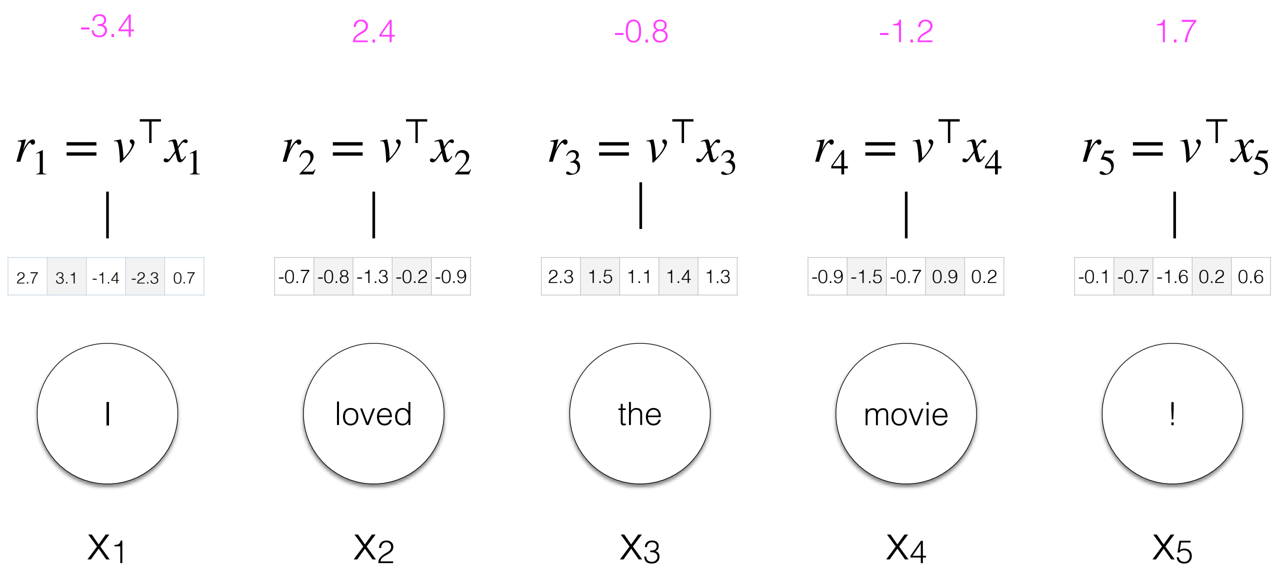
| | | | | |
|---|---|---|---|---|
| 2.7 | 3.1 | -1.4 | -2.3 | 0.7 |

Define v to be a vector to be learned; think of it as an "important word" vector. The dot product here measures how similar each input vector is to that "important word" vector

| | | | | |
|---|---|---|---|---|
| 2.7 3.1 -1.4 -2.3 0.7 | -0.7 -0.8 -1.3 -0.2 -0.9 | 2.3 1.5 1.1 1.4 1.3 | -0.9 -1.5 -0.7 0.9 0.2 | -0.1 -0.7 -1.6 0.2 0.6 |
| I | loved | the | movie | ! |
| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ |

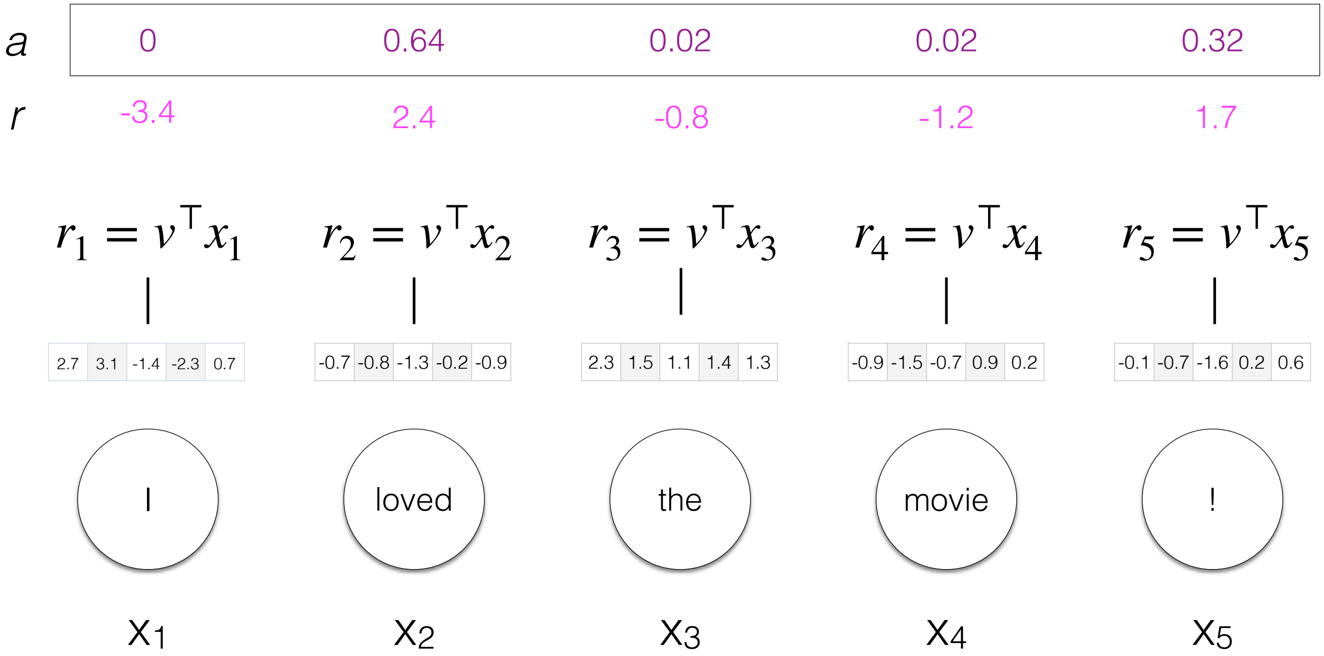
$$v \in \mathscr{R}^H$$

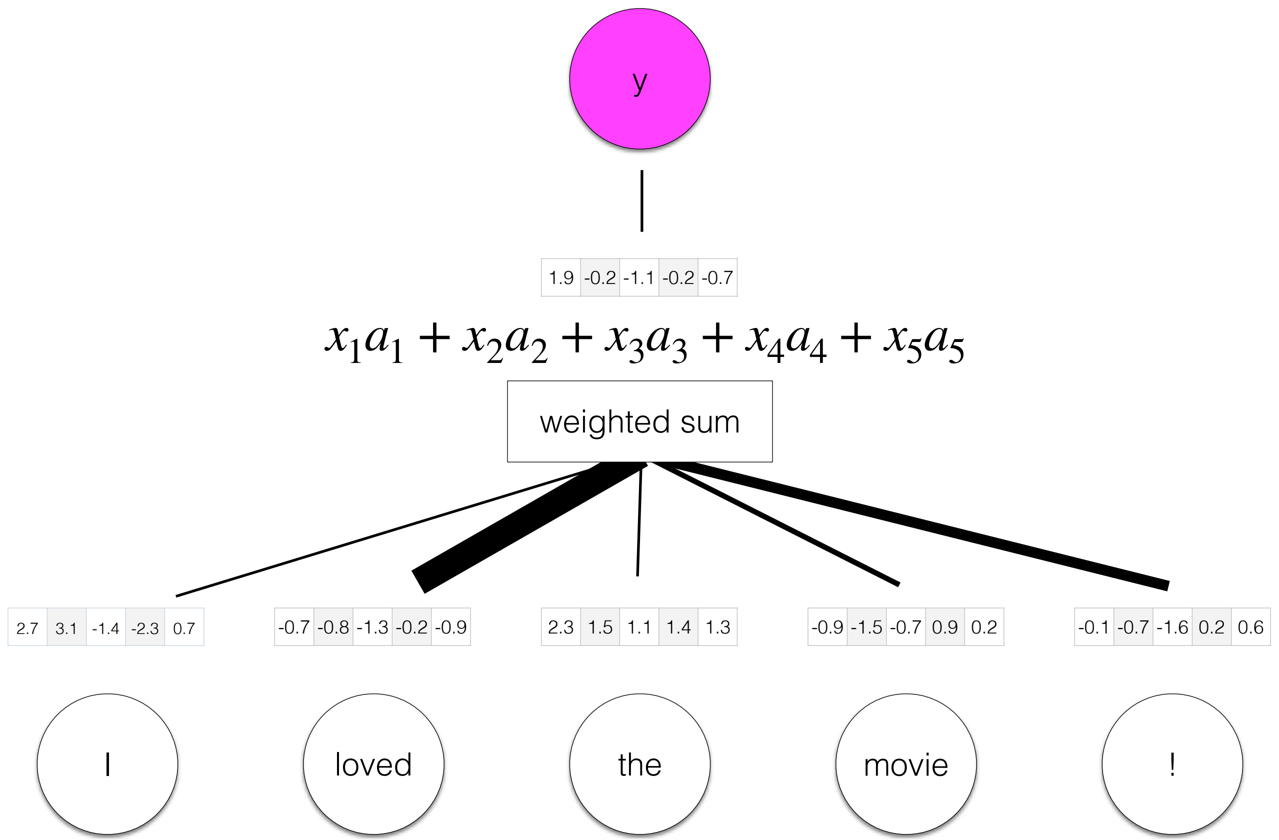
| | | | | |
|---|---|---|---|---|
| 2.7 | 3.1 | -1.4 | -2.3 | 0.7 |



Convert r into a vector of normalized weights that sum to 1.

$$a = \text{softmax}(r)$$




y
1.9 -0.2 -1.1 -0.2 -0.7
$x_1a_1 + x_2a_2 + x_3a_3 + x_4a_4 + x_5a_5$
weighted sum
2.7 3.1 -1.4 -2.3 0.7
-0.7 -0.8 -1.3 -0.2 -0.9
2.3 1.5 1.1 1.4 1.3
-0.9 -1.5 -0.7 0.9 0.2
-0.1 -0.7 -1.6 0.2 0.6
I
loved
the
movie
!

# Attention

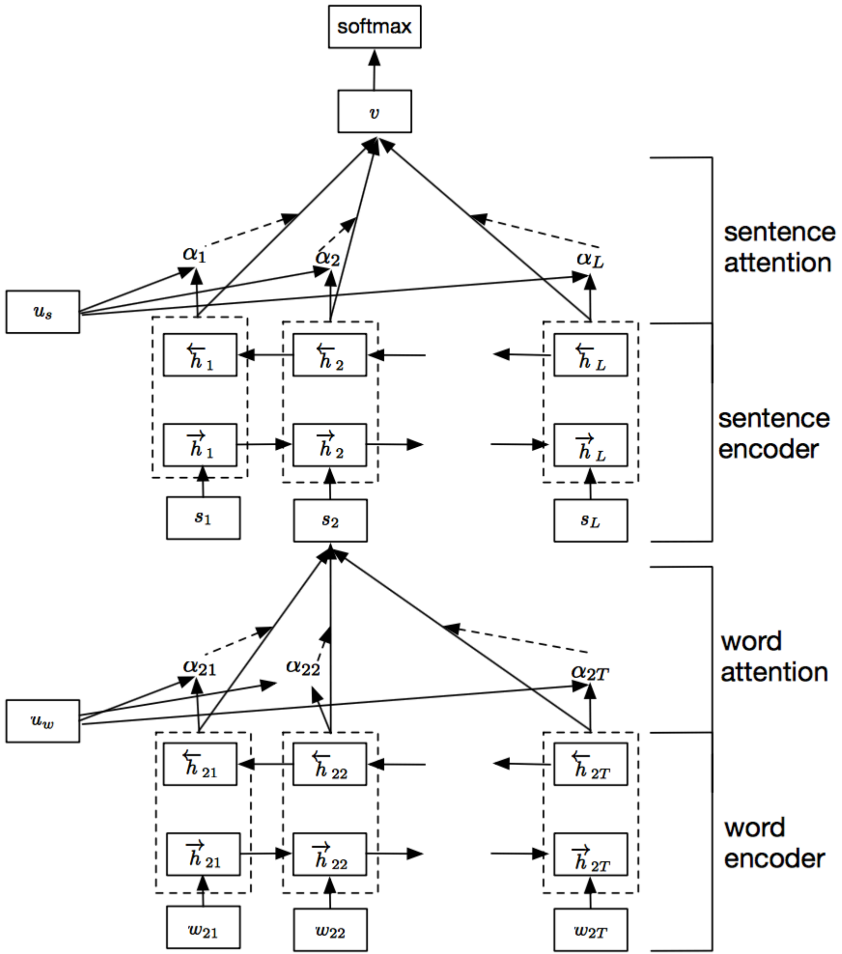
- Lots of variations on attention:
    - Linear transformation of x into before dotting with v
    - Non-linearities after each operation.
    - “Multi-head attention”: multiple v vectors to capture different phenomena that can be attended to in the input.
    - Hierarchical attention (sentence representation with attention over words + document representation with attention over sentences).

attention over sentences

bidirectional GRU over sentence representations

attention over words

bidirectional GRU over word representations

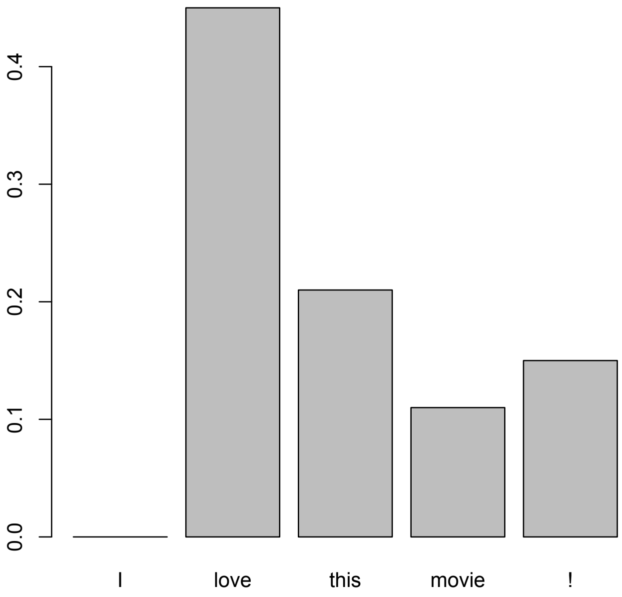



Yang et al. (2016), "Hierarchical Attention Networks for Document Classification"

