Advanced Topics in Machine Learning

Alejo Nevado-Holgado

Lecture 9 (NLP 1) - Introduction and embeddings 1

V 0.6 (13 Feb 2020 - final version)

Course structure

- ➢ This course will explain to you how to use neural networks for natural language processing (NLP)
- ➢ This course is organized in a **problem-oriented manner**:
  - We will explain NNs NLP by introducing the typical NLP problems that the field tries to solve. While introducing each problem, we will explain the NN methods that were created to solve that problem
- ➢ Other courses are organized in a **method-oriented manner**:
  - They explain NNs NLP by introducing the typical NNs methods used in NLP. While introducing each method, they explain some of the NLP problems that they solve

Course structure

- ➢ **Introduction:** What is NLP. Why it is hard. Why NNs work well ← Lecture 9 (NLP 1)
- ➢ **Word representation:** How to represent the meaning of individual words
  - Old technology: One-hot representations, synsets ← Lecture 9 (NLP 1)
  - Embeddings: First trick that boosted the performance of NNs in NLP ← Lecture 9 (NLP 1)
    - Word2vec: Single layer NN. CBOW and skip-gram ← Lecture 10 (NLP 2)
    - Co-occurrence matrices: Basic counts and SVD improvement ← Lecture 10 (NLP 2)
    - Glove: Combining word2vec and co-occurrence matrices idea ← Lecture 10 (NLP 2)
    - Evaluating performance of embeddings ← Lecture 10 (NLP 2)
- ➢ **Named Entity Recognition (NER):** How to find words of specific meaning within text
  - Multilayer NNs: Margin loss. Forward- and back-propagation ← Lecture 11 (NLP 3)
  - Better loss functions: margin loss, regularisation ← Lecture 11 (NLP 3)
  - Better initializations: uniform, xavier ← Lecture 11 (NLP 3)
  - Better optimizers: Adagrad, RMSprop, Adam... ← Lecture 11 (NLP 3)

Course structure

- ➢ **Language modelling:** How to represent the meaning of full pieces of text
  - Old technology: N-grams ← Lecture 12 (NLP 4)
  - Recursive NNs language models (RNNs) ← Lecture 12 (NLP 4)
  - Evaluating performance of language models ← Lecture 12 (NLP 4)
  - Vanishing gradients: Problem. Gradient clipping ← Lecture 13 (NLP 5)
  - Improved RNNs: LSTM, GRU, Bidirectional... ← Lecture 13 (NLP 5)
- ➢ **Machine translation:** How to translate text
  - Old technology: Georgetown–IBM experiment and ALPAC report ← Lecture 16 (NLP 6)
  - Seq2seq: Greedy decoding, encoder-decoder, beam search ← Lecture 16 (NLP 6)
  - Attention: Simple attention, transformers, reformers ← Lecture 16 (NLP 6)
  - Evaluating performance: BLEU ← Lecture 16 (NLP 6)

Introduction: What is NLP?

- **Definition:** Natural Language Processing (NLP) is a field of computer science and linguistics, whose objective is to automatically process natural language as produced by humans.
- **Challenges to solve:** Text classification, machine translation, language modelling, question answering, conference resolution...
- **Applications:** Digital assistants (Siri, Hey Google, Alexa…), automatic translators (Google translate), search engines (Google, Bing…), advertising matching (Amazon, eBay...), speech recognition (annoying automatic call centres), automatic business analysis...

Introduction: Why NLP is hard?

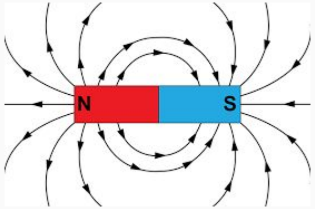
As many other AI tasks, the problem that NLP tries to solve appears simple on first sign, but turns out to be very hard upon careful consideration:

- ➢ Language is often ambiguous
- ➢ Grama/syntax is complex and often ignored
- ➢ Humans use very large amounts of prior knowledge about how the world works, context

Tesla explosions concern mr musk

What Tesla? Do people explode? Do magnetic field explode? What is mr (lower case)? Isn't musk an animal?

Think how someone from the 90s would have interpreted that sentence



Introduction: Why Neural Networks?

- ➢ Manually designed features are often over-specified, incomplete, and take a long time to design and validate.
- ➢ Learned features are easy to adapt. You only need new data and run your learning algorithm
- ➢ Deep learning provides a very flexible, (almost?) universal, learnable framework for representing world, visual, and linguistic information.
- ➢ Deep learning can learn unsupervised (from raw text) and
- ➢ supervised (with specific labels like positive/negative).

Introduction: Why Neural Networks?

➢ In about 2010, deep learning techniques started outperforming other machine learning techniques. Why this decade?

- Better **data**, or rather, much much more data is available
- Better **hardware**, such as GPUs, had drastically developed during 2000s
- Better **software**, such as new models, algorithms, and ideas:
    - better, more flexible learning of intermediate representations
    - effective end-to-end joint system learning
    - effective learning methods for using contexts and transferring between tasks
    - better regularization and optimization methods

⇒ improved performance (first in speech and vision, then NLP

Introduction: Why Neural Networks?

- ➢ Since 2010, successive improvements in **software** have further improved the performance of NNs in NLP:
  - Using LSTMs rather than simpler RNNs
  - Using embeddings (vectors in $\mathbb{R}^n$) to represent words
  - End to end learning, rather than a chain of individual computational modules
  - Language models to capture meaning of text
  - Attention
  - Transformer layers
  - Better and easier to use libraries (pytorch, tensorflow…), with standardized datasets
- ➢ Since 2010, **industry** is interested, and industry has deep pockets… very deep pockets

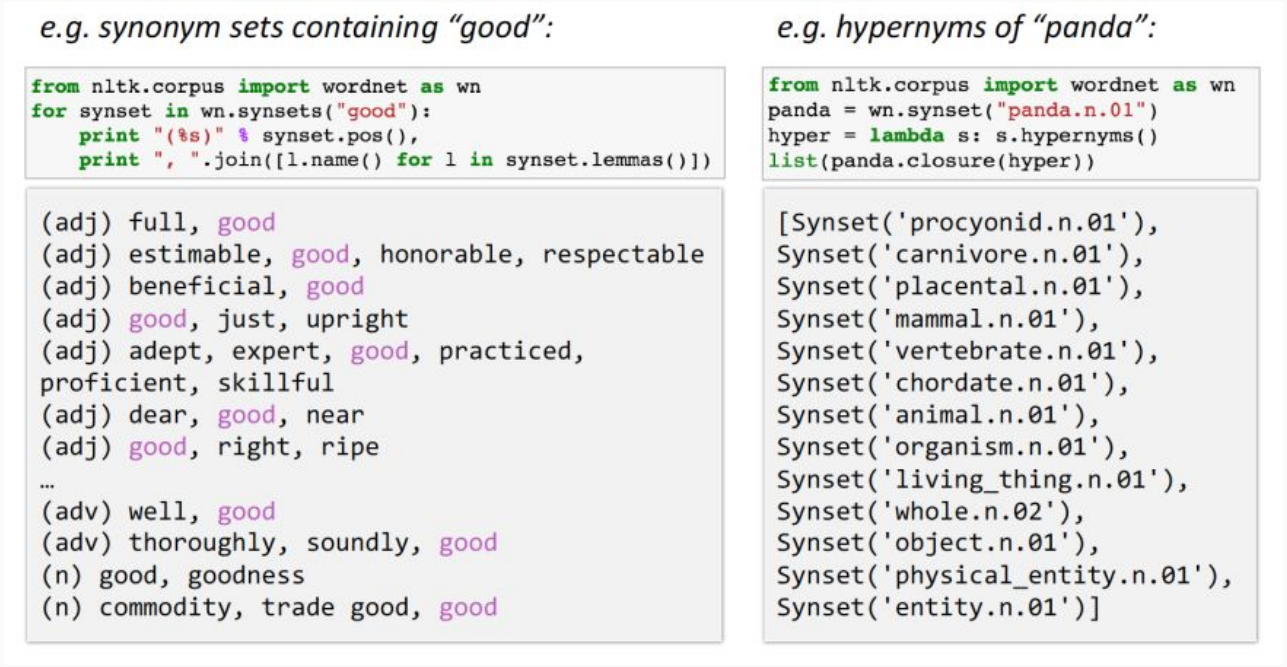
  $\Rightarrow$ further improved performance

Representing words: One-hot coding

- In traditional NLP, we represent words as discrete symbols: hotel, conference, motel.
- A common method to implement this was with 1-hot representations:
  - Each word is a vector in $\mathbb{R}^n$ space.
  - In that vector, all dimensions are 0, except one per word:
    - Hotel = [0, 0, 0, 1, 0, 0, 0, 0, … 0, 0]
    - Conference = [0, 0, 0, 0, 1, 0, 0, 0, … 0, 0]
    - Motel = [0, 0, 0, 0, 0, 1, 0, 0, … 0, 0]
  - The vectors have as many dimensions as words in the vocabulary (>100K)
- These had a number of problems:
  - Too many dimensions
  - No representation of word similarity (e.g. all vectors are orthogonal)

Representing words: Synsets

➢ An attempt to ameliorate the problems of 1-hot representations: WordNet and Synsets were manually curated to group words into groups of synonyms



e.g. synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
for synset in wn.synsets("good"):
    print "(%s)" % synset.pos(),
    print ", ".join([l.name() for l in synset.lemmas()])
```

```
(adj) full, good
(adj) estimable, good, honorable, respectable
(adj) beneficial, good
(adj) good, just, upright
(adj) adept, expert, good, practiced,
proficient, skillful
(adj) dear, good, near
(adj) good, right, ripe
…
(adv) well, good
(adv) thoroughly, soundly, good
(n) good, goodness
(n) commodity, trade good, good
```

e.g. hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

Representing words: Embeddings

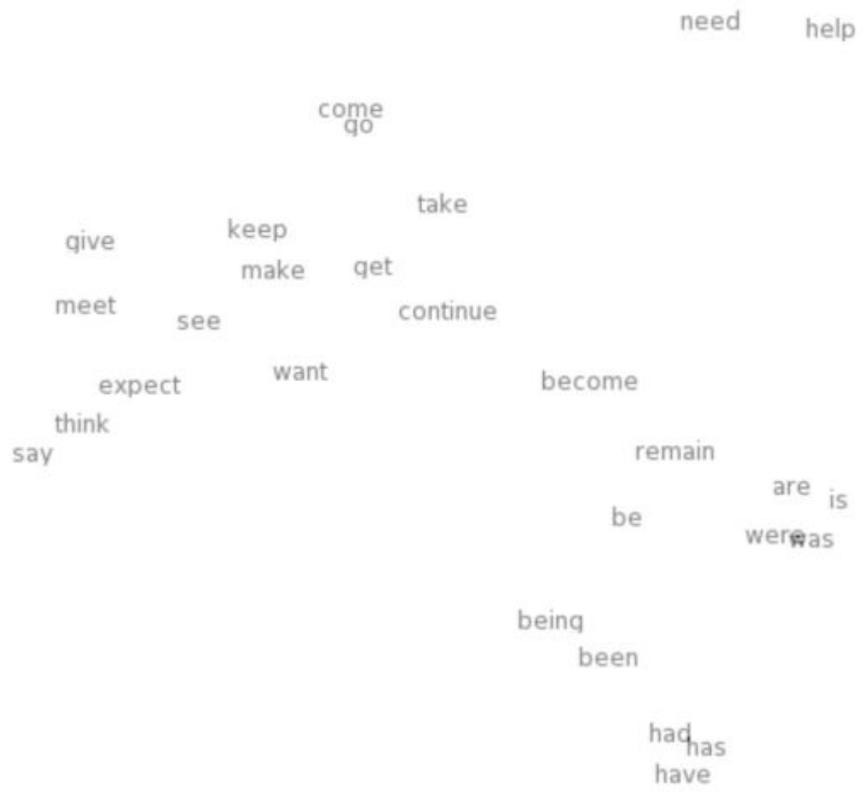
- **Idea:** Linguists say that the meaning of words come from the contexts where they are used. We could use a very simple NN to build word representations from this ‘context’ phenomenon. This was a very significant improvement used by NNs
  - … after a few days the fur of the dog was unkept and dirty, and this spread ...
  - … you will soon realise that walks in the park are dog’s priority, and they will ...
  - … he was worried that his neighbour’s dog kept barking during all night ...
  - … the warden had a labrador of brown fur who kept chasing squirrels in ...
  - … breeders recommend to daily take your labrador to the park to tempter ....
  - … at the end of the day, a labrador barks less than other breeds, but he also …
- Successfully implementing this idea was the first breakthrough of the new wave of NN’s NLP … and it only “happened” in 2013 (Mikolov and word2vec)

Representing words: Embeddings

- Word2vec very successfully implemented word embeddings using this context-meaning idea.
  - We start with a very large corpus of text (e.g. all of Wikipedia)
  - Every word is represented by a vector in $\mathbb{R}^n$ space (n~200 dimensions)
  - You have a model (e.g. a NN) that tries to predict the vector of a word (i.e. the central word) given the vectors of the words around it (i.e. its context). In probability terms, the NN models the probability $P(w_c \mid w_{c-3}, w_{c-2}, w_{c-1}, w_{c+1}, w_{c+2}, w_{c+3})$
  - Go through each central word - context pair in the corpus
  - In each iteration, modify the NN and vectors a little bit for words with similar contexts to have similar vectors
  - Repeat last 2 steps many times

Representing words: Embeddings

$$\textit{expect} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$



Representing words: Embeddings

- Nearest words to frog:
  1. trogs
  2. toad
  3. litoria
  4. leptodactylidae
  5. rana ← that is spanish
  6. lizard
  7. eleutherodactylus

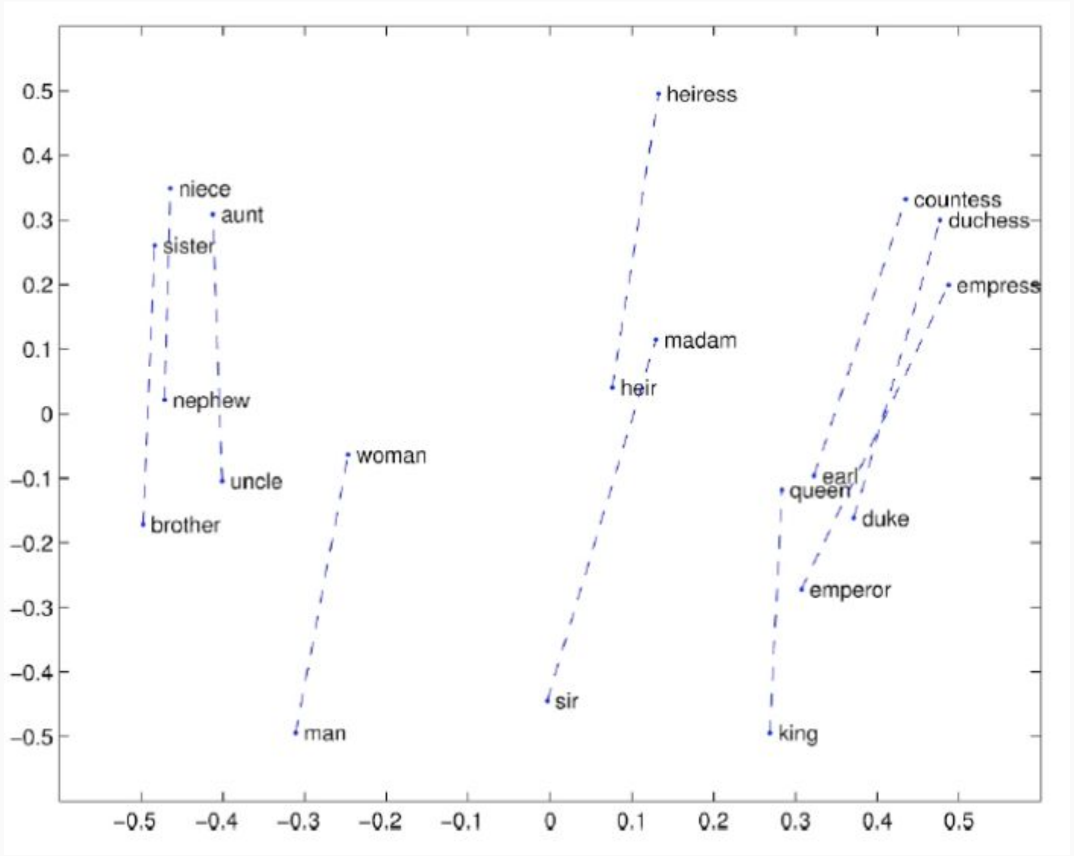
litoria

leptodactylidae

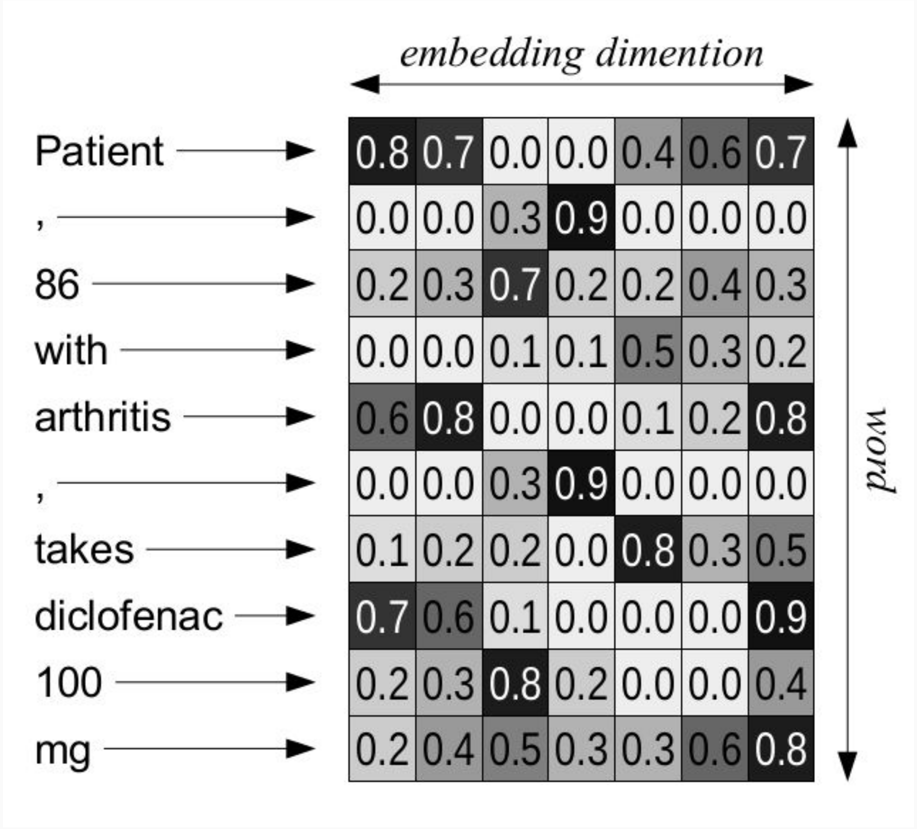
rana

eleutherodactylus

Representing words: Embeddings



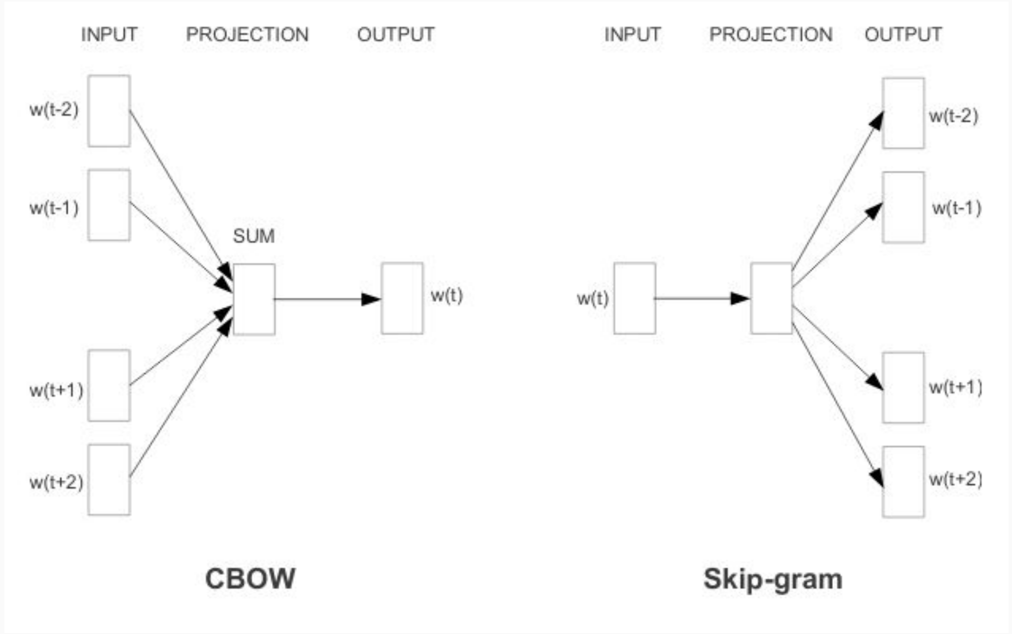
Representing words: Embeddings



Patient	0.8	0.7	0.0	0.0	0.4	0.6	0.7
,	0.0	0.0	0.3	0.9	0.0	0.0	0.0
86	0.2	0.3	0.7	0.2	0.2	0.4	0.3
with	0.0	0.0	0.1	0.1	0.5	0.3	0.2
arthritis	0.6	0.8	0.0	0.0	0.1	0.2	0.8
,	0.0	0.0	0.3	0.9	0.0	0.0	0.0
takes	0.1	0.2	0.2	0.0	0.8	0.3	0.5
diclofenac	0.7	0.6	0.1	0.0	0.0	0.0	0.9
100	0.2	0.3	0.8	0.2	0.0	0.0	0.4
mg	0.2	0.4	0.5	0.3	0.3	0.6	0.8

Representing words: Embeddings

➢ OK, awesome, all seems very nice… but how do you actually implement the model? How does word2vec represent P(w_c | w_{c-3}, w_{c-2}, w_{c-1}, w_{c+1}, w_{c+2}, w_{c+3}) in a NN?

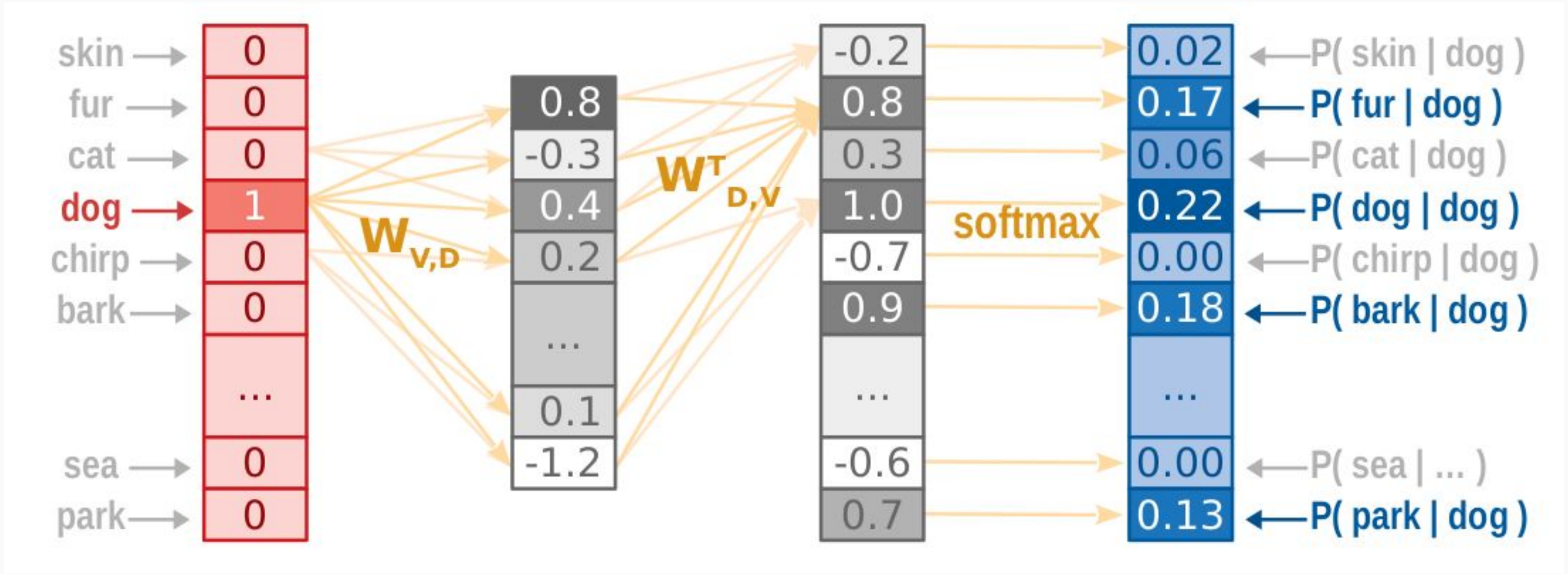


➢ You represent P(w_c | w_{c-3} …) with a very simple NN. The reason to use a very simple NN is that you have to train over an extremely large amount of data. This approach is called CBOW

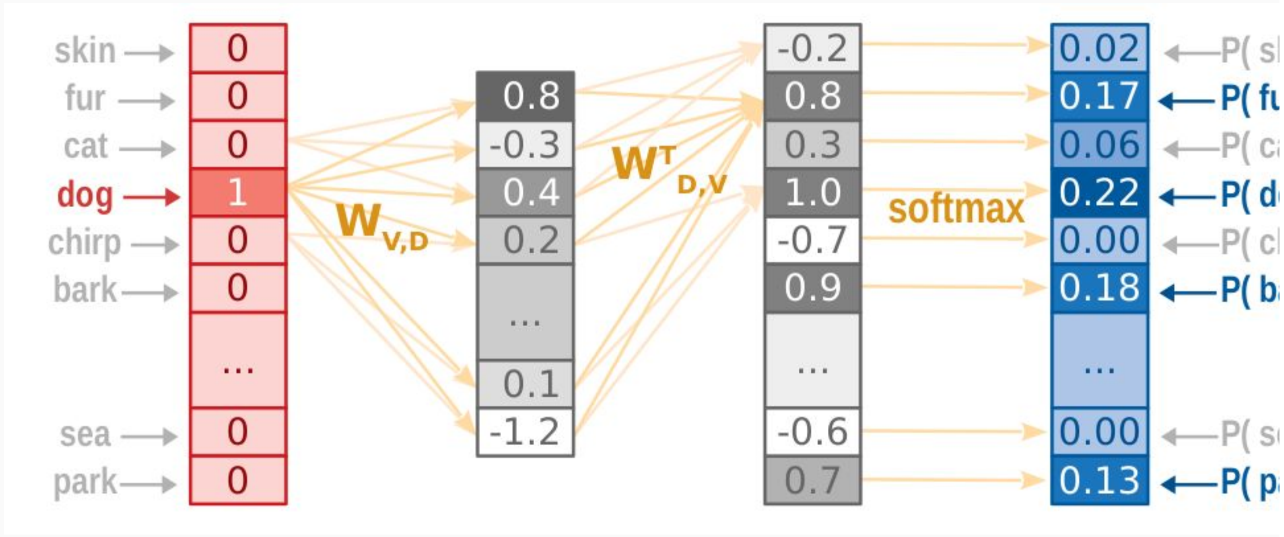
➢ Alternatively, you can rather model P(w_{c+i} | w_c). This approach is call skip-gram.

Representing words: Embeddings

