Regular exp. (cont.)

1 turkey costs 30 $ → Positive lookahead

Is there a digit followed by a $

\d+ (?=\$)

30

Positive lookahead

Is there a digit not followed by $

\d+ (?!\$) → Negative lookahead

1

zero/none

(?<=\$)\d+ → positive lookbehind

(?<!\$)\d+ → Negative lookbehind

Text Normalization

A set of collective tasks
Primary pre-processing tasking before feeding to model

1. Tokenization
2. Normalizing word formats
3. Segmenting sentences

1 " The U.S. is a big nation.

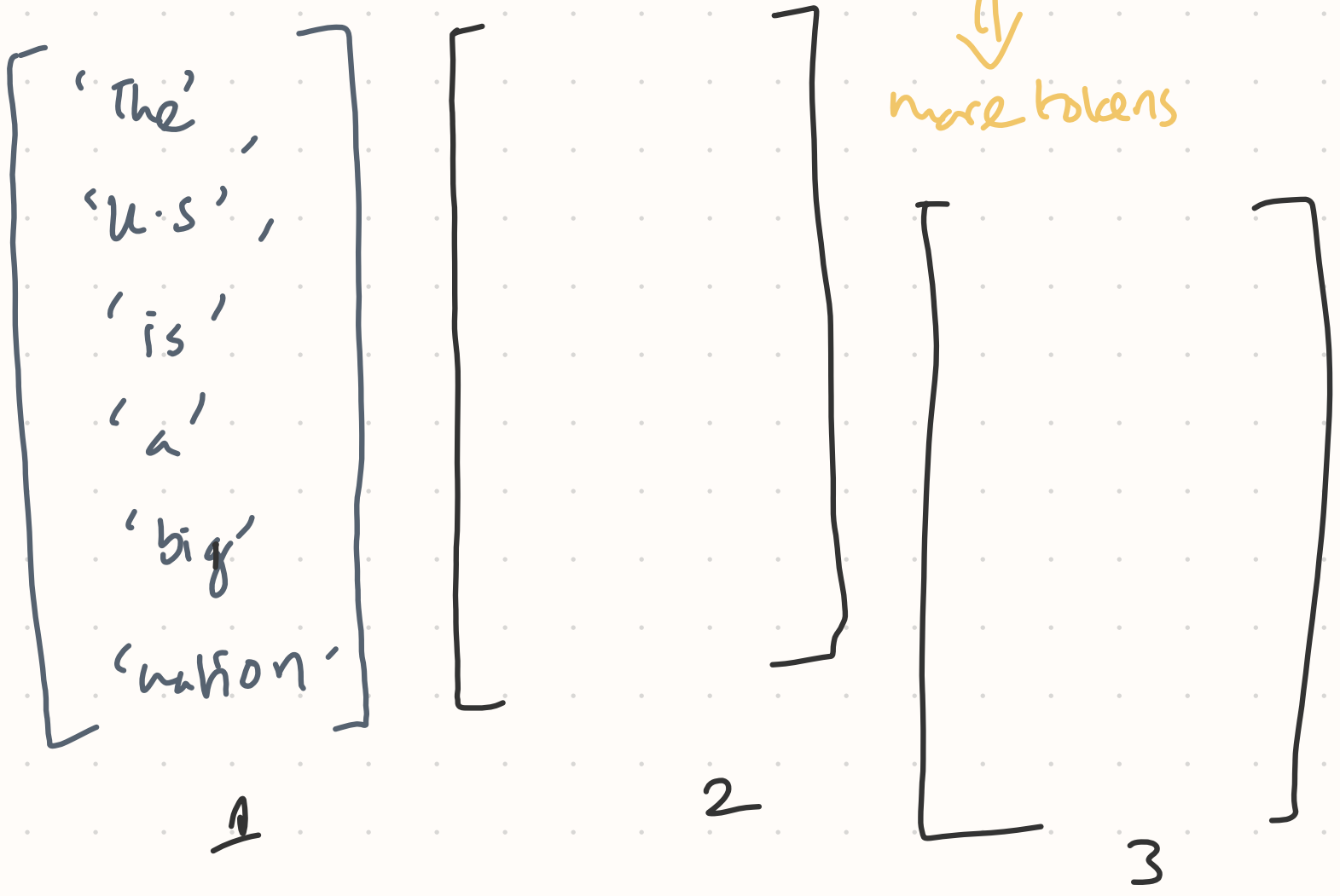
2 The U.S.A poster-print costs 30 $.

3 The driving speed unit here is m.p.h, not km. "

Embedded vectors

more words

more tokens



Tokenization : segmenting a running text into words (tokens)

Represent tokens (words) as numbers

3 diff size vectors

Rule based tokenization — prev.

Each sentence end w/ . → for 1st sentence you get 3 diff sentences

abbreviations periods] .

white space split →

↙ prob

Penn Treebank standard (PTS) → not all . are sentence breakers

↙ soln

post-print — 1 single token

'30', '5'

PTS + regex

Based on tokenization, it depends how effective embedding will be.

Before bert/transformer based models — RNN earlier

bert-based model — embedded vector

Power of transformer / pretraining

Knows

which words are similar

king → queen → cluster1
student → teacher → cluster2
country → patriot

} cluster of tokens

corpus

'nation' : similar vector value like country (Embedding vector)

RNN doesn't have this info only bert has

bert tokenizer / bert embedding

better representation of words ✓ → optimizing parameters

automatic token parsing techniques: (Tokenization)

1. Byte pair Encoding (BPE)
2. unigram language modeling (ULM)
3. wordpiece → bert

} 2 parts:
- token learner
- token segmenter

[sentence piece : BPE + ULM]

Corpus → { low, new, newer, wider, lowest }

PTS fails to understand what to do w/ unknown words.

BPE: Corpus – 18 tokens

Divide all words based on whitespace and word frequency

all unique chars ← vocabulary

count	corpus	vocabulary
5	low_ → whitespace	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_
2	lowest_	
6	newer_	
3	wider_	
2	new_	

morphemes

Task: create a vocab

List most frequent pair of symbols together

er = 6 + 3 = 9

n e w er_

w i d er_

n e w er_

w i d er_

(Learns an unseen data.
new word:
lower rule
low, er_
present in learner vocab
low er_
diff token (merged)

morphemes: smallest unit of

BPE → lower < low, er > } 2 tokens

Subwording: Dividing the tokens into smaller parts

Subword = smartphone

ne w er_

ne w _

BPE: K parameter ⇒ how many cycles you want to run

BPE(corpus, K)

9

Normalizing:

am is are — same word containing same meaning

case folding:

(

lower case everything

Woodhouse
woodhouse
} same value for embedding vector

→ Info retrieval ✓

Text/sentiment classification
+ machine translation

(

not doing lower casing

US → country
us → we

Lemmatization: Determine which words are the same root

addition
additional
} same root

I have some additional information.

I can do some addition on information.

same

words (additional) ⟨ stems (addition)
                    ⟨ Affixes (al)

replace each word w/ roots → Lemmatization ✓ (better)
using pre existing dictionary
sings, sang, sung

Do Lemmatization before feeding corpus to language model.

Stemming : Remove affixes from words
✗

~~additional~~

Changed actual meaning

(doing → doe) → error prone (stemming)

(doing → do) → not error prone (Lemmatization)

happiness → happy

"ness" "ness"

Lemmatization over stemming ✓