Introduction, Basic Text Processing, Edit Distance

Language Technology

mostly solved

Spam detection

Let's go to Agra! ✓

Buy V1AGRA … ✗

Part-of-speech (POS) tagging

ADJ ADJ NOUN VERB ADV

Colorless green ideas sleep furiously.

Named entity recognition (NER)

PERSON ORG LOC

Einstein met with UN officials in Princeton

making good progress

Sentiment analysis

Best roast chicken in San Francisco!

The waiter ignored us for 20 minutes.

Coreference resolution

Carter told Mubarak he shouldn't run again.

Word sense disambiguation (WSD)

I need new batteries for my ***mouse***.

Parsing

I can see Alcatraz from the window!

Machine translation (MT)

第13届上海国际电影节开幕…

The 13th Shanghai International Film Festival…

Information extraction (IE)

You're invited to our dinner party, Friday May 27 at 8:30

Party
May 27
add

still really hard

Question answering (QA)

Q. How effective is ibuprofen in reducing fever in patients with acute febrile illness?

Paraphrase

XYZ acquired ABC yesterday

ABC has been taken over by XYZ

Summarization

The Dow Jones is up

The S&P500 jumped

Housing prices rose

Economy is good

Dialog

Where is Citizen Kane playing in SF?

Castro Theatre at 7:30. Do you want a ticket?

Text Processing 1. Regular expression

- A formal language for specifying text strings
  How can we search for any of these?

- Apple
- apple
- Apples
- apples

Regular expression: Disjunction

- Letters inside square brackets []

Patterns	Matches
[Aa]pple	Apple, apple
[1234567890]	Any digit

- Ranges

Patterns	Matches	
[A-Z]	Any upper case letter	I am going.
[a-z]	Any lower case letter	what are you doing
[0-9]	Any digit	Chapter 1:

Regular expression: Negation in Disjunction

- Negations [^Ss] : carat means negation if used as 1st char inside []

Patterns	Matches	
[^A-Z]	Not an upper case letter	I<u>n</u>ot am going.
[^aA]	Not a or A	<u>w</u>hat are you doing
[^e^]	Neither e nor ^	e<u>l</u>even. ^<u>g</u>o
a^b	Pattern a carat b	Find <u>a^b</u>

Regular expression: More Disjunction

- Pipe | is used for disjunction.

<table>
<tr><th>Patterns</th><th>Matches</th></tr>
<tr><td>apple|banana</td><td>apple, banana</td></tr>
<tr><td>1|2|3</td><td>=[1-3]</td></tr>
<tr><td>[aA]pple|[bB]anana</td><td></td></tr>
</table>

Regular expression: * + ? .

Patterns	Matches	
oo*h!	Zero or more of the previous character	oh!, ooh!, oooh!,..
oo+h!	one or more of the previous character	ooh!, oooh!,…; does not match oh!
colou?r	Zero or one of the previous character	colour, color
col.r	Any character	color, coler, colar,colwr

Regular expression: Anchor ^ $

- ^ matches the beginning of a sentence, $ matches the end of a text.

Patterns	Matches
^[A-Z]	**<u>G</u>**pple
^[^a-zA-Z]	**<u>1</u>** <u>“</u>hi”
\.$	end<u>.</u>
.$	end<u>.</u> end<u>!</u>

Regular expression: examples

- Find all occurrences of word ‘the’ in the text.

Patterns	Matches
the	misses capitalized examples
[tT]he	Incorrectly captures other
[^a-zA-Z][tT]he[^a-zA-Z]	correct.

Visit https://regexr.com/ to test all these regular expressions type.

Text Processing 2. Word Tokenization

- Every NLP task needs to do text normalization:
  1. Segmenting/tokenizing words in running text
  2. Normalizing word formats
  3. Segmenting sentences in running text

Words

- Lemma: same stem, part of speech, rough word sense
  - cat and cats = same lemma
- Wordform: the full inflected surface form
  - cat and cats = different wordforms
- Type: an element of the vocabulary.
- Token: an instance of that type in running text

they lay back on the San Francisco grass and looked at the stars and their

15 tokens (or 14),

13 types

Tokenization in Linux

- Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt
     | sort
     | uniq –c
```

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

```
1945 A
  72 AARON
  19 ABBESS
   5 ABBOT
 ... ...
```

```
25 Aaron
 6 Abate
 1 Abates
 5 Abbess
 6 Abbey
 3 Abbot
.... …
```

The first step: tokenizing

```
tr -sc 'A-Za-z' '\n' < shakes.txt | head
```

```
THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...
```

The second step: sorting

```
tr -sc 'A-Za-z' '\n' < shakes.txