Chapter 5 Natural Language Processing

COMP 6721 Introduction of AI

Russell & Norvig – Chapter 23.1 + 23.2 + 23.3

Topics

- Word Embeddings
- N-gram Models
- Word Embeddings (continued)
- Contextualized Word Embeddings

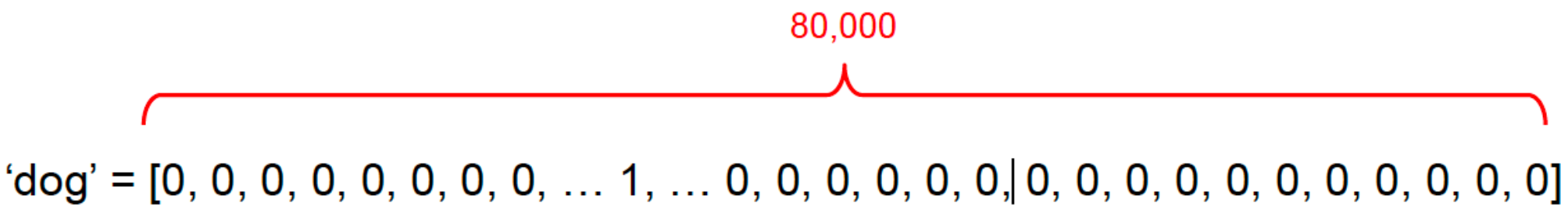
Beyond One Hot Vectors

- One hot vectors are local representations.
- They represent a 1:1 mapping between a word (e.g., ‘cat’, ‘dog’, ‘house’) and a position in a vector (i.e., the position with the value of 1).

Hinton, McClelland, Rumelhart, “Parallel distributed processing: explorations in the microstructure of cognition, vol. 1”

Beyond One Hot Vectors

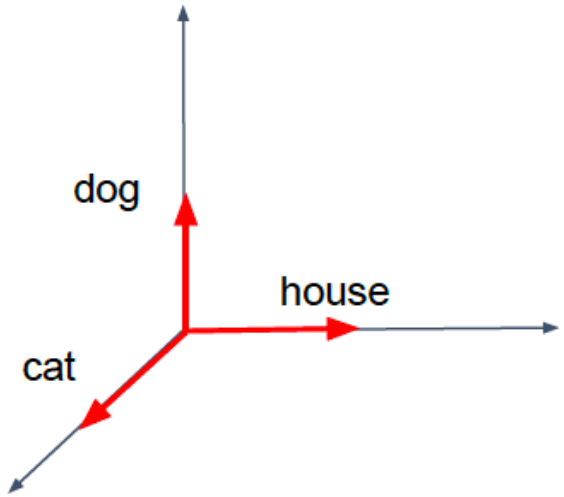
- This representation has two main disadvantages:
  - The vector dimension is proportional to the vocabulary size.
  - It does not capture relationships between words.
- E.g., with a vocabulary containing 80,000 words:



Beyond One Hot Vectors

- This representation has two main disadvantages:
  - The vector dimension is proportional to the vocabulary size.
  - It does not capture relationships between words.
    - All words are at the same distance.

‘cat’	=	[0, 0, 1]
‘dog’	=	[0, 1, 0]
‘house’	=	[1, 0, 0]



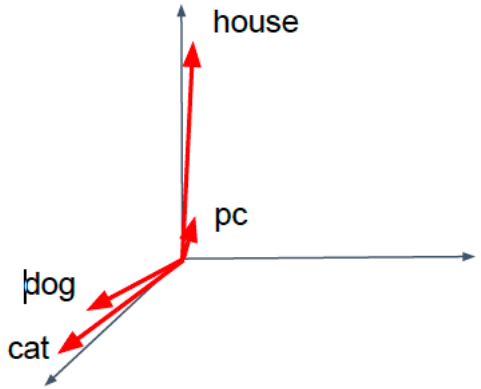
Distributed Representation

- A different approach is to distribute the representation of a word across all available dimensions in a continuous space in a way that leads to “similar” words being close to each other.
- This is called a distributed representation.

Distributed Representation

- Let's assume that the space dimensionality is 3 and that we have the following distributed representations:

'cat'	=	[1.1, 0.46, 45]
'dog'	=	[1.1, 1.10, 30]
'house'	=	[0.2, 4.51, 0]
'pc'	=	[0.1, 0.3 , 0]

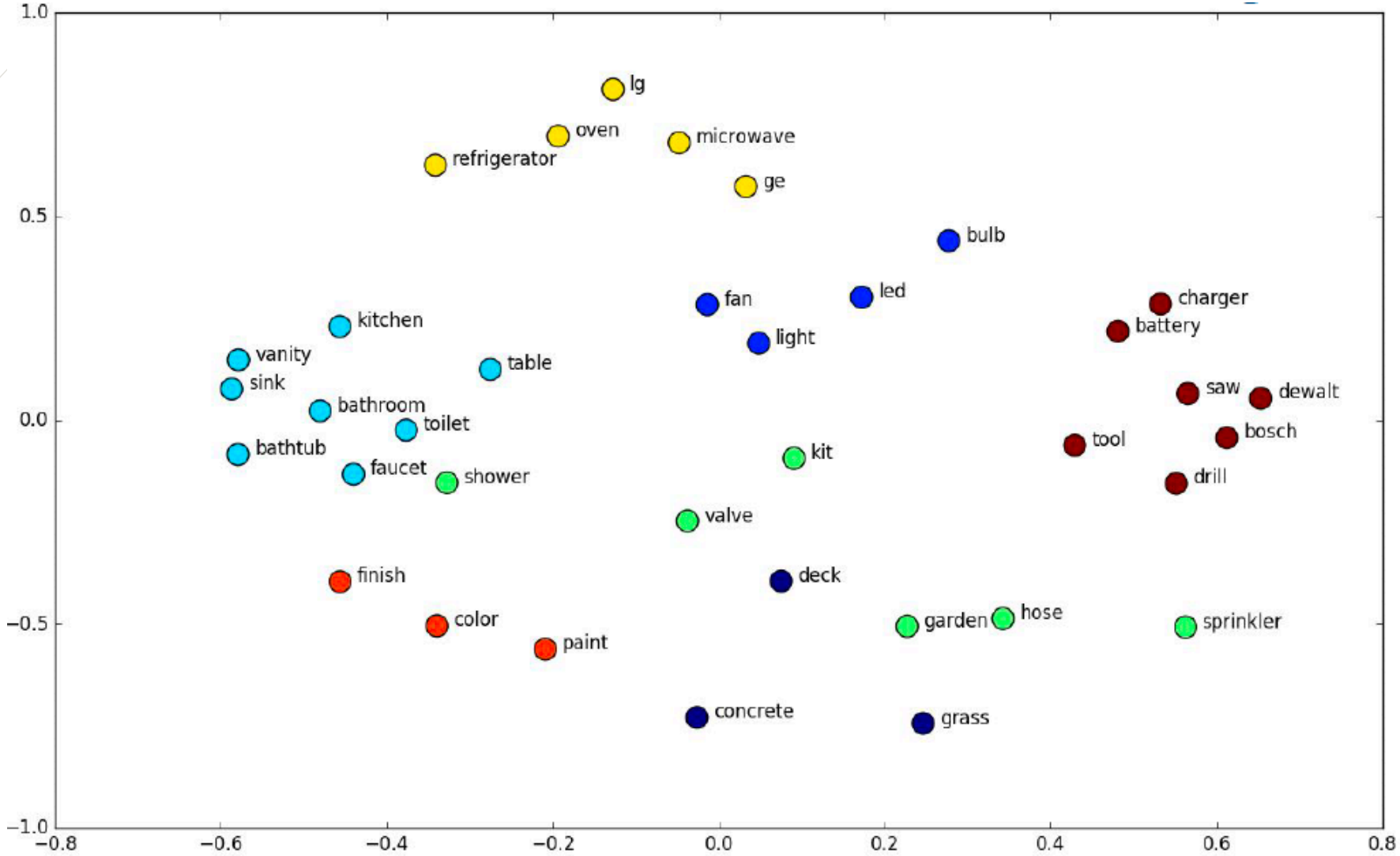


- Note how the vector size is not equal to the vocabulary size.
  - The vector size will in general be much smaller than the vocabulary size.
- Also note how 'dog' and 'cat' are more "similar" than 'dog' and 'house'.