# Attention

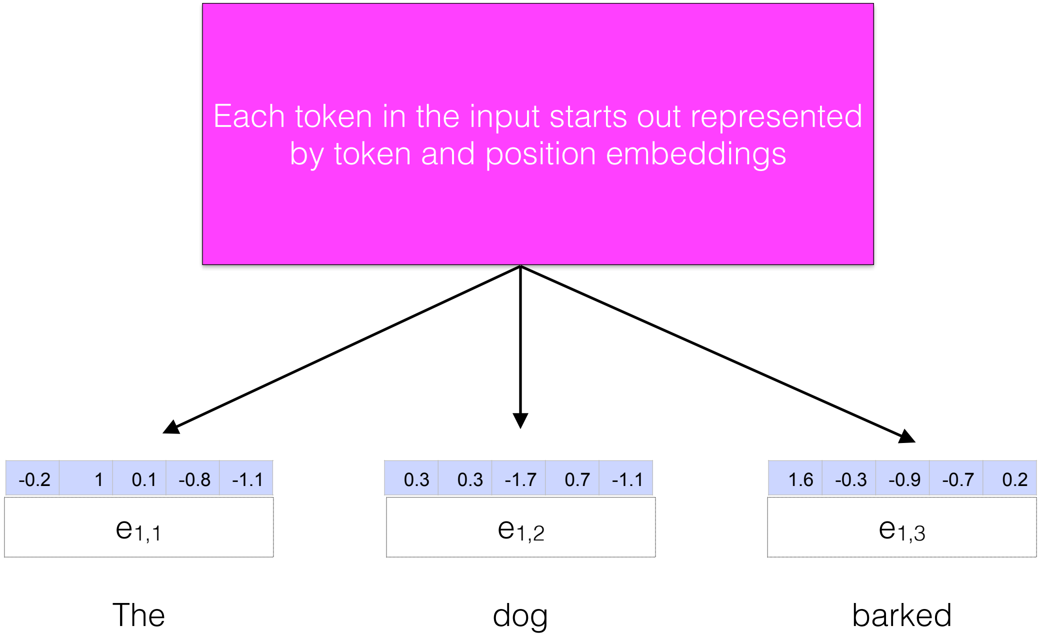
- Attention gives us a normalized weight for every token in a sequence that tells us how important that word was for the prediction

