

Neural nets for NLP: Word Vectors/Embeddings

Neural Networks

Word vectors/embeddings

dog

cat

pizza

Word vectors/embeddings

dog

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

cat

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

pizza

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

Word vectors/embeddings

dog

$$\begin{bmatrix} 0.8 \\ 0.3 \\ 0.1 \end{bmatrix}$$

cat

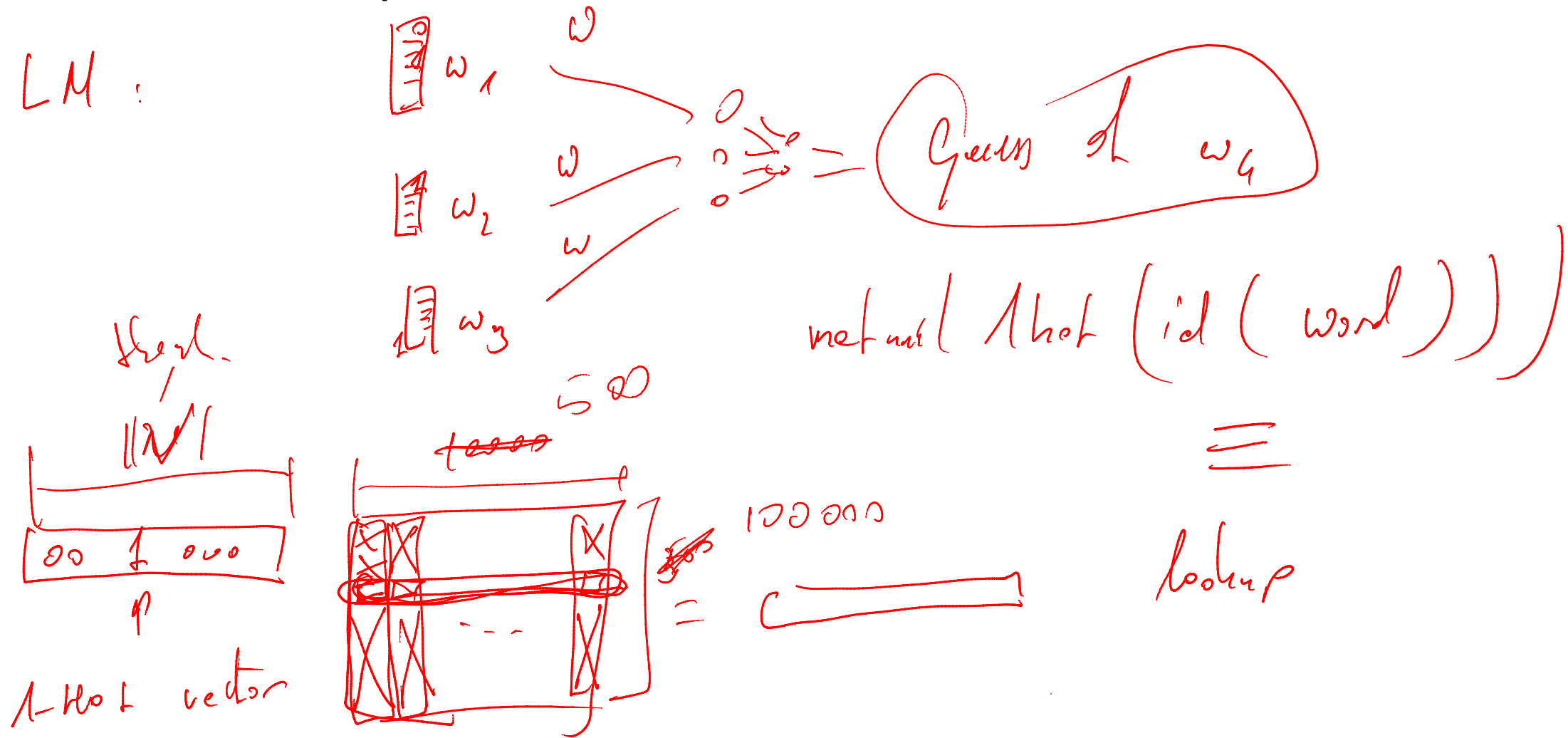
$$\begin{bmatrix} 0.7 \\ 0.5 \\ 0.1 \end{bmatrix}$$