Distributed Representation

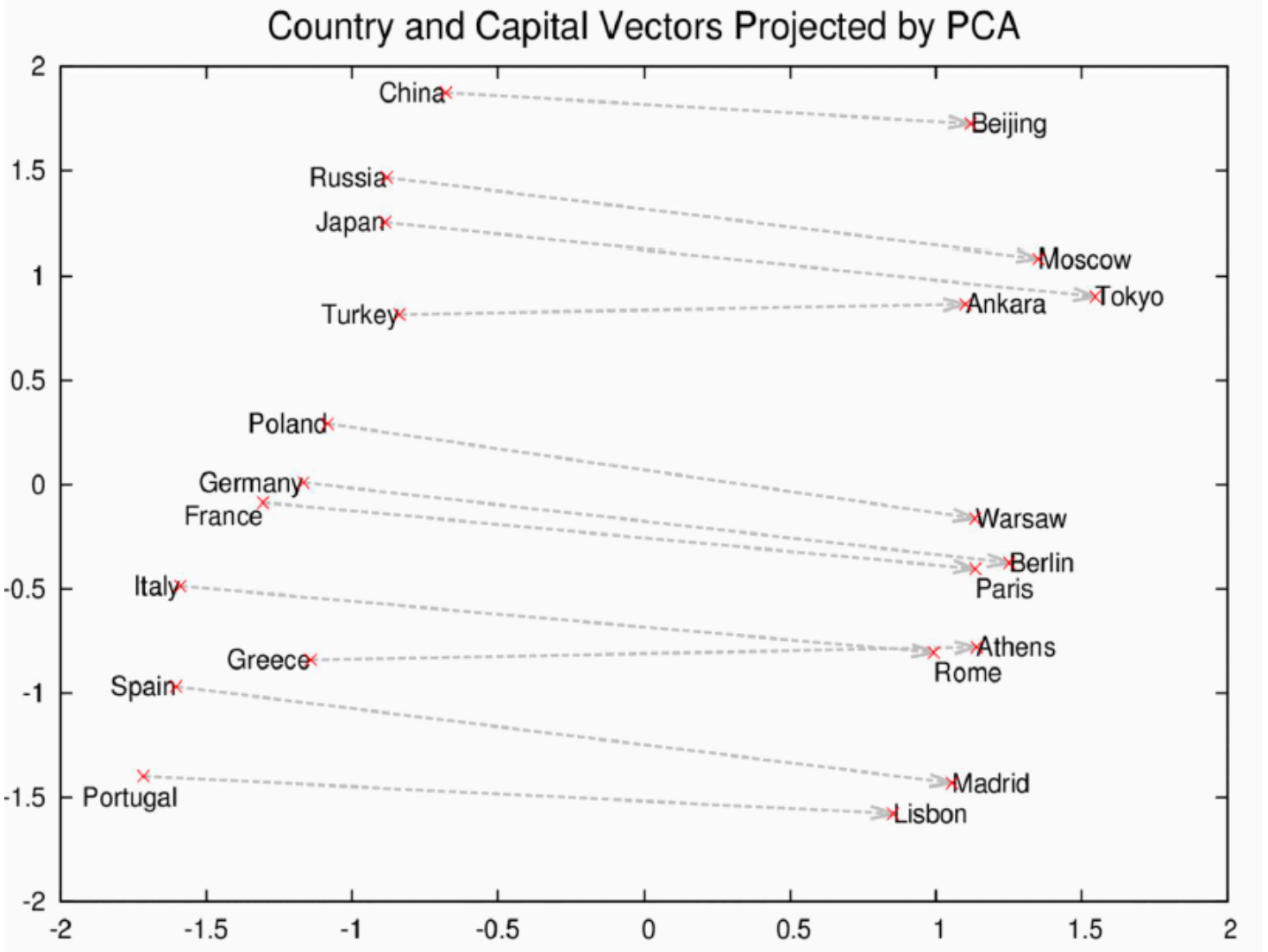
- Distributed representations can expose useful properties:
  - Semantically similar words can be close to each other.
  - Relationships between words can be captured.

Distributed Representation - Similarity



https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

Distributed Representation - Relations



Mikolov et al., “Distributed Representations of Words and Phrases and their Compositionality”

Word Embeddings

- Distributed representations for words are called word embeddings.
- There are several ways to generate word embeddings that expose the desired properties that we mentioned, including:
  - Word2Vec
  - FastText

Topics

- ***Word Embeddings***
- N-gram Models
- *Word Embeddings (continued)*
- *Contextualized Word Embeddings*

N-gram Model

- An n-gram model is a probability distribution over sequences of events (grams/units/items)
- models the order of the events
- Used when the past sequence of events is a good indicator of the next event to occur in the sequence
- i.e. To predict the next event in a sequence of event

N-gram Model

- E.g.
  - next move of player based on his/her past moves

    left right right up ... up? down? left? right?
  - next base pair based on past DNA sequence

    AGCTTCG ... A? G? C? T?
  - next word based on past words

    Hi dear, how are ... helicopter? laptop? you? magic?

What is a Language Model

- A Language model is a n-gram model over word/character sequences
- ie: events = words or events = character
- P("I'd like a coffee with 2 sugars and milk") $\approx$ 0.001
- P("I'd hike a toffee with 2 sugars and silk") $\approx$ 0.000000001

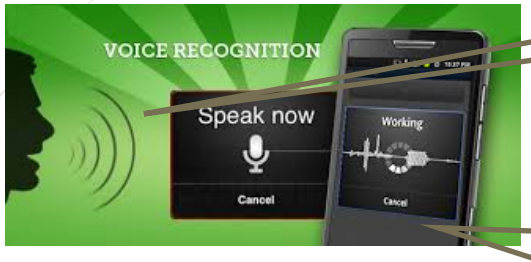
Applications of Language Model

- Speech Recognition
- Statistical Machine Translation
- Language Identification
- Spelling correction

  He is trying to fine out.

  He is trying to find out.
- Optical character recognition / Handwriting recognition
- …

In Speech Recognition



Given: Observed sound - O

Find: The most likely word/sentence – S*
S1: How to recognize speech. ?
S2: How to wreck a nice beach. ?
S3: …

- Goal: find most likely sentence (S*) given the observed sound (O) …
- i.e. pick the sentence with the highest probability: $S^* = \underset{S \in L}{\operatorname{argmax}} P(S \mid O)$
- We can use Bayes rule to rewrite this as: $S^* = \underset{S \in L}{\operatorname{argmax}} \frac{P(O \mid S)P(S)}{P(O)}$
- Since denominator is the same for each candidate S, we can ignore it for the argmax: $S^* = \underset{S \in L}{\operatorname{argmax}} P(O \mid S) \times P(S)$

Acoustic model --
Probability of the possible phonemes in the language +
Probability of ≠ pronunciations

Language model -- P(a sentence)
Probability of the candidate sentence in the language

In Speech Recognition

$$S^* = \underset{S \in L}{\operatorname{argmax}}\ P(O \mid S) \times P(S)$$

Acoustic model

Language model

$$\underset{\text{word sequence}}{\operatorname{argmax}}\ P(\text{word sequence} \mid \text{acoustic signal})$$

$$\underset{\text{wordsequence}}{\operatorname{argmax}}\ \frac{P(\text{acoustic signal} \mid \text{word sequence}) \times P(\text{word sequence})}{P(\text{acoustic signal})}$$

$$\underset{\text{wordsequence}}{\operatorname{argmax}}\ P(\text{acoustic signal} \mid \text{word sequence}) \times P(\text{word sequence})$$

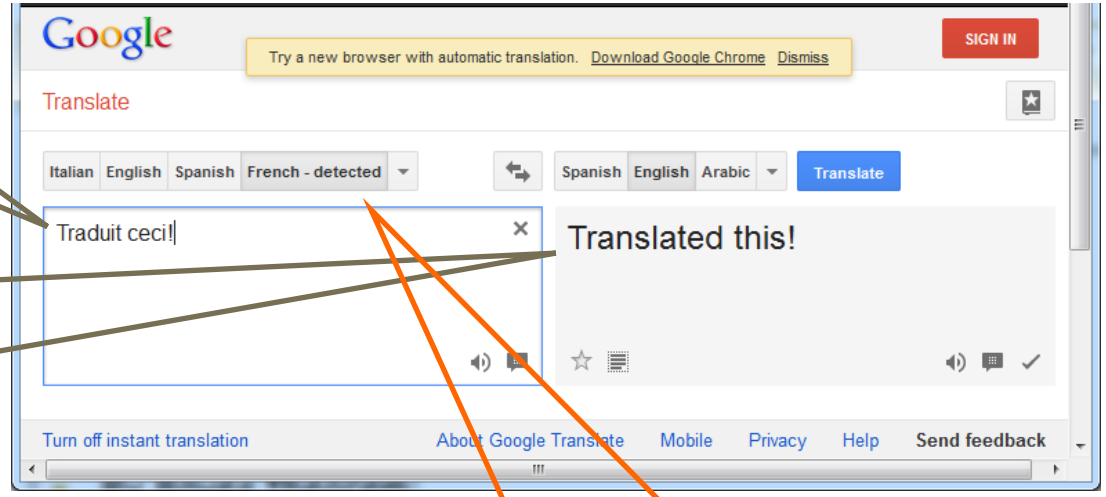
In statistical Machine Translation

- Assume we translate from fr[foreign] to English i.e: (en | fr)

Given: Foreign sentence - fr

Find: The most likely English sentence – en*

S1: Translate that!
S2: Translate this!
S3: Eat your soup!
S4…



Automatic Language Identification… guess how that's done?

$$en^* = \underset{en}{\operatorname{argmax}}\, P(fr \mid en) \times P(en)$$

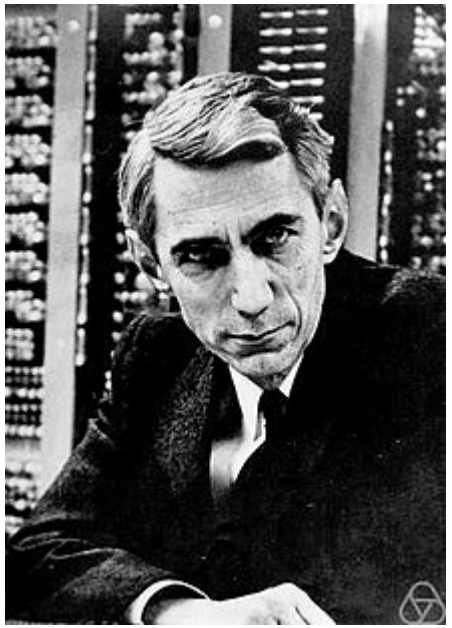
Translation model

Language model

“Shannon Game” (Shannon, 1951)

- Predict the next word/character given the *n-1* previous words/characters.

“I am going to make a collect …”



1st approximation

- each word has an equal probability to follow any other
  - with 100,000 words, the probability of each word at any given point is .00001

- but some words are more frequent then others…
  - "*the*" appears many more times, than "*rabbit*"

2nd approximation: unigrams

- take into account the frequency of the word in some training corpus
  - at any given point, "*the*" is more probable than "*rabbit*"
- but does not take word order into account. This is the bag of word approach.
  - *"Just then, the white …"*
- so the probability of a word also depends on the previous words (the history)

$$P(w_n | w_1 w_2 \ldots w_{n-1})$$

n-grams

- *"the large green _____ ."*
  - *"mountain"? "tree"?*
- *"Sue swallowed the large green _____ ."*
  - *"pill"? "broccoli"?*

- Knowing that Sue "*swallowed*" helps narrow down possibilities
- ie. Going back 3 words before helps
- But, how far back do we look?

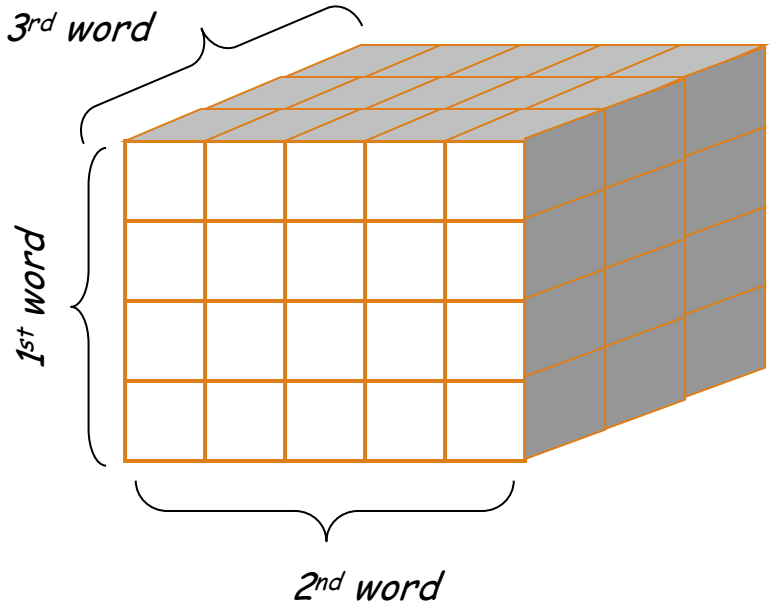
Bigrams

- first-order Markov models $P(w_n|w_{n-1})$
- N-by-N matrix of probabilities/frequencies
- N = size of the vocabulary we are using

1st word \ 2nd word	a	aardvark	aardwolf	aback	…	zoophyte	zucchini
a	0	0	0	0	…	8	5
aardvark	0	0	0	0	…	0	0
aardwolf	0	0	0	0	…	0	0
aback	26	1	6	0	…	12	2
…	…	…	…	…	…	…	…
zoophyte	0	0	0	1	…	0	0
zucchini	0	0	0	3	…	0	0

Trigrams

- second-order Markov models $P(w_n|w_{n-1}w_{n-2})$
- N-by-N-by-N matrix of probabilities/frequencies
- N = size of the vocabulary we are using



Why use only bi- or tri-grams

- Markov approximation is still costly
  with a 20 000 word vocabulary:
  - bigram needs to store 400 million parameters
  - trigram needs to store 8 trillion parameters
  - using a language model > trigram is impractical

Building N-gram Models

1. Data preparation:
   - Decide on training corpus
   - Clean and tokenize
   - How do we deal with sentence boundaries?
     - I eat. I sleep.
       - (I eat) (eat I) (I sleep)
     - <s>I eat <s> I sleep <s>
       - (<s> I) (I eat) (eat <s>) (<s> I) (I sleep) (sleep <s>)

Building N-gram Models

2. Count words and build model
   - Let $C(w_1 \ldots w_n)$ be the frequency of n-gram $w_1 \ldots w_n$

$$P(w_n \mid w_1 \ldots w_{n-1}) = \frac{C(w_1 \ldots w_n)}{C(w_1 \ldots w_{n-1})}$$

3. Smooth your model (see later)

Example 1:

- in a training corpus, we have 10 instances of "*come across*"
  - 8 times, followed by "*as*"
  - 1 time, followed by "*more*"
  - 1 time, followed by "*a*"
- so we have:
  - $$P(as \mid come\ across) = \frac{C(come\ across\ as)}{C(come\ across)} = \frac{8}{10}$$
  - P(more | come across) = 0.1
  - P(a | come across) = 0.1
  - P(X | come across) = 0 where X ≠ "*as*", "*more*", "*a*"

Example 2:

P(on|eat) =	.16	P(want|I) =	.32	P(eat|to) =	.26
P(some|eat) =	.06	P(would|I) =	.29	P(have|to) =	.14
P(British|eat) =	.001	P(don't|I) =	.08	P(spend|to)=	.09
…		…		…	
P(I|<s>) =	.25	P(to|want) =	.65	P(food|British) =	.6
P(I'd|<s>) =	.06	P(a|want) =	.5	P(restaurant|British) =	.15
…		…		…	

P(*I want to eat British food*)

= P(I | <s>) x P(want | I) x P(to | want) x P(eat | to) x P(British | eat) x P(food | British)

= .25 x .32 x .65 x .26 x .001 x .6

= .000008

Remember this slide…

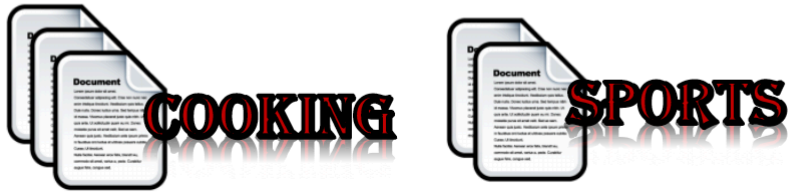
41

Using logs

- if we really do the product of probabilities…
  - $\text{argmax}_{c_j} P(c_j) \prod P(w_i \mid c_j)$
  - we soon have numerical underflow…
  - ex: 0.01 x 0.02 x 0.05 x …
- so instead, we add the log of the probs
  - $\text{argmax}_{c_j} \log(P(c_j)) + \sum \log(P(w_i \mid c))$
  - ex: log(0.01) + log(0.02) + log(0.05) + …

Remember this slide…

Example



- Dataset

c1: COOKING	c2: SPORTS
doc_1: … stove… kitchen… the… heat doc_2: … kitchen… pasta… stove… … doc_{100000}: … stove…heat… ball…	doc_1: … ball… heat… doc_2: … the… referee… player… … doc_{75000}: goal… injury …

- Assume:
  - |V| = 100 vocabulary = {ball, heat, kitchen, referee, stove, the, … }
  - 500,000 words in Cooking
  - 300,000 words in Sports
  - 100,000 docs in Cooking
  - 75,000 docs in Sports