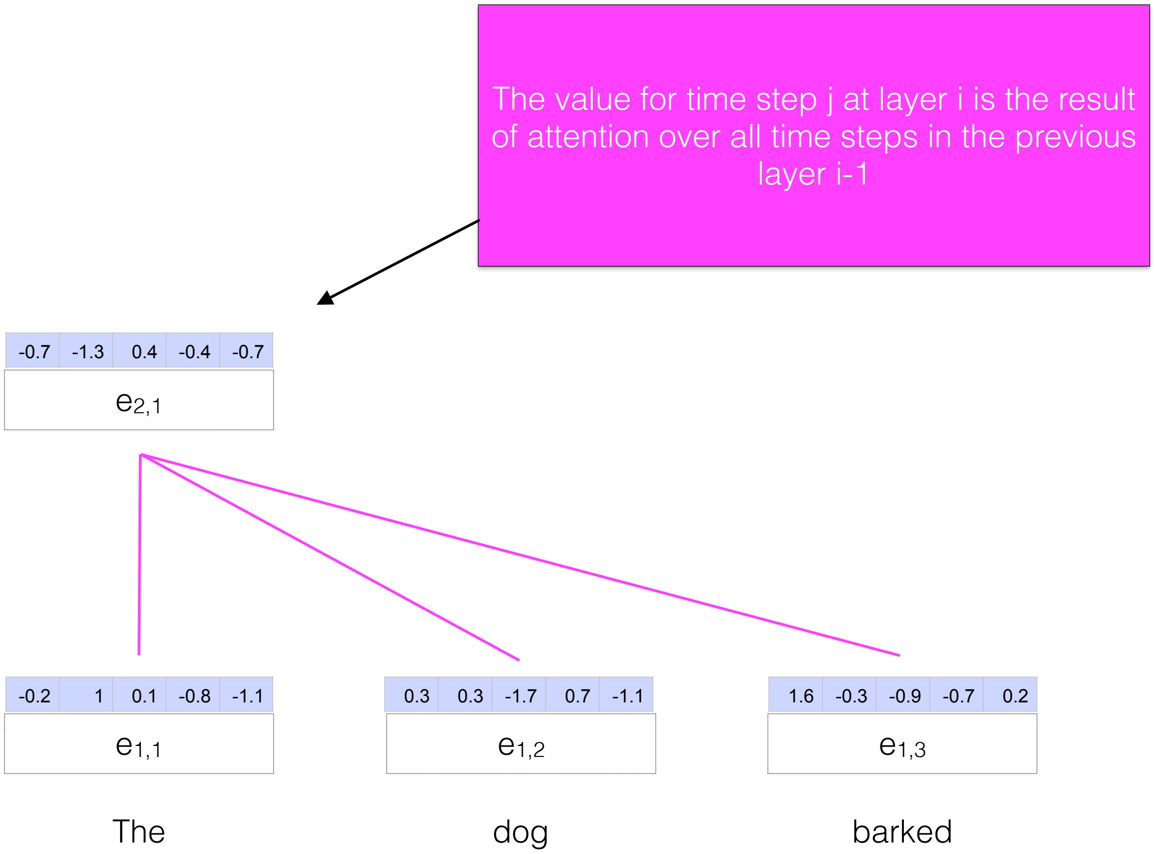
- This can be useful for visualization

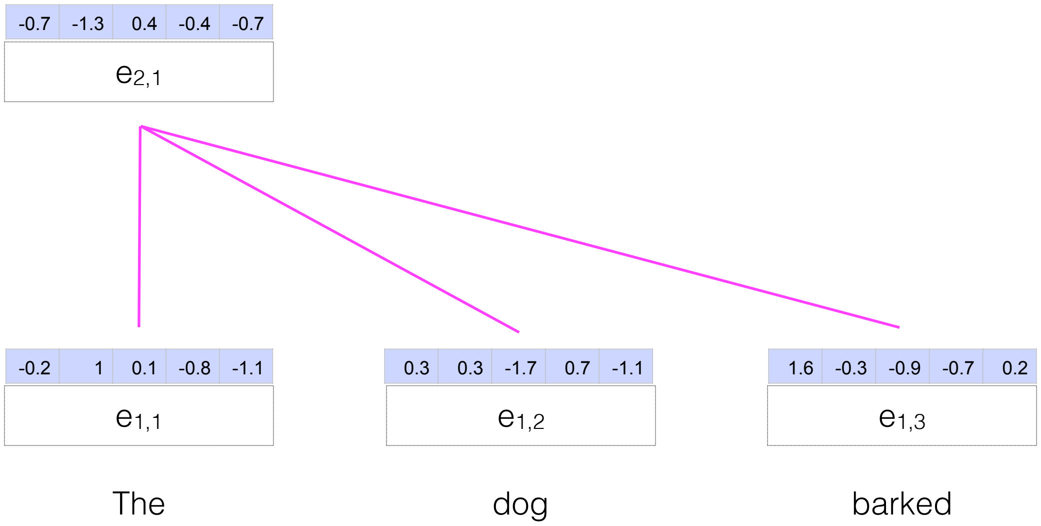


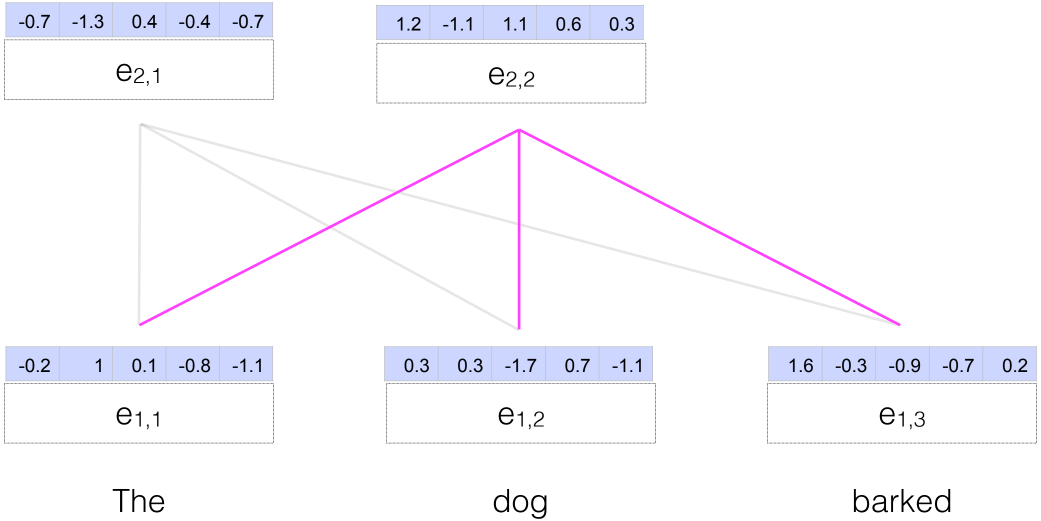
# BERT

- Transformer-based model (Vaswani et al. 2017) to predict masked word using bidirectional context + next sentence prediction.
- Generates multiple layers of representations for each token sensitive to its context of use.

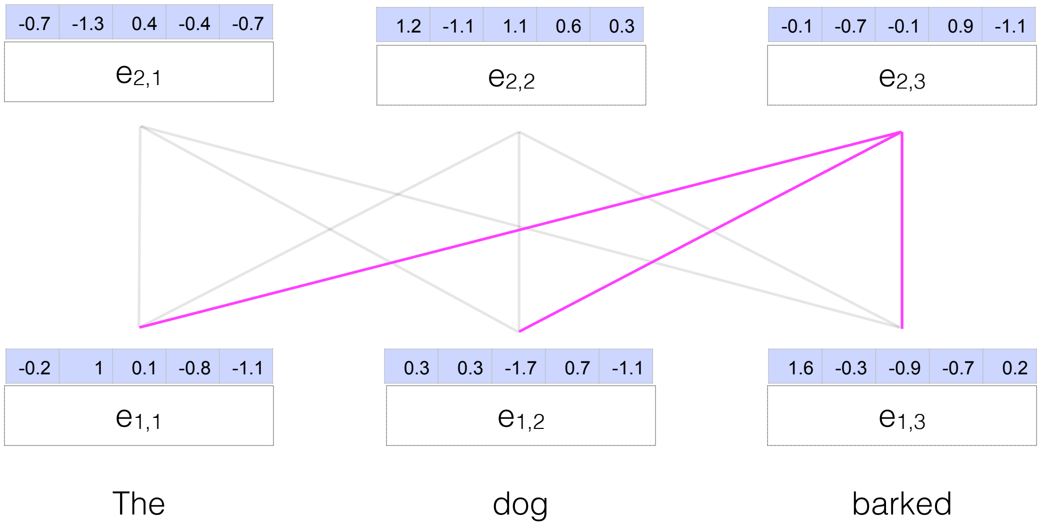

Each token in the input starts out represented by token and position embeddings
-0.2 1 0.1 -0.8 -1.1
$e_{1,1}$
The
0.3 0.3 -1.7 0.7 -1.1
$e_{1,2}$
dog
1.6 -0.3 -0.9 -0.7 0.2
$e_{1,3}$
barked


The value for time step j at layer i is the result of attention over all time steps in the previous layer i-1
-0.7 -1.3 0.4 -0.4 -0.7
$e_{2,1}$
-0.2 1 0.1 -0.8 -1.1
$e_{1,1}$
0.3 0.3 -1.7 0.7 -1.1
$e_{1,2}$
1.6 -0.3 -0.9 -0.7 0.2
$e_{1,3}$
The
dog
barked

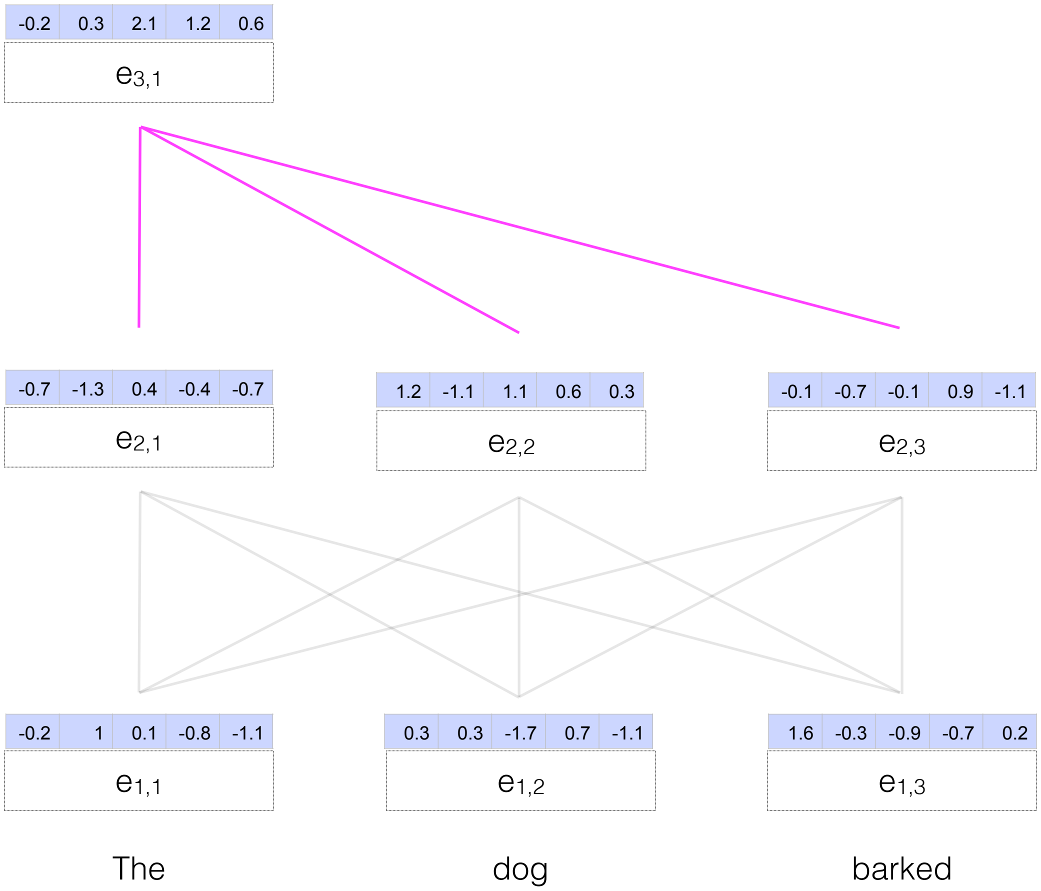

-0.7 -1.3 0.4 -0.4 -0.7
e2,1
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked

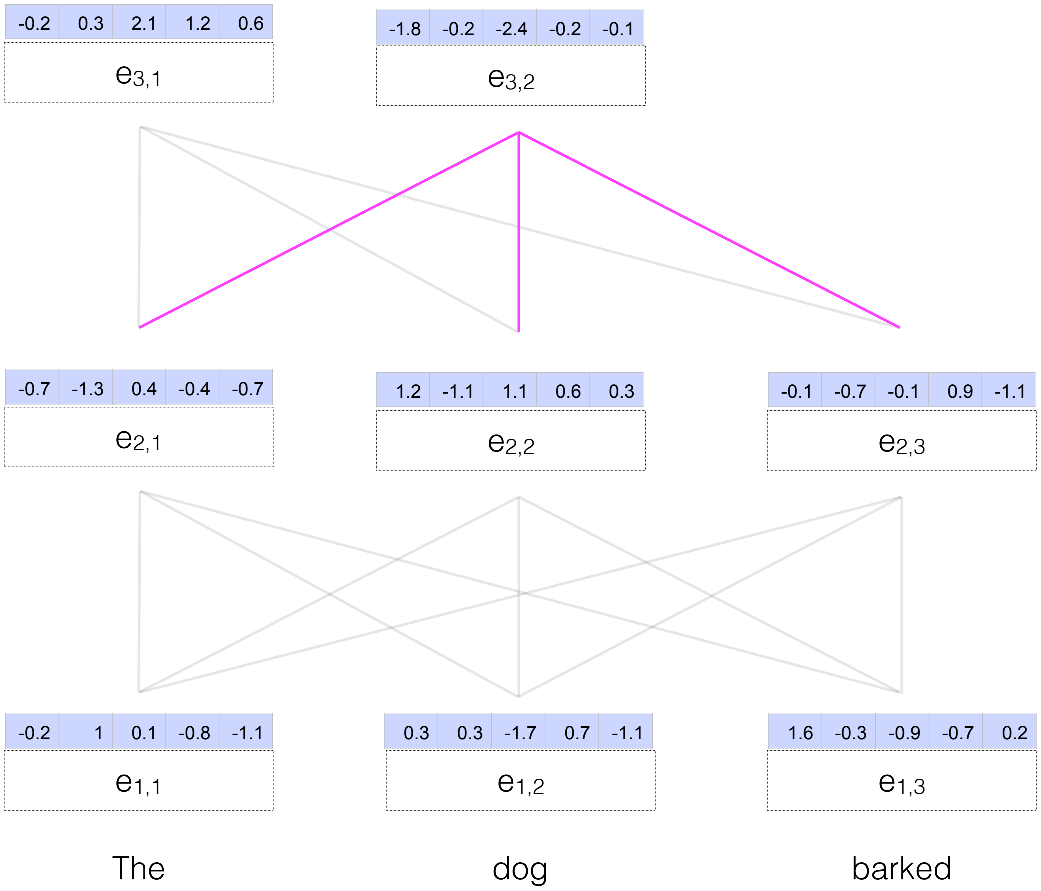


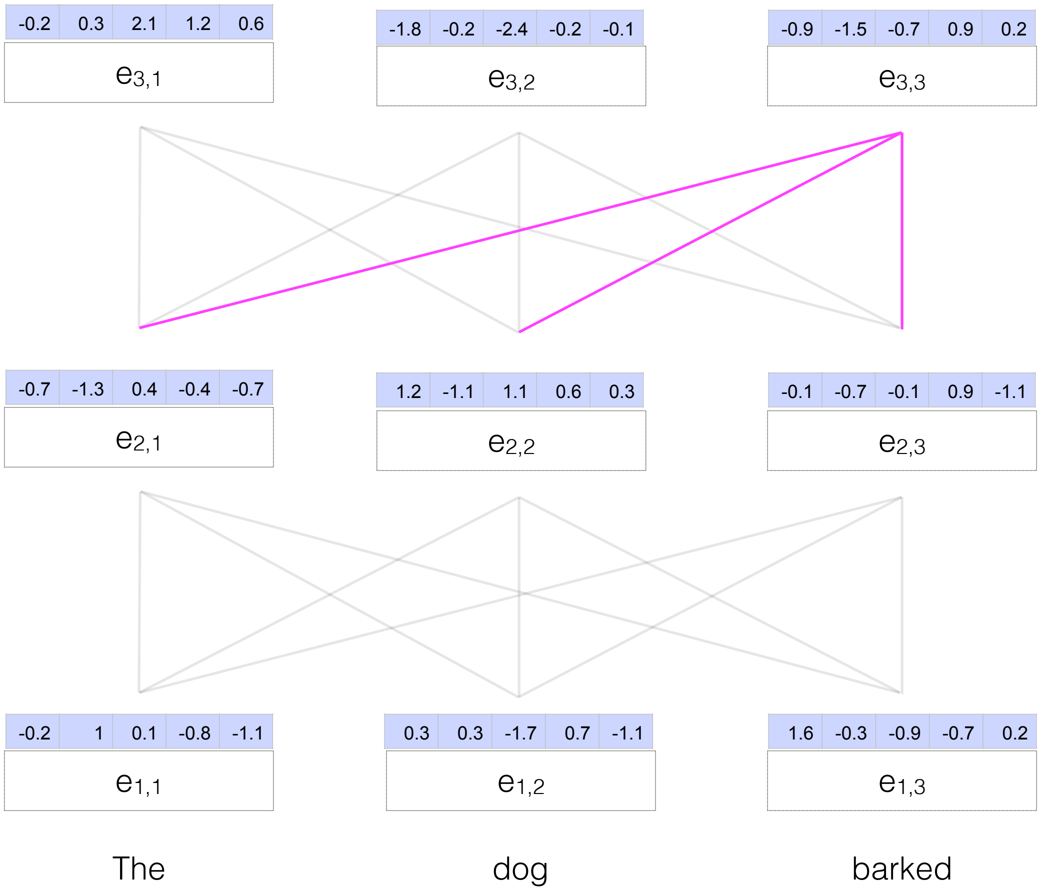
-0.7 -1.3 0.4 -0.4 -0.7
e2,1
1.2 -1.1 1.1 0.6 0.3
e2,2
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked



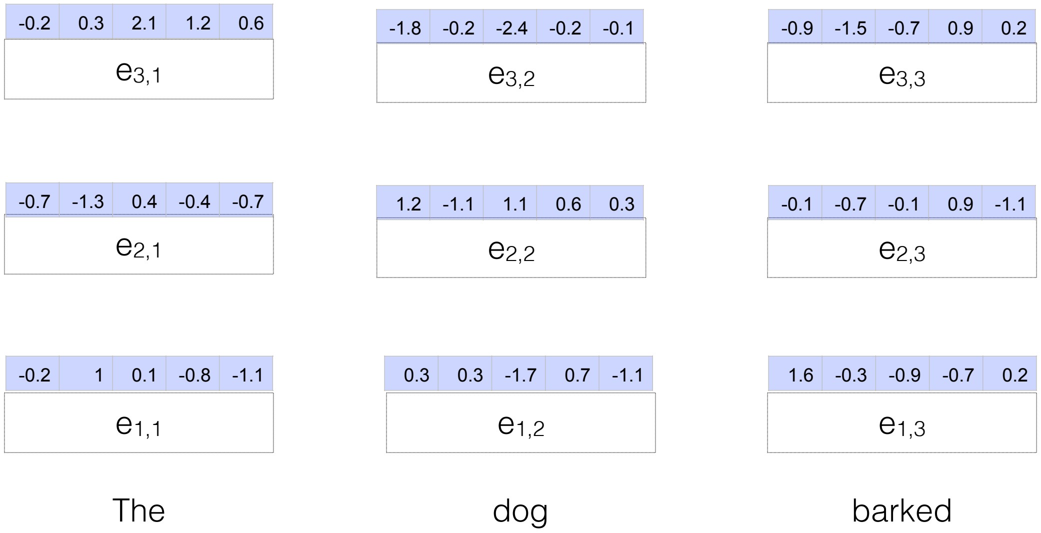
-0.7 -1.3 0.4 -0.4 -0.7
$e_{2,1}$
1.2 -1.1 1.1 0.6 0.3
$e_{2,2}$
-0.1 -0.7 -0.1 0.9 -1.1
$e_{2,3}$
-0.2 1 0.1 -0.8 -1.1
$e_{1,1}$
0.3 0.3 -1.7 0.7 -1.1
$e_{1,2}$
1.6 -0.3 -0.9 -0.7 0.2
$e_{1,3}$
The
dog
barked


-0.2 0.3 2.1 1.2 0.6
e3,1
-0.7 -1.3 0.4 -0.4 -0.7
e2,1
1.2 -1.1 1.1 0.6 0.3
e2,2
-0.1 -0.7 -0.1 0.9 -1.1
e2,3
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked


-0.2 0.3 2.1 1.2 0.6
$e_{3,1}$
-1.8 -0.2 -2.4 -0.2 -0.1
$e_{3,2}$
-0.7 -1.3 0.4 -0.4 -0.7
$e_{2,1}$
1.2 -1.1 1.1 0.6 0.3
$e_{2,2}$
-0.1 -0.7 -0.1 0.9 -1.1
$e_{2,3}$
-0.2 1 0.1 -0.8 -1.1
$e_{1,1}$
0.3 0.3 -1.7 0.7 -1.1
$e_{1,2}$
1.6 -0.3 -0.9 -0.7 0.2
$e_{1,3}$
The
dog
barked


-0.2 0.3 2.1 1.2 0.6
e3,1
-1.8 -0.2 -2.4 -0.2 -0.1
e3,2
-0.9 -1.5 -0.7 0.9 0.2
e3,3
-0.7 -1.3 0.4 -0.4 -0.7
e2,1
1.2 -1.1 1.1 0.6 0.3
e2,2
-0.1 -0.7 -0.1 0.9 -1.1
e2,3
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked

At the end of this process, we have one representation for each layer for each token



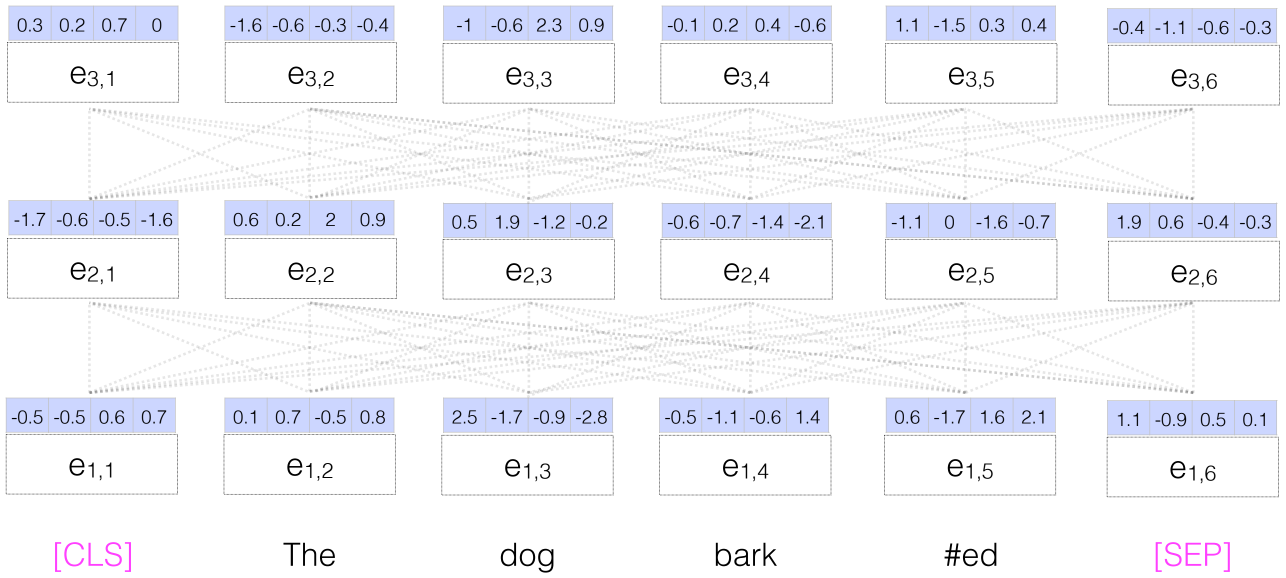
# WordPiece

- BERT uses WordPiece tokenization, which segments some morphological structure of tokens
- Vocabulary size: 30,000