pizza

$$\begin{bmatrix} 0.1 \\ 0.2 \\ 0.8 \end{bmatrix}$$

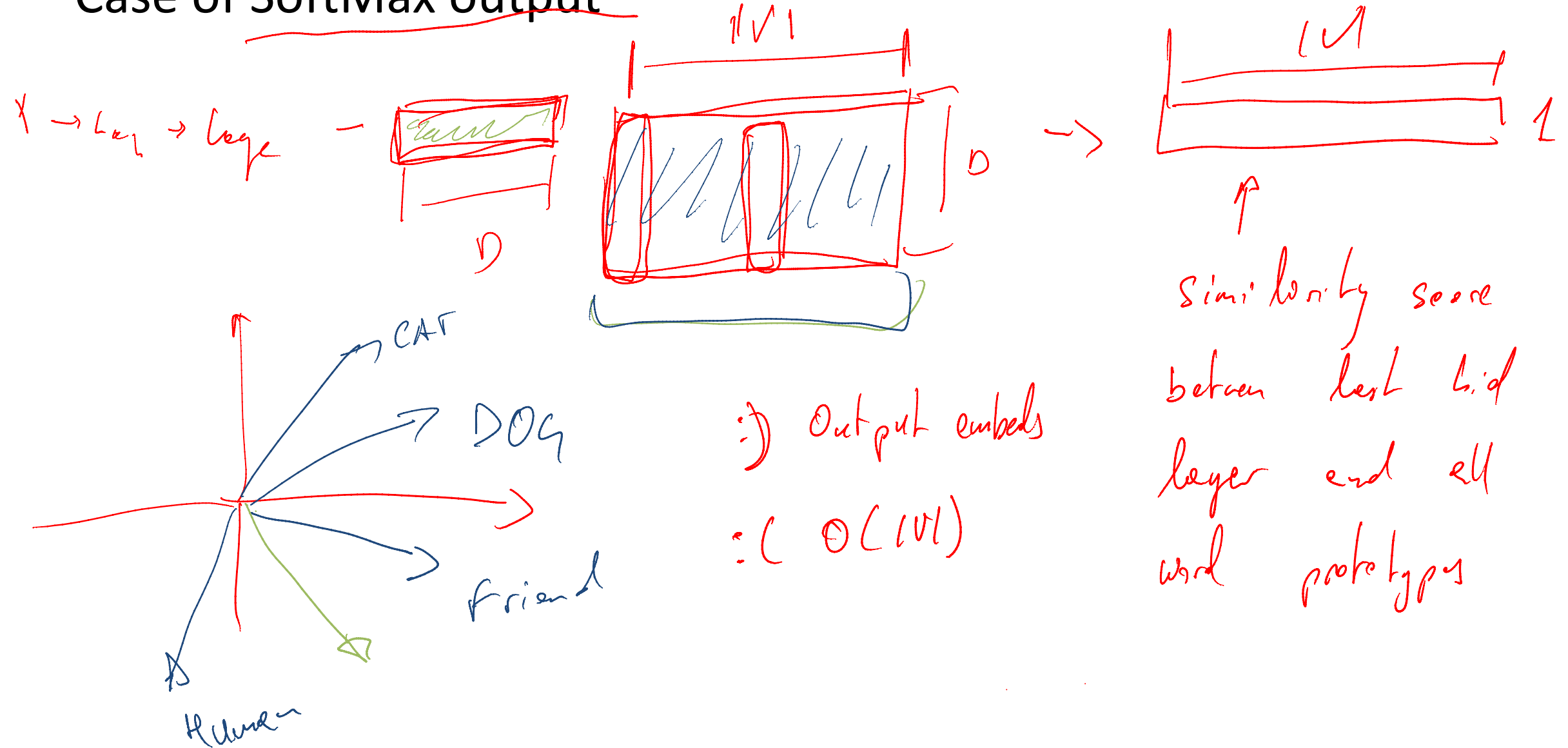
1-Hots & SoftMax: embeddings ahead

Case of 1-Hot inputs



1-Hots & SoftMax: embeddings ahead

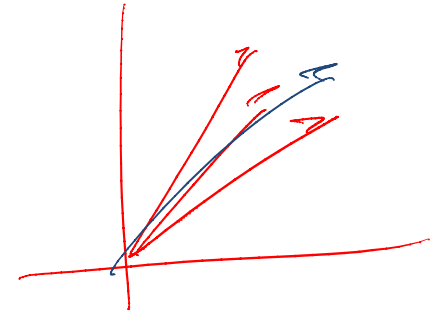
Case of SoftMax output



Embeddings: by-product of language models

Encoding of words:

represent each word by its own vector
we call it the word embedding



Decoding of responses (next word prediction)


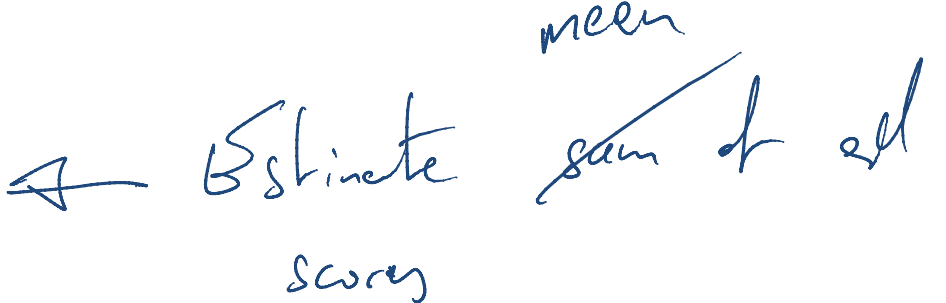
each word is assigned an output vector (output embedding)

Bottom line: each word is implicitly represented as a vector (in \mathbb{R}^D)

The word vectors are **parameters** of the network, which are trained as usual
Similar words get similar vectors

Trouble with a large SoftMax

We have two embedding matrices: input and output, both assign a vector to each word.

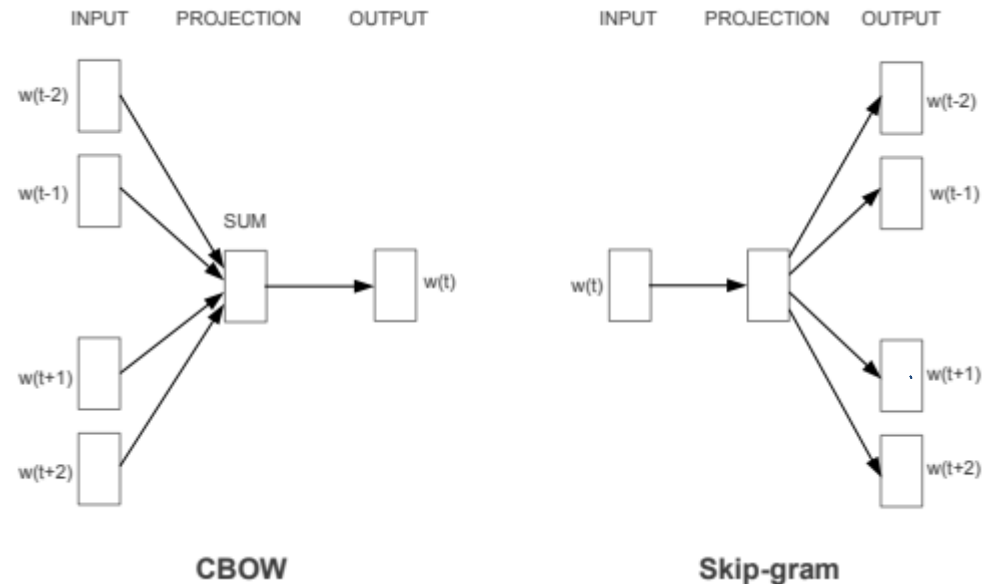
- To encode a word, we do table lookup in time $O(1)$, independent of dictionary size
- To apply the SoftMax we need to:
 1. score all words and 
 2. normalize the scores to probabilities, 
- Idea: sample a subset of words, which include the correct one, and only apply softmax to these. Scales with $O(\text{sample size})$!

See e.g. Tensorflow docs: https://www.tensorflow.org/extras/candidate_sampling.pdf

Word2Vec: fast word vectors

- Do we need to train a large language model to learn word vectors?
- Train small models to predict words based on context (or vice-versa)

Intuition: We understand the meaning from the context



- Word2Vec is blazing fast
 - Only 1 matrix multiplication (projection + sum)
 - Sampled SoftMax, or negative sampling:
sample some words, train a logistic output for each word
(no need to score and normalize over all words)

Word2Vec

Pre-trained word and phrase vectors

We are publishing pre-trained vectors trained on part of Google News dataset (about 100 billion words). The model contains 300-dimensional vectors for 3 million words and phrases. The phrases were obtained using a simple data-driven approach described in [2]. The archive is available here: [GoogleNews-vectors-negative300.bin.gz](#).