- The NN uses a single hidden layer, a single weight matrix (W_{VD}), the transpose of this weight matrix (W^T_{VD}), and a single activation function (softmax)



Representing words: Embeddings



$$\underset{V}{softmax}\left([x]_V \times [W]_{VD} \times [W^T]_{DV}\right) = [y]_V$$

$$softmax(x) = \frac{\exp(x)}{\sum \exp(x)}$$

$$\frac{\exp\left([x]_V \times [W]_{VD} \times [W^T]_{DV}\right)}{\sum_V \exp\left([x]_V \times [W]_{VD} \times [W^T]_{DV}\right)} = [y]_V$$

$$\frac{\exp(\vec{w}_y \times \vec{w}_x)}{\sum_i \exp(\vec{w}_i \times \vec{w}_x)} = y(v)$$

$$[0 \quad 0 \quad 0 \quad \boxed{1} \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ \boxed{10} & \boxed{12} & \boxed{19} \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

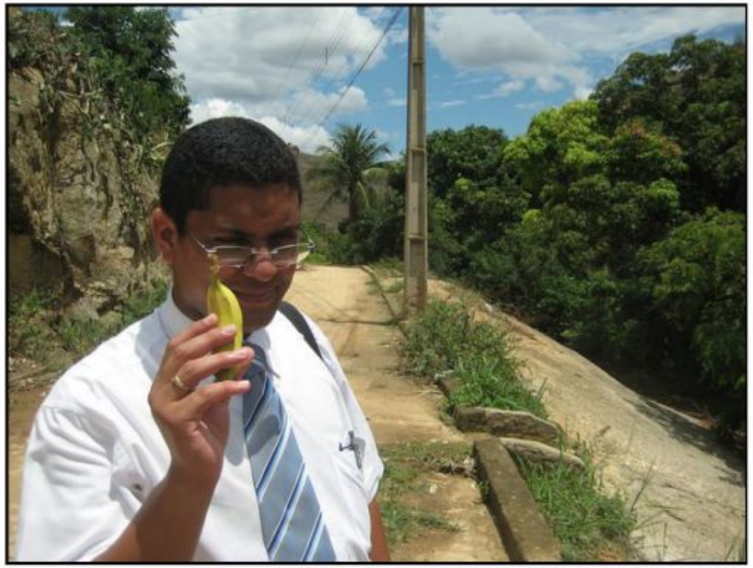
Industry: Question answering

CNN article:

Document	The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the "Top Gear" host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon "to an unprovoked physical and verbal attack." . . .
Query	Who does the article say will not press charges against Jeremy Clarkson?
Answer	Oisin Tymon

Industry: Question answering

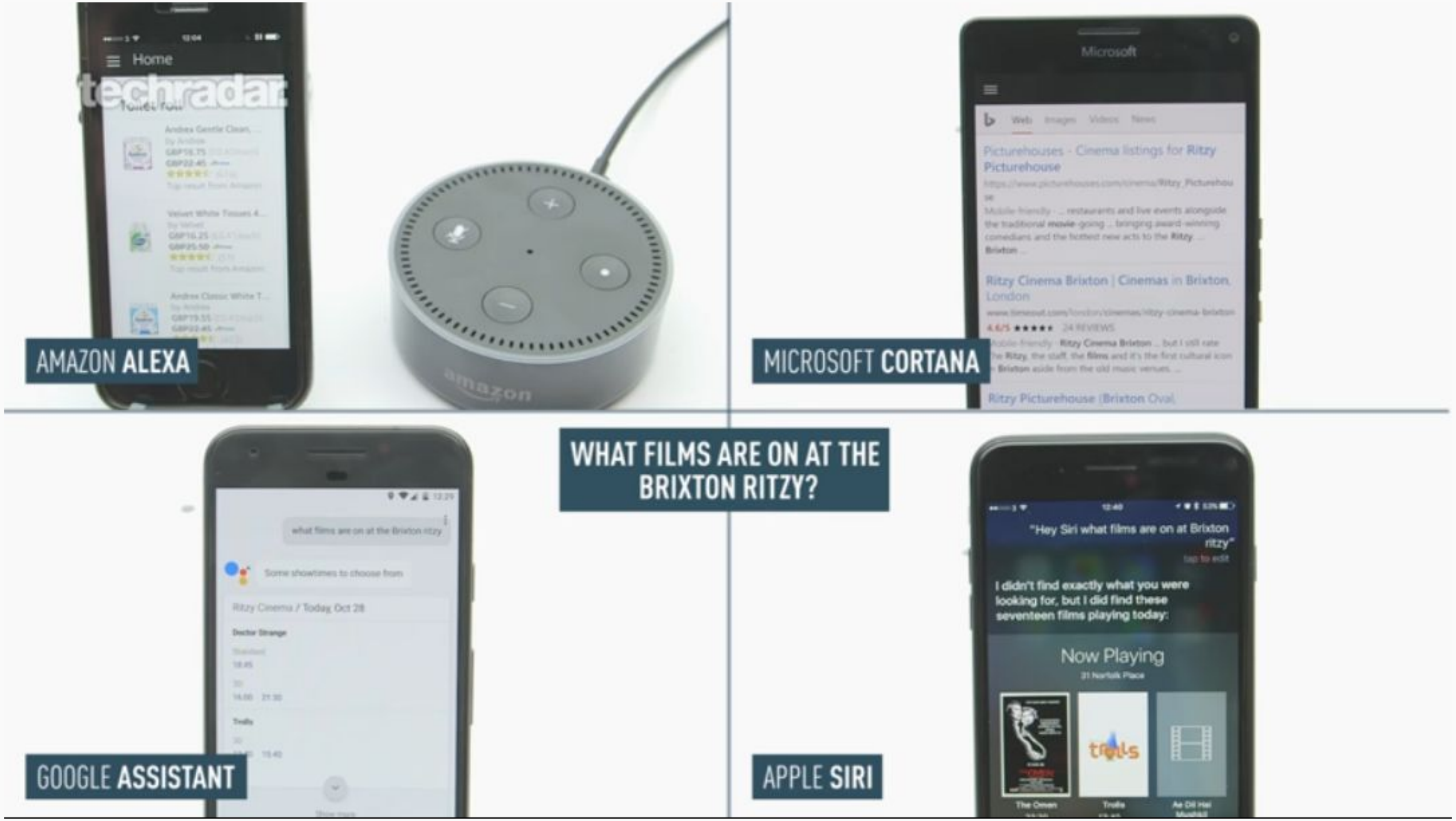