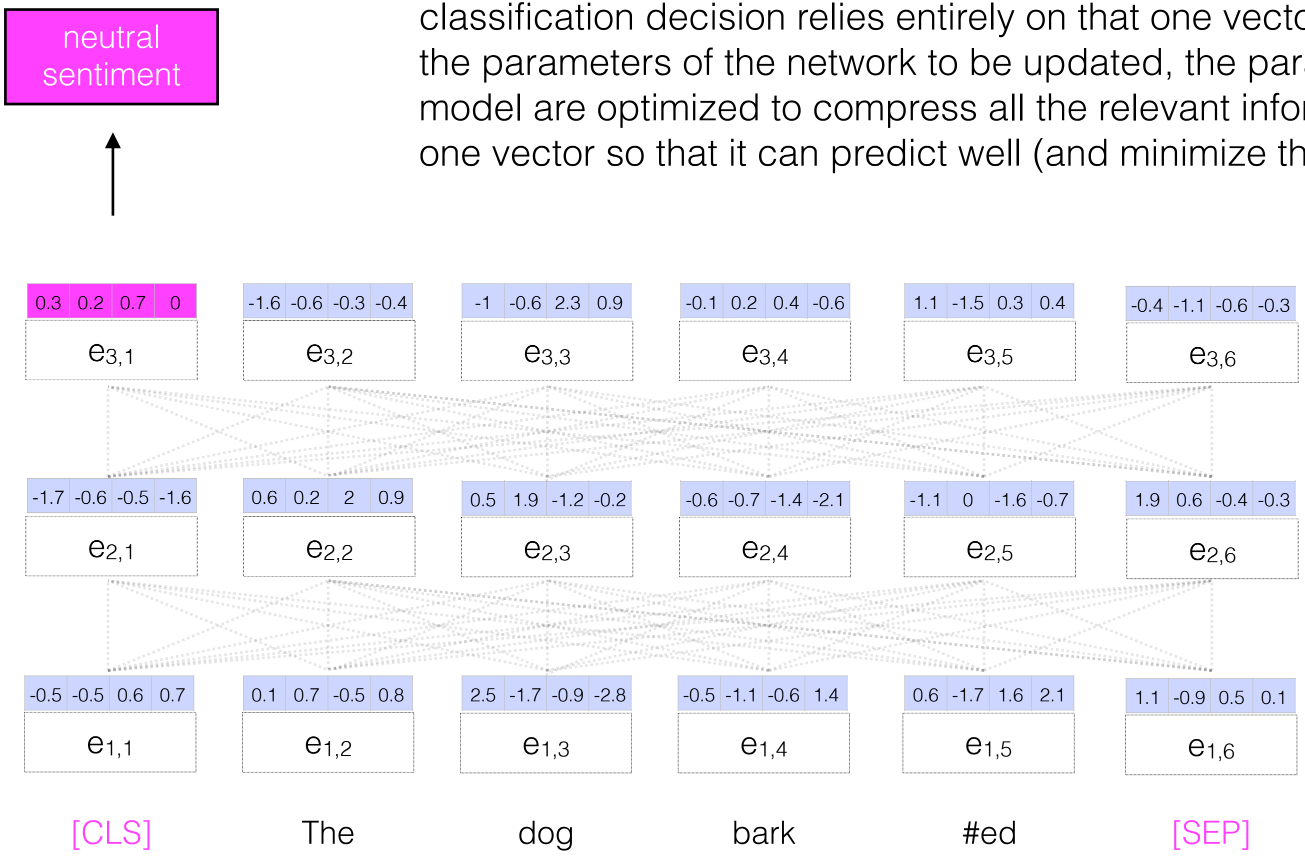
| | |
|---|---|
| The | The |
| dog | dog |
| barked | bark #ed |

- BERT also encodes each sentence by appending a special token to the beginning ([CLS]) and end ([SEP]) of each sequence.

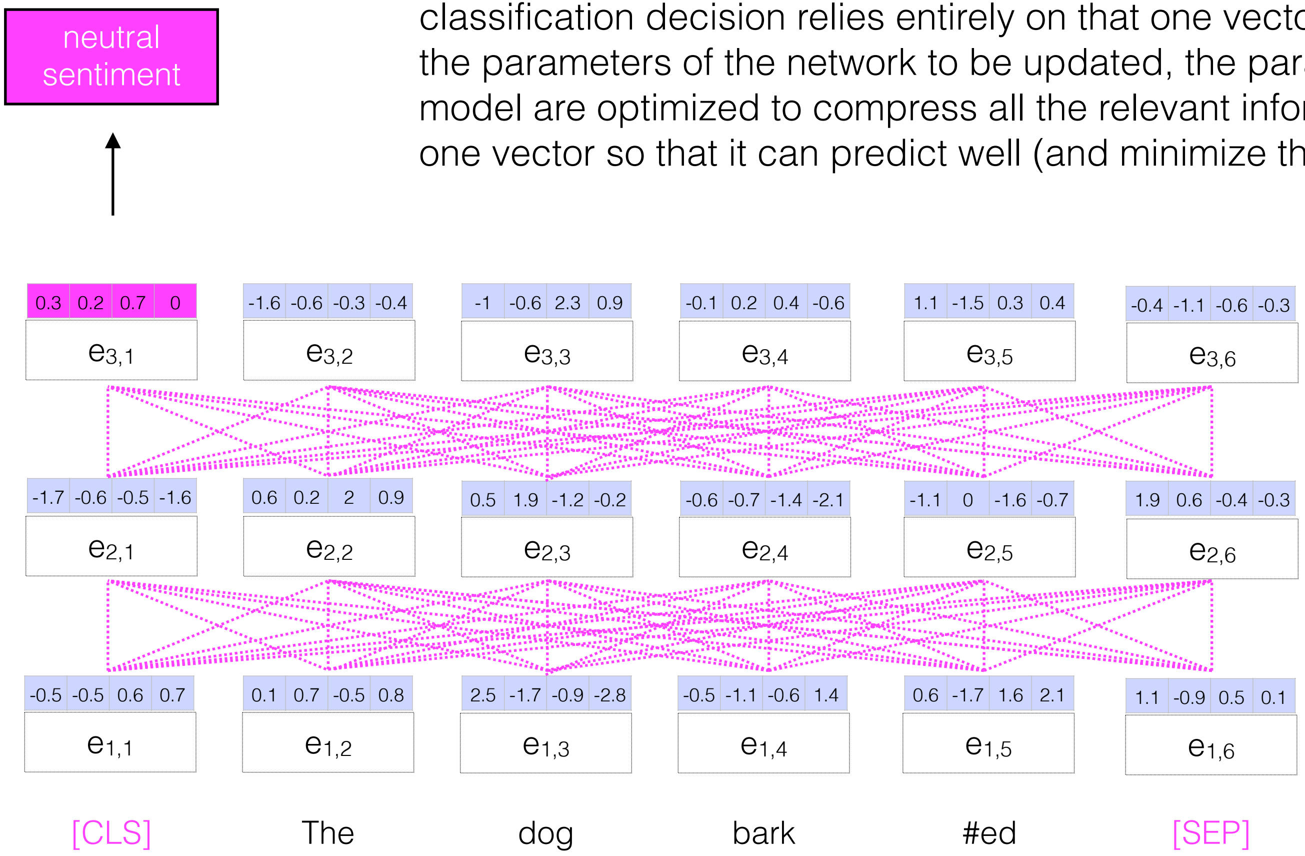
- This helps provides a single token that can be optimized to represent the entire sequence (e.g., for document classification)



- We can represent the entire document with this *one* [CLS] vector
- Why does this work? When we design our network so that a classification decision relies entirely on that one vector and allow all the parameters of the network to be updated, the parameters of the model are optimized to compress all the relevant information into that one vector so that it can predict well (and minimize the loss).



- We can represent the entire document with this *one* [CLS] vector
- Why does this work? When we design our network so that a classification decision relies entirely on that one vector and allow all the parameters of the network to be updated, the parameters of the model are optimized to compress all the relevant information into that one vector so that it can predict well (and minimize the loss).



# BERT

- Deep layers (12 for BERT base, 24 for BERT large)
- Large representation sizes (768 per layer)
- Pretrained on English Wikipedia (2.5B words) and BooksCorpus (800M words)

# BERT

| | H=128 | H=256 | H=512 | H=768 |
|---|---|---|---|---|
| L=2 | **2/128 (BERT-Tiny)** | 2/256 | 2/512 | 2/768 |
| L=4 | 4/128 | **4/256 (BERT-Mini)** | **4/512 (BERT-Small)** | 4/768 |
| L=6 | 6/128 | 6/256 | 6/512 | 6/768 |
| L=8 | 8/128 | 8/256 | **8/512 (BERT-Medium)** | 8/768 |
| L=10 | 10/128 | 10/256 | 10/512 | 10/768 |
| L=12 | 12/128 | 12/256 | 12/512 | **12/768 (BERT-Base)** |

https://github.com/google-research/bert

SIGN IN MODELS FORUM

v4.11.3 ▼

transformers

Star 52,449

Search docs

GET STARTED

- Quick tour
- Installation
- Philosophy
- Glossary

USING 🤗 TRANSFORMERS

- Summary of the tasks
- Summary of the models
- Preprocessing data

Docs » Pretrained models

View page source

# Pretrained models

Here is a partial list of some of the available pretrained models together with a short presentation of each model.