An example output of `./distance GoogleNews-vectors-negative300.bin:`

```
``` Enter word or sentence (EXIT to break): Chinese river
```

## Word Cosine distance

Yangtze_River	0.667376
Yangtze	0.644091
Qiantang_River	0.632979

<https://code.google.com/archive/p/word2vec/>

# FastText

*fast*Text

Library for efficient text classification and representation learning

GET STARTED

DOWNLOAD MODELS

## What is fastText?

FastText is an open-source, free, lightweight library that allows users to learn text representations and text classifiers. It works on standard, generic hardware. Models can later be reduced in size to even fit on mobile devices.

## Download pre-trained models



### English word vectors

Pre-trained on English webcrawl and Wikipedia



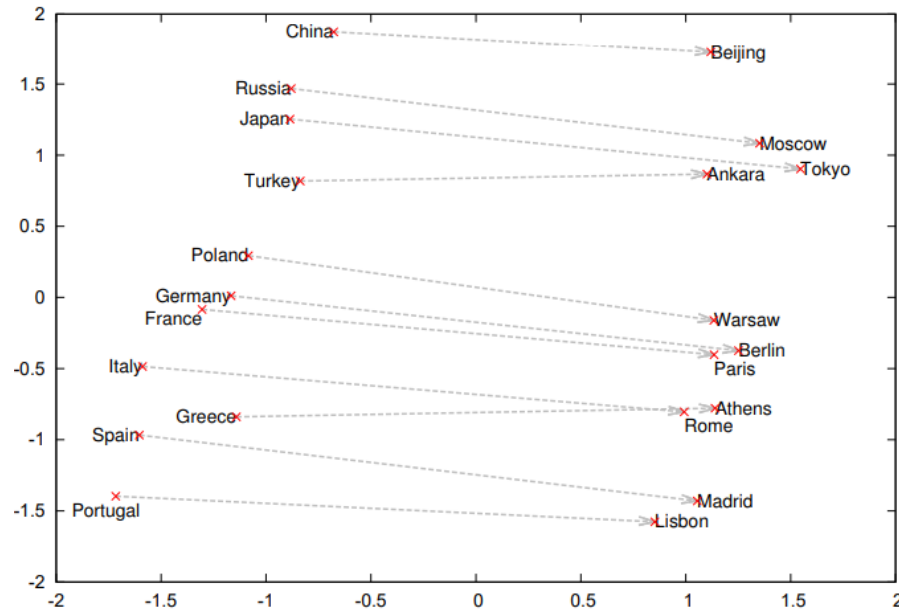
### Multi-lingual word vectors

Pre-trained models for 157 different languages

<https://fasttext.cc/>

# Word vector arithmetic

- King – queen  $\approx$  man – woman
- Italy – Rome  $\approx$  Poland – Warsaw



Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwana	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

# Wordpieces

- possible <-> IMpossible
- possible <-> possibLITY
- possible <-> IMpossibLITY

Problem #1: there are millions of words

Problem #2: words are related. Even English has limited word flexion, other languages have more

Idea: use characters, or subword units (wordpieces)

# FastText: sum overlapping char ngrams

Embedding(where) ==

$$E(<wh) + E(whe) + E(her) + E(ere) + E(re>) + E(<where>)$$

Model a word as a sum of embeddings of all its character ngrams.  
Use special beginning/end-of-word symbols <>

# Wordpieces aka BPE (byte-pair encoding)

Consider a corpus:

„I like walking, I walk a lot, a walking lot.”

1. Start  $D$  with a set containing all letters
2. Look at all pairs of letters, add the most frequent to  $D$ , replace all occurrences by the new symbol
3. Repeat 2. until  $D$  reaches a given size (e.g. 30k units).

First few merges:  $w + a \rightarrow wa$     $l + k \rightarrow lk$     $wa + lk \rightarrow walk$

This detects frequent morphemes!

<https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/37842.pdf>

<https://www.aclweb.org/anthology/P16-1162/>