- ➢ NLP tasks are often also combined with other modalities, such as image recognition
- ➢ Agrawal et al., VQA: Visual Question Answering, ICCV 2015.



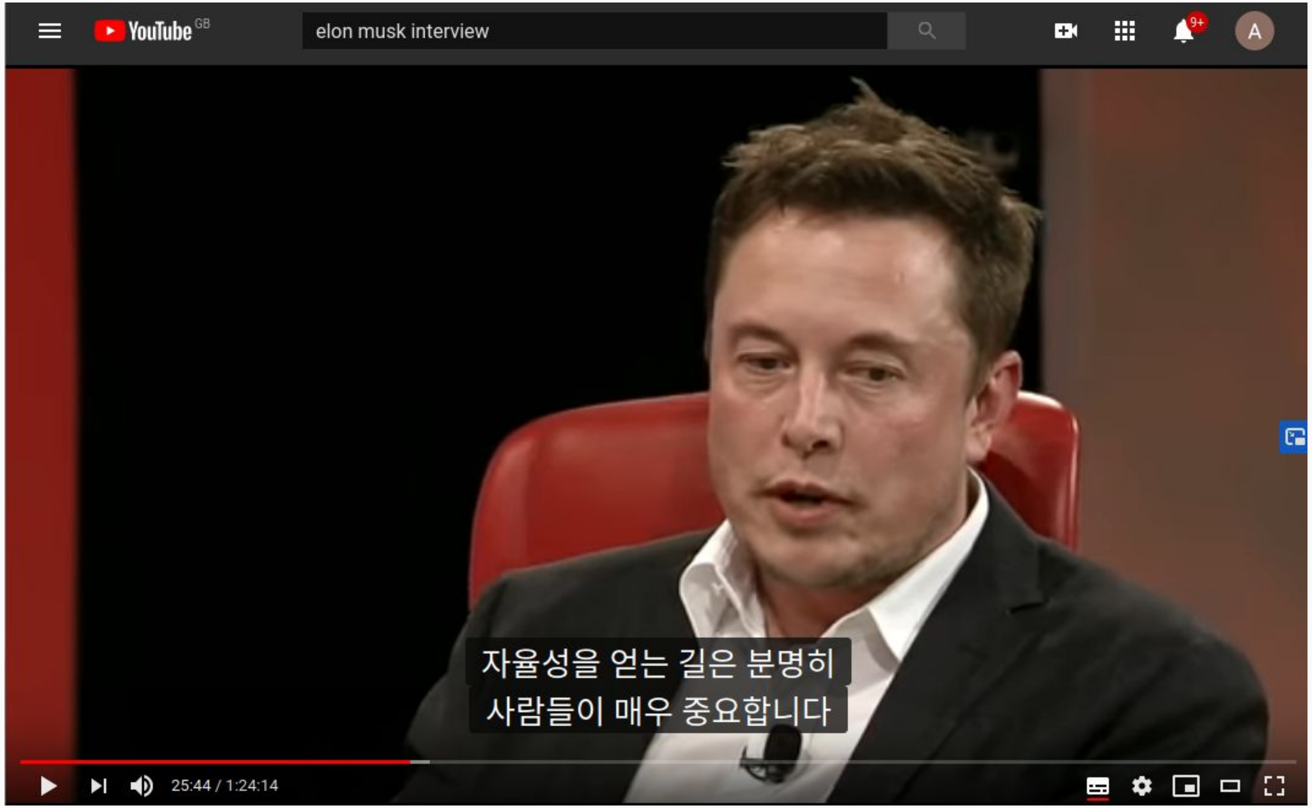
What is the man holding?
Does it appear to be raining?
Does this man have 20/20 vision?

Industry: Lip reading

Industry: Virtual assistants


AMAZON ALEXA
MICROSOFT CORTANA
WHAT FILMS ARE ON AT THE BRIXTON RITZY?
GOOGLE ASSISTANT
APPLE SIRI
techradar
"Hey Siri what films are on at Brixton ritzy"
I didn't find exactly what you were looking for, but I did find these seventeen films playing today:
Now Playing
what films are on at the Brixton ritzy
Some showtimes to choose from
Ritzy Cinema / Today, Oct 28

Industry: Live translation



자율성을 얻는 길은 분명히
사람들이 매우 중요합니다

Course structure

- ➢ **Introduction:** What is NLP. Why it is hard. Why NNs work well ← Lecture 9 (NLP 1)
- ➢ **Word representation:** How to represent the meaning of individual words
  - Old technology: One-hot representations, synsets ← Lecture 9 (NLP 1)
  - Embeddings: First trick that boosted the performance of NNs in NLP ← Lecture 9 (NLP 1)
    - Word2vec: Single layer NN. CBOW and skip-gram ← Lecture 10 (NLP 2)