Remember this slide…

Example

- Training – Unsmoothed / Smoothed probs:
  - P(ball | COOKING) = $\frac{10{,}000}{500{,}000}$ $\frac{??}{??}$ P(ball | SPORTS) = $\frac{10{,}000}{300{,}000}$ $\frac{??}{??}$
  - P(heat | COOKING) = $\frac{255}{500{,}000}$ $\frac{??}{??}$ P(heat | SPORTS) = $\frac{1{,}8000}{300{,}000}$ $\frac{??}{??}$
  - P(kitchen | COOKING) = $\frac{2{,}600}{500{,}000}$ $\frac{??}{??}$ P(kitchen | SPORTS) = $\frac{0}{300{,}000}$ $\frac{??}{??}$
  - P(referee | COOKING) = $\frac{0}{500{,}000}$ $\frac{??}{??}$ P(referee | SPORTS) = $\frac{1{,}500}{300{,}000}$ $\frac{??}{??}$
  - P(stove | COOKING) = $\frac{3{,}600}{500{,}000}$ $\frac{??}{??}$ P(stove | SPORTS) = $\frac{4}{300{,}000}$ $\frac{??}{??}$
  - P(the | COOKING) = $\frac{400{,}000}{500{,}000}$ $\frac{??}{??}$ P(the | SPORTS) = $\frac{19{,}000}{300{,}000}$ $\frac{??}{??}$
  - …
  - P(COOKING) = $\frac{100{,}000}{175{,}000}$ P(SPORTS) = $\frac{75{,}000}{175{,}000}$

Remember this slide…

Example

- Training – Unsmoothed / Smoothed probs:
  - P(COOKING) = $\frac{100,000}{175,000}$ P(SPORTS) = $\frac{75,000}{175,000}$
- Testing: "the referee hit the ~~blue~~ ~~bird~~"
  - Score(COOKING)=log($\frac{100,000}{175,000}$) + log(P(the | COOKING)) + log(P(referee | COOKING)) + log(P(hit | COOKING)) + log(P(the | COOKING))
  - Score(SPORTS)=log($\frac{75,000}{175,000}$) + log(P(the | SPORTS)) + log(P(referee | SPORTS)) + log(P(hit | SPORTS)) + log(P(the | SPORTS))

Some Adjustments

- product of probabilities… numerical underflow for long sentences
- so instead of multiplying the probs, we add the log of the probs

P(*I want to eat British food)*

= log(P(I | <s>)) + log(P(want | I)) + log(P(to | want)) + log(P(eat | to)) +
log(P(British | eat)) + log(P(food | British))

= log(.25) + log(.32) + log(.65) + log (.26) + log(.001) + log(.6)

Problem: Data Sparseness

- What if a sequence never appears in training corpus? P(X)=0
  - "*come across the men*" --> *prob* = 0
  - "*come across some men*" --> *prob* = 0
  - "*come across 3 men*" --> *prob* = 0
- The model assigns a probability of zero to unseen events …
- probability of an n-gram involving unseen words will be zero!
- Solution: smoothing
  - decrease the probability of previously seen events
  - so that there is a little bit of probability mass left over for previously unseen events

Remember this slide …

40

Smooth Probabilities

- normally: $P(w_i \mid c_j) = \frac{(\text{frequency of } w_i \text{ in } c_j)}{\text{total number of words in } c_j}$
- what if we have a $P(w_i \mid c_j) = 0$…?
  - ex. the word "fake" never appeared in the class SPAM?
  - then P("fake" | SPAM) = 0
- so if a text contains the word "fake", the class SPAM is completely ruled out !
- to solve this: we assume that every word always appears at least once (or a smaller value)
  - ex: add-1 smoothing:

$$P(w_i \mid c_j) = \frac{(\text{frequency of } w_i \text{ in } c_j) + 1}{\text{total number of words in } c_j + \text{size of vocabulary}}$$

Add-one Smoothing

- Pretend we have seen every n-gram at least once
- Intuitively:
  - new_count(n-gram) = old_count(n-gram) + 1
- The idea is to give a little bit of the probability space to unseen events

Add-one: Example

unsmoothed bigram counts (frequencies):

2nd word

1st word	I	want	to	eat	Chinese	food	lunch	…	Total
I	8	1087	0	13	0	0	0		C(I)=3437
want	3	0	786	0	6	8	6		C(want)=1215
to	3	0	10	860	3	0	12		C(to)=3256
eat	0	0	2	0	19	2	52		C(eat)=938
Chinese	2	0	0	0	0	120	1		C(Chinese)=213
food	19	0	17	0	0	0	0		C(food)=1506
lunch	4	0	0	0	0	1	0		C(lunch)=459
…									…
									…
									N=10,000

- Assume a vocabulary of 1616 (different) words
  - V = {*a, aardvark, aardwolf, aback, … , I, …, want,… to, …, eat, Chinese, …, food, …, lunch, …, zoophyte, zucchini*}
  - |V| = 1616 words
- And a total of N = 10,000 bigrams (~word instances) in the training corpus

Add-one: Example

unsmoothed bigram counts:

2nd word

1st word	I	want	to	eat	Chinese	food	lunch	…	Total
I	8	1087	0	13	0	0	0		C(I)=3437
want	3	0	786	0	6	8	6		C(want)=1215
to	3	0	10	860	3	0	12		C(to)=3256
eat	0	0	2	0	19	2	52		C(eat)=938
Chinese	2	0	0	0	0	120	1		C(Chinese)=213
food	19	0	17	0	0	0	0		C(food)=1506
lunch	4	0	0	0	0	1	0		C(lunch)=459
…									
									N=10,000

unsmoothed bigram conditional probabilities:

	I	want	to	eat	Chinese	food	lunch	…	Total
I	0.002 (8/3437)	0.316	0	0.0037	0	0	0		
want	0.0025	0	0.647 (786/1215)	0	0.0049 (6/1215)	0.0066	0.0049		
to	0.0009	0	0.0030	0.264	0.0009	0	0.0037		
eat	0	0	0.002	0	0.020	0.002	0.0554		
Chinese	0.0094	0	0	0	0	0.56	0.0047		
food	0.0126	0	0.0113	0	0	0	0		
lunch	0.0087	0	0	0	0	0.0002	0		
…									

note :

$$P(II) = \frac{8}{10\,000}$$

$$P(I \mid I) = \frac{8}{3\,437}$$

Add-one: Example

add-one smoothed bigram counts:

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	***Total***
I	~~8~~ 9	~~1087~~ 1088	1	14	1	1	1		~~3437~~ C(I) + |V| = 5053
want	~~3~~ 4	1	787	1	7	9	7		C(want) + |V| = 2831
to	4	1	**11**	**861**	**4**	**1**	**13**		C(to) + |V| = 4872
eat	1	1	23	1	20	3	53		C(eat) + |V| = 2554
Chinese	3	1	1	1	1	121	2		C(Chinese) + |V| = 1829
food	20	1	18	1	1	1	1		C(food) + |V| = 3122
lunch	5	1	1	1	1	2	1		C(lunch) + |V| = 2075
…									~~total = 10,000~~ N+|V|^2 = 10,000 + (1616)^2 = 2,621,456

add-one bigram conditional probabilities:

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…
I	.0018 (9/5053)	.215	.00019	.0028	.00019	.00019	.00019	
want	.0014	.00035	.278	.00035	.0025	.0031	.00247	
to	.00082	.0002	.00226	.1767	.00082	.0002	.00267	
eat	.00039	.00039	.0009	.00039	.0078	.0012	.0208	
…								

Add-one, more formally

$$P_{Add1}(w_1\, w_2 \ldots w_n) = \frac{C(w_1\, w_2 \ldots w_n) + 1}{N + B}$$

如上例中P（I I I）=（8+1）/（3437+1616）

3437 N: size of the corpus

i.e. number of n-gram tokens in training corpus

1616 B: number of "bins"

i.e. number of different n-gram types

i.e. number of cells in the matrix

e.g. for bigrams, it's (size of the vocabulary)^2

Add-delta Smoothing

- every previously unseen n-gram is given a low probability
- but there are so many of them that too much probability mass is given to unseen events
- instead of adding 1, add some other (smaller) positive value δ

$$P_{AddD}(w_1\, w_2 \ldots w_n) = \frac{C(w_1\, w_2 \ldots w_n) + \delta}{N + \delta B}$$