For the full list, refer to https://huggingface.co/models.

| Architecture | Model id | Details of the model |
|---|---|---|
| | `bert-base-uncased` | 12-layer, 768-hidden, 12-heads, 110M parameters.<br>Trained on lower-cased English text. |
| | `bert-large-uncased` | 24-layer, 1024-hidden, 16-heads, 336M parameters.<br>Trained on lower-cased English text. |

https://huggingface.co/transformers/pretrained_models.html

BERT LANG STREET

## Lost in (language-specific) BERT models? We are here to help!

We currently have indexed **31** BERT-based models, **19** Languages and **28** Tasks.
We have a total of **178** entries in this table; we also show **Multilingual Bert (mBERT)** results if available! (see our paper)

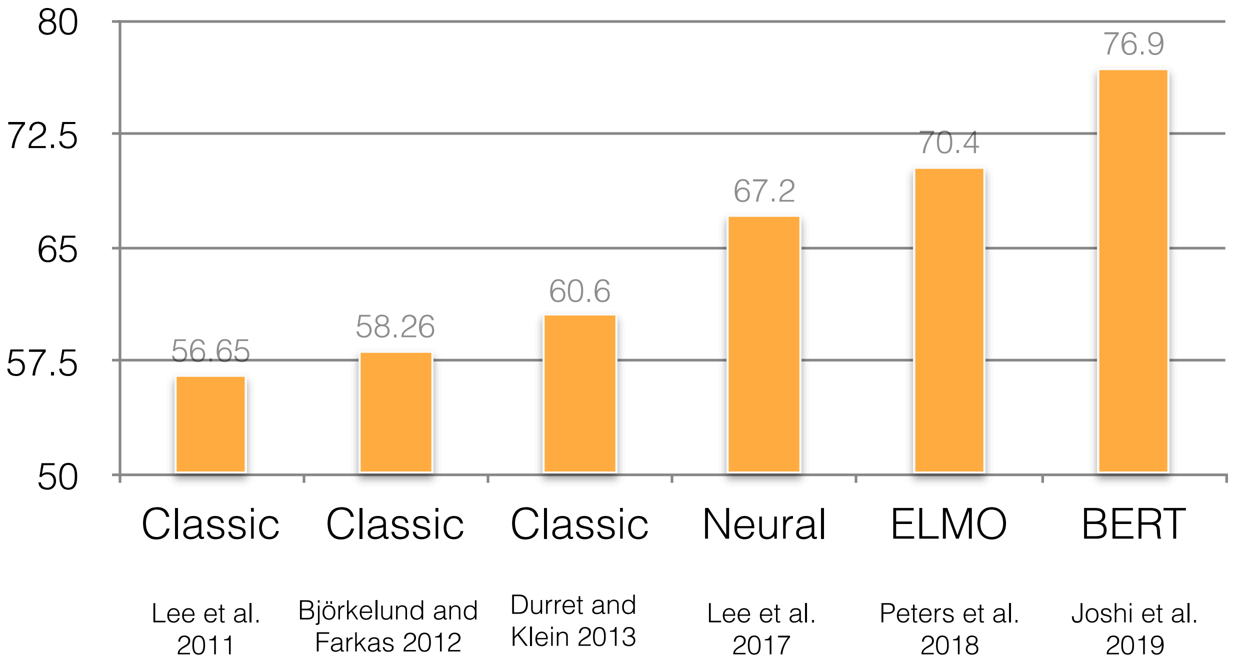
Curious which BERT model is the best for named entity recognition in Italian? Just type ***"Italian NER"*** in the search bar!

Show 10 entries                                        Search:

| Language | Model | NLP Task | Dataset | Dataset-Domain | Measure | Performance | mBERT | Difference with mBERT | Source |
|---|---|---|---|---|---|---|---|---|---|
| Arabic | Arabert v1 | SA | AJGT | twitter | Accuracy | 93.8 | 83.6 | 10.2 | |
| Arabic | Arabert v1 | SA | HARD | hotel reviews | Accuracy | 96.1 | 95.7 | 0.4 | |
| Arabic | Arabert v1 | SA | ASTD | twitter | Accuracy | 92.6 | 80.1 | 12.5 | |
| Arabic | Arabert v1 | SA | ArSenTD-Lev | twitter | Accuracy | 59.4 | 51.0 | 8.4 | |

https://bertlang.unibocconi.it

# Progress — Coreference resolution



# Bertology

- Hewitt et al. 2019
- Tenney et al. 2019
- McCoy et al. 2019
- Liu et al. 2019
- Clark et al. 2019
- Goldberg 2019
- Michel et al. 2019

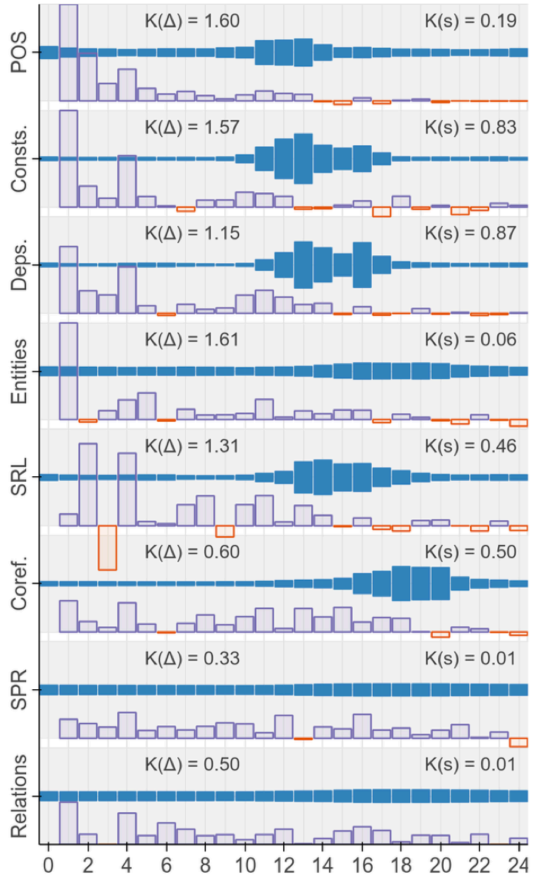
# Code

Pre-trained models for BERT, Transformer-XL, ALBERT, RoBERTa, DistilBERT, GPT-2, etc. for English, French, "Multilingual"

https://huggingface.co

# Probing

- Even though BERT is mainly trained on a language modeling objective, it learns a lot about the structure of language — even without direct training data for specific linguistic tasks.
- Probing experiments uncover what—-and where (in what layers)—-pretrained BERT encodes this information.




Tenney et al. (2019), "BERT Rediscovers the Classical NLP Pipeline"

# Activity

`9.neural/BERTClassification`

- Explore BERT for document classification using Google Colab