    - Co-occurrence matrices: Basic counts and SVD improvement ← Lecture 10 (NLP 2)
    - Glove: Combining word2vec and co-occurrence matrices idea ← Lecture 10 (NLP 2)
    - Evaluating performance of embeddings ← Lecture 10 (NLP 2)
- ➢ **Named Entity Recognition (NER):** How to find words of specific meaning within text
  - Multilayer NNs: Margin loss. Forward- and back-propagation ← Lecture 11 (NLP 3)
  - Better loss functions: margin loss, regularisation ← Lecture 11 (NLP 3)
  - Better initializations: uniform, xavier ← Lecture 11 (NLP 3)
  - Better optimizers: Adagrad, RMSprop, Adam... ← Lecture 11 (NLP 3)

Literature

- ➢ Papers =
  - "Efficient estimation of word representations in vector space", Mikolov et al., 2013. http://arxiv.org/pdf/1301.3781
  - "Distributed representations of words and phrases and their compositionality", Mikolov et al., 2013. https://arxiv.org/abs/1310.4546
- ➢ Nice tutorials
  - "Word2Vec tutorial - The skip-gram model", McCormick, 2016. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
  - How to do it in PyTorch = https://pytorch.org/tutorials/beginner/nlp/word_embeddings_tutorial.html
  - How to do it in Tensorflow = https://www.tensorflow.org/tutorials/text/word_embeddings