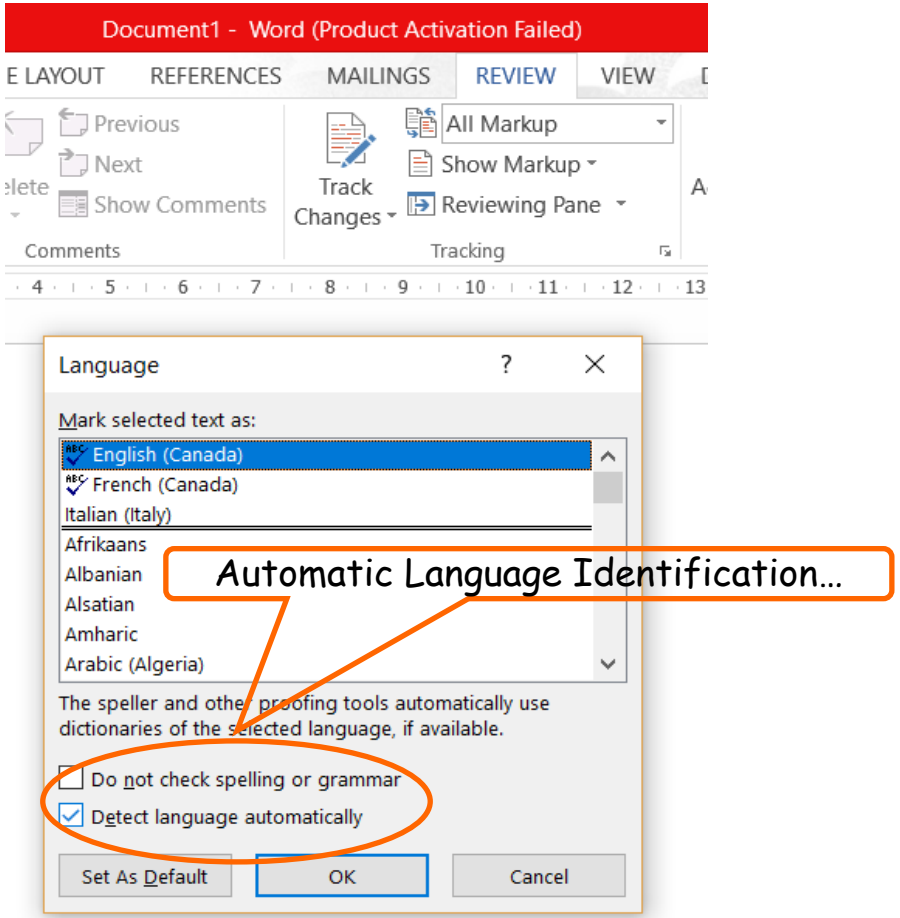
- most widely used value for $\delta = 0.5$
- better than add-one, but still…

Factors of Training Corpus

- Size:
  - the more, the better
  - but after a while, not much improvement…
    - bigrams (characters) after 100's million words
    - trigrams (characters) after some billions of words
- Genre (adaptation):
  - training on cooking recipes and testing on aircraft maintenance manuals

Example: Language Identification

- hypothesis: texts that resemble each other (same author, same language) share similar character/word sequences
  - In English character sequence "*ing*" is more probable than in French
- Training phase:
  - construction of the language model
  - with pre-classified documents (known language/author)
- Testing phase:
  - apply language model to unknown text



Example: Language Identification

- bigram of characters
  - characters = 26 letters (case insensitive)
  - possible variations: case sensitivity, punctuation, beginning/end of sentence marker, …

Example: Language Identification

1. Train a character-based language model for Italian:

	A	B	C	D	…	Y	Z
A	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
B	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
C	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
D	0.0042	0.0014	0.0014	0.0014	…	0.0014	0.0014
E	0.0097	0.0014	0.0014	0.0014	…	0.0014	0.0014
…	…	…	…	…	…	…	0.0014
Y	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
Z	0.0014	0.0014	0.0014	0.0014	0.0014	0.0014	0.0014

2. Train a character-based language model for Spanish:

	A	B	C	D	…	Y	Z
A	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
B	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
C	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
D	0.0042	0.0014	0.0014	0.0014	…	0.0014	0.0014
E	0.0097	0.0014	0.0014	0.0014	…	0.0014	0.0014
…	…	…	…	…	…	…	0.0014
Y	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
Z	0.0014	0.0014	0.0014	0.0014	0.0014	0.0014	0.0014

Example: Language Identification

3. Given a unknown sentence “che bella cosa” is it in Italian or in Spanish?

   P(“che bella cosa”) with the Italian LM

   P(“che bella cosa”) with the Spanish LM

4. Highest probability -->language of sentence

Google's Web 1T 5-gram Model

- 5-grams
- generated from 1 trillion words
- 24 GB compressed
  - Number of tokens: 1,024,908,267,229
  - Number of sentences: 95,119,665,584
  - Number of unigrams: 13,588,391
  - Number of bigrams: 314,843,401
  - Number of trigrams: 977,069,902
  - Number of fourgrams: 1,313,818,354
  - Number of fivegrams: 1,176,470,663
- See discussion: http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html
- See Google N-gram Viewer: http://en.wikipedia.org/wiki/Google_Ngram_Viewer

Problem with N-gram

- Natural language is not linear ....
- there may be *long-distance dependencies*.
  - Syntactic dependencies
    - *The man next to the large oak tree near … is tall.*
    - *The men next to the large oak tree near … are tall.*
  - *Semantic dependencies*
    - *The bird next to the large oak tree near … flies rapidly.*
    - *The man next to the large oak tree near … talks rapidly.*
  - *World knowledge*
    - *Michael Jackson, who was featured in ..., is buried in California.*
    - *Michael Bublé, who was featured in ..., is living in California.*
  - ...
- More complex models of language are needed to handle such dependencies.

Topics

- ***Word Embeddings***
- ***N-gram Models***
- Word Embeddings (continued)
- Contextualized Word Embeddings

Word2Vec

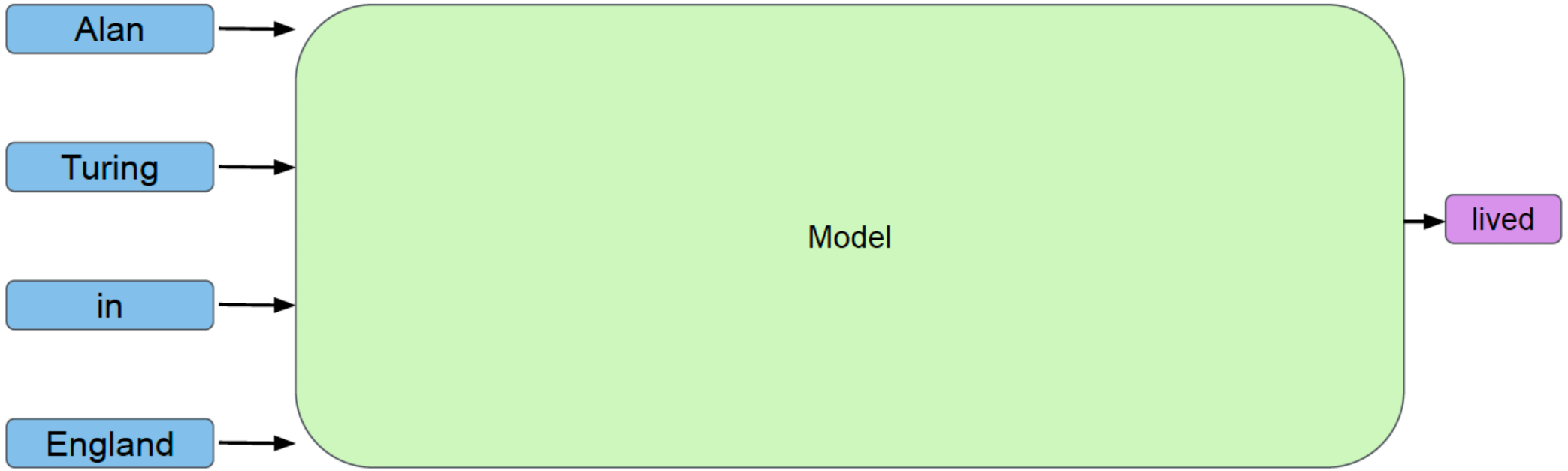
- Word2Vec is an efficient way to compute word embeddings.
- It is inspired from the distributional hypothesis.
- It is based on a very simple linear model.
- There are two versions:
  - Continuous bag-of-words.
  - Continuous Skip-Gram.

Word2Vec – Continuous bag-of-words

- The task is simple:
  - Mask a word in a sequence of text.
  - Train a model to predict this word given the words that surround it.
- For example, for "Alan Turing lived in England":
  - Input: "Alan Turing in England"
  - Target: "lived"

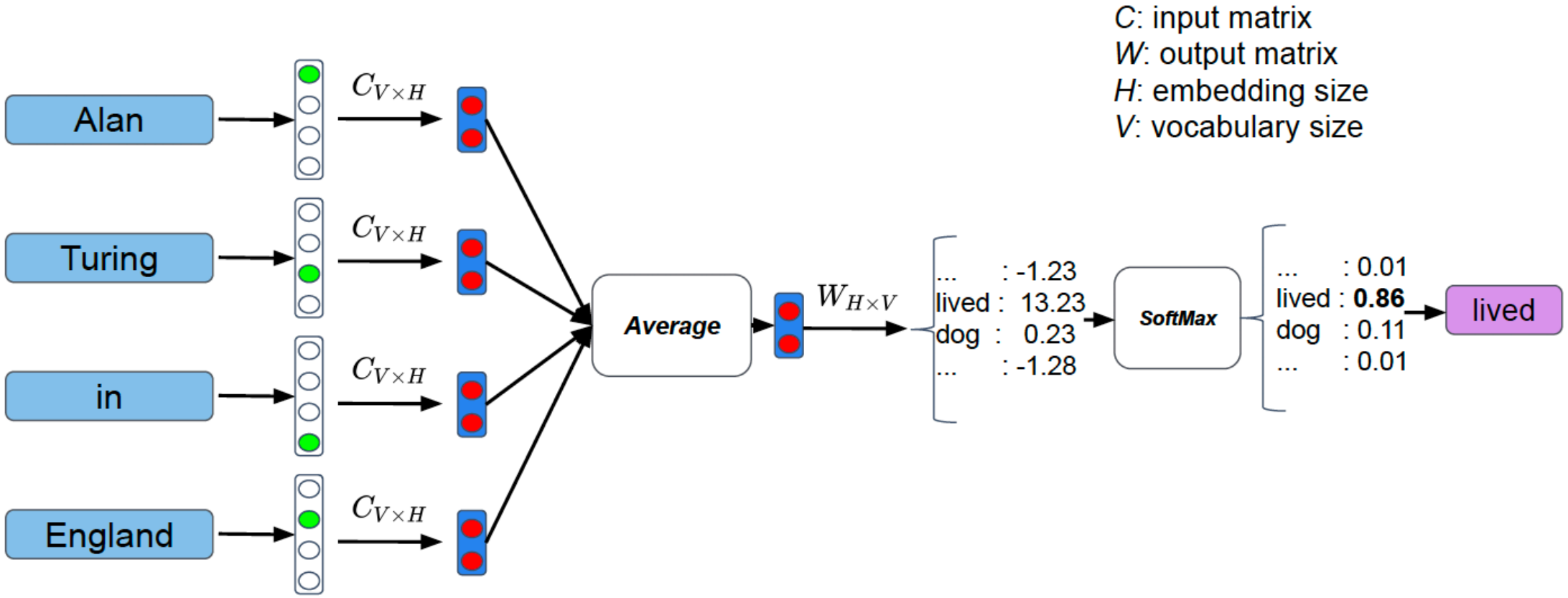
Word2Vec – Continuous bag-of-words

- Task: predict a word given some context window.



Mikolov et al. "Efficient estimation of word representations in vector space."

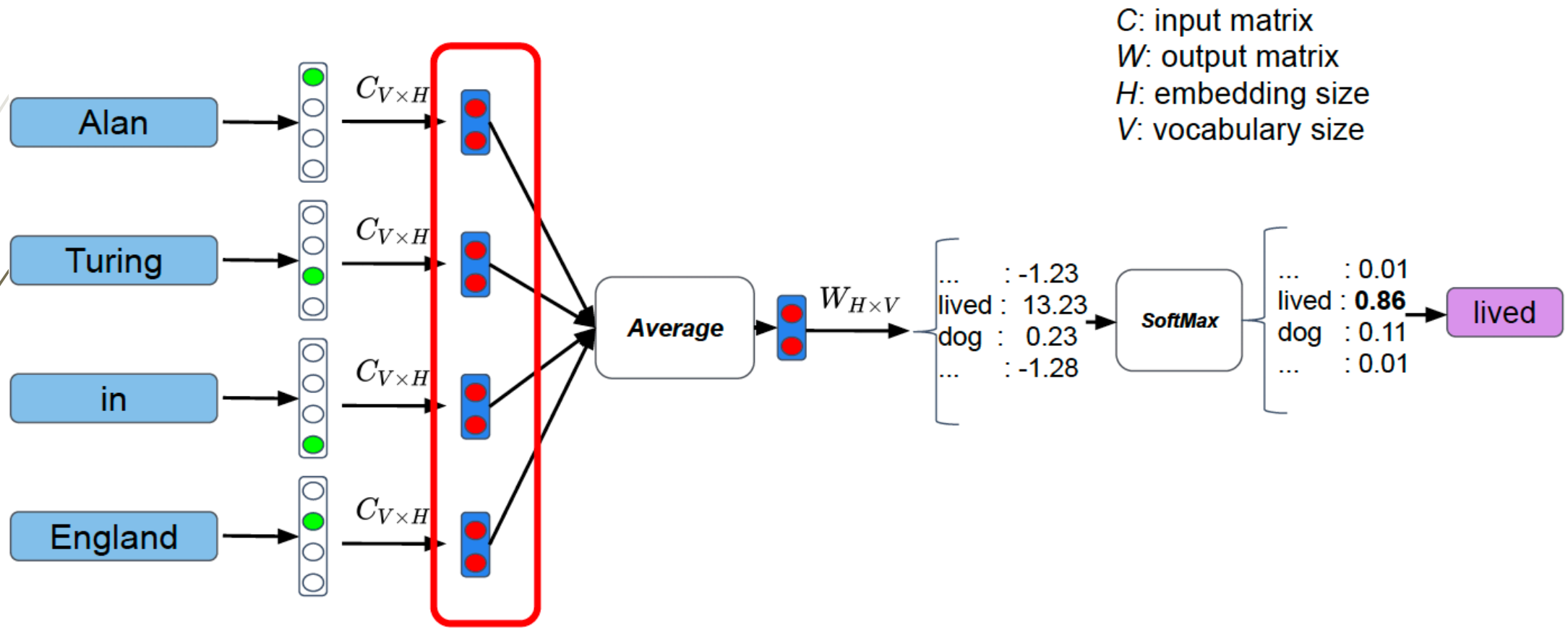
Word2Vec – Continuous bag-of-words



Mikolov et al. "Efficient estimation of word representations in vector space."

Word2Vec – Continuous bag-of-words

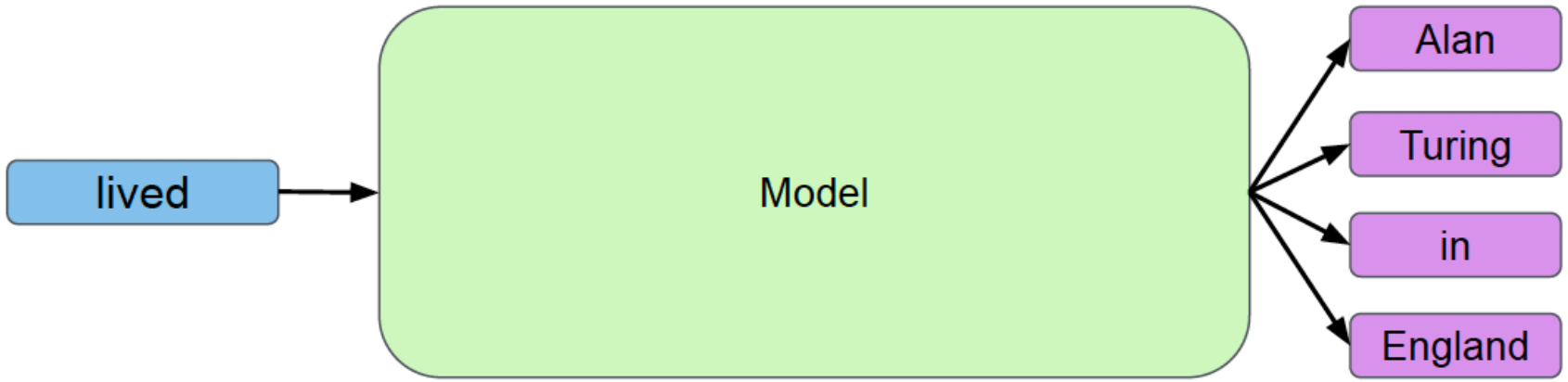
- The embeddings is what we really care about.



Mikolov et al. "Efficient estimation of word representations in vector space."

Word2Vec – Continuous Skip-Gram

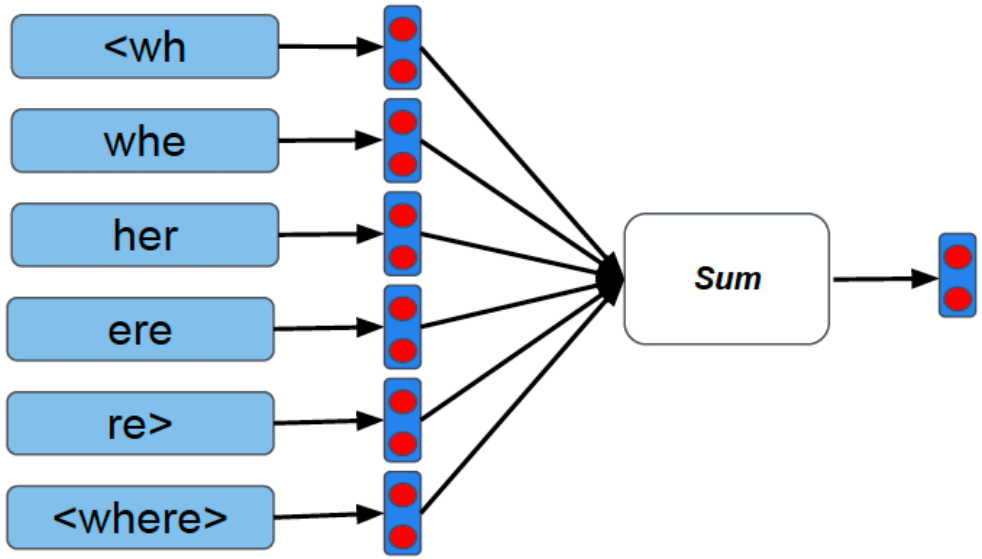
- In the Continuous Skip-Gram version, the Word2Vec algorithm has a different goal:
  - It predicts the context given the central word.
- It works better than continuous bag-of-words in the presence of smaller amounts of data.



Mikolov et al. "Efficient estimation of word representations in vector space."

FastText

- Extension of the Continuous Skip-Gram model to:
  - better handle out-of-vocabulary words,
  - consider the morphology of the words.
- The final embedding for one word is the sum of the original word embedding and the embeddings of its n-grams.



FastText representation of the word "where" if we only use 3-grams

Bojanowski et al. "Enriching Word Vectors with Subword Information."

Pre-Training

- Prior to training a model on a task of interest, the embeddings can be learned in a pre-training phase.
- Pre-training can be greatly beneficial given that algorithms like Word2Vec and FastText can learn embeddings in an unsupervised way, thus making it possible to leverage huge amounts of data.

Pre-Training

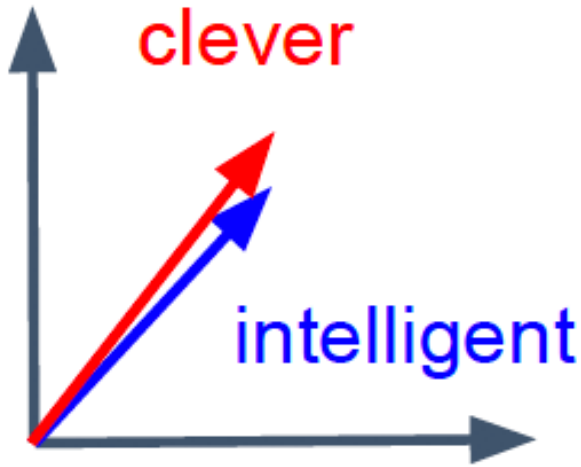
- Once pre-training is done, the embeddings can then used for the real task of interest (i.e., the training phase).
  - This task is usually supervised and cannot therefore leverage large amounts of data.
- This idea greatly helps to deal with scarcity of data in the training phase.

Topics

- ***Word Embeddings***
- ***N-gram Models***
- ***Word Embeddings (continued)***
- Contextualized Word Embeddings

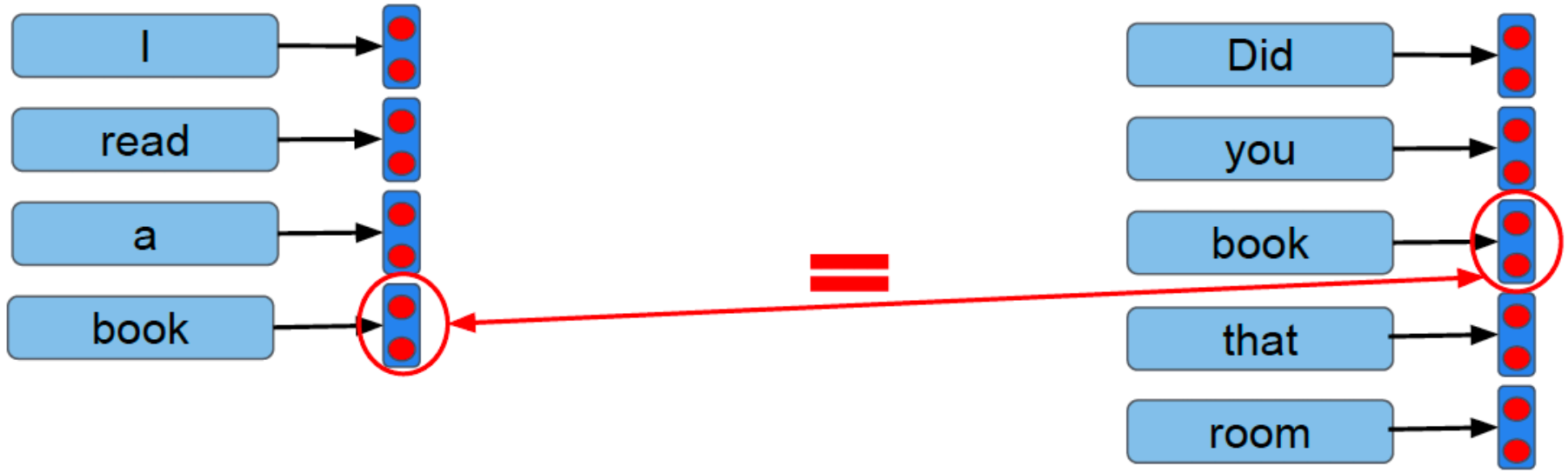
Word Embeddings and Synonymy

- Word embeddings are very useful representations that often lead to improved results.
- This is in part due to the fact that they can capture synonymy properties.
  - Synonyms in general have very close embedding vectors.



Word Embeddings and Polysemy

- Word embeddings do **not** help with polysemy though.
- A word such as "book" has the **same** embedding **regardless of the context**.



Word Embeddings and Polysemy

- Word embeddings reply to the question:

  what is the embedding for “book”?

- In order to consider the context, the question should become:

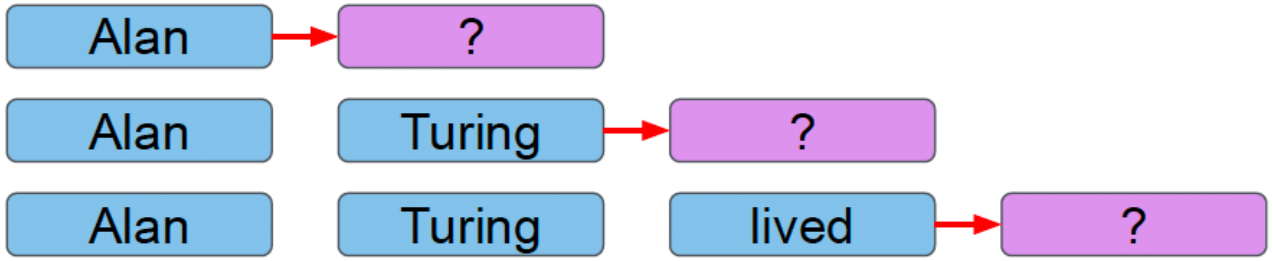
  what is the embedding for “book” in the sentence “I read a book”?

ELMo

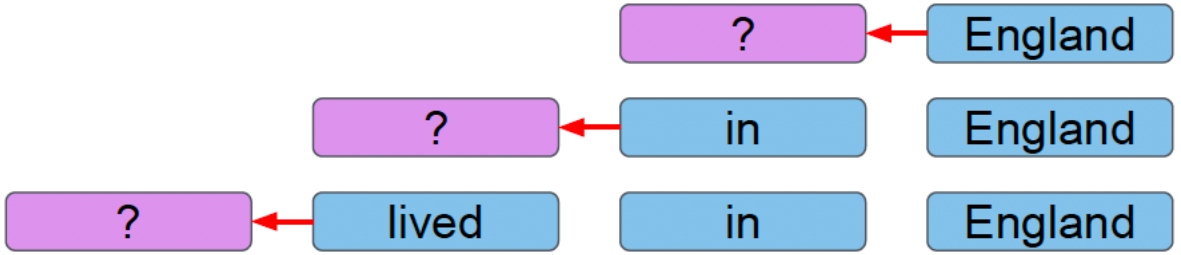
- ELMo (Embeddings from Language Models) proposes a solution to two problems:
  - Generate contextualized word embeddings...
  - … in an unsupervised pre-training phase.
- To pre-train, they select the Language Modeling (LM) task.

Language Modeling

- Given n words (from a sentence), predict the next word (n+1).



- Language Modeling is an unsupervised task.
- Note it can also work 'right – to – left'.

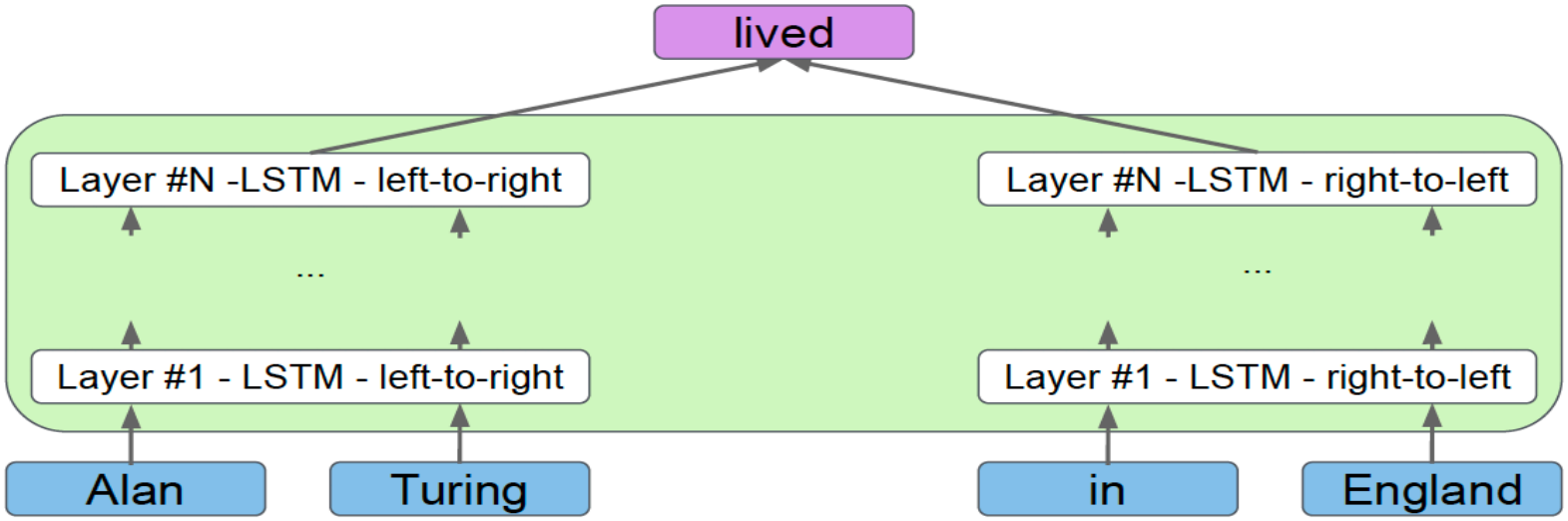


ELMo

- ELMo model is a N-layer bidirectional LSTM.

(Note: **LSTM** (Long Short Term Memory) Networks are called fancy recurrent neural networks with some additional features.)

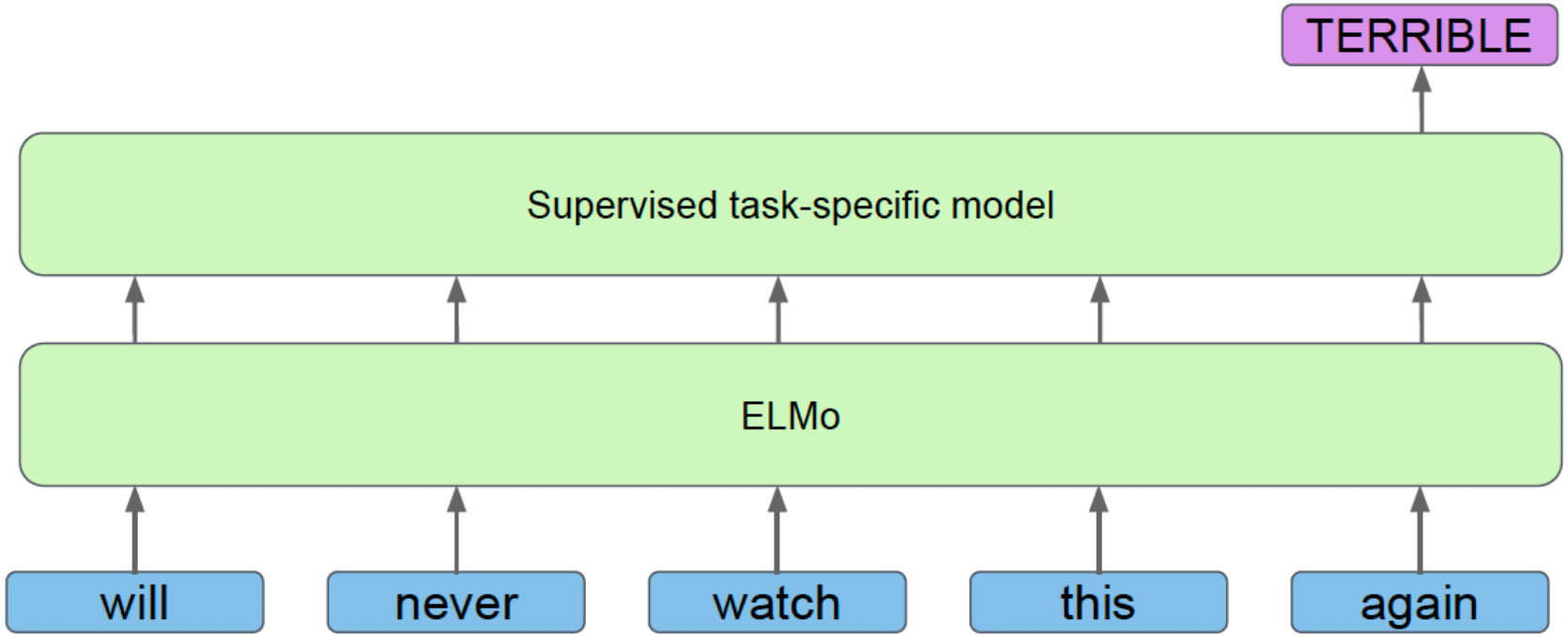
- The contextualized embedding at a time step corresponds to a combination of the hidden states of all the various layers.



Peters et al. "Deep contextualized word representations."

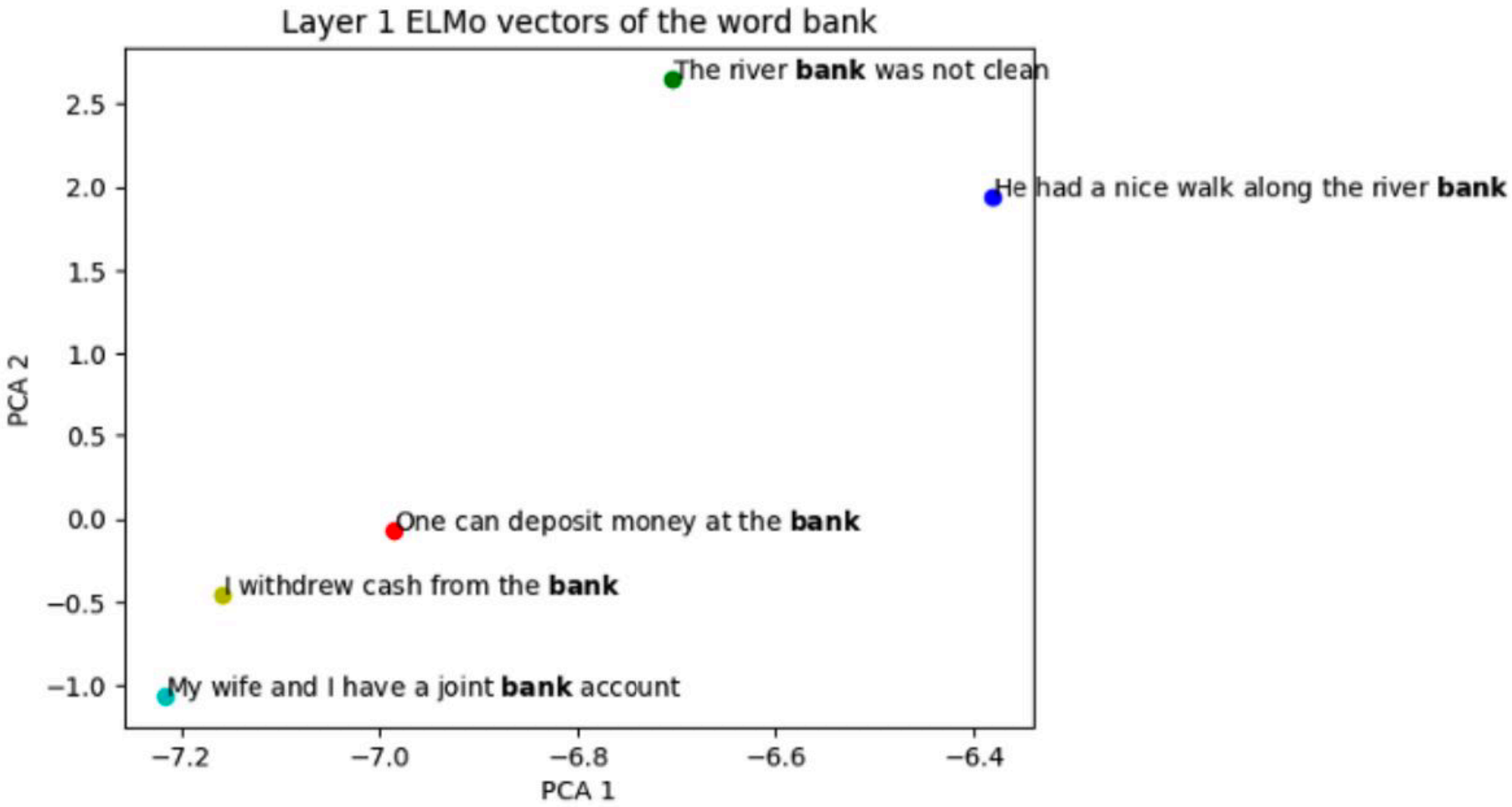
ELMo

- After pre-training, ELMo can be used with any other model (trained on supervised task).



Peters et al. "Deep contextualized word representations."

ELMo - Visualization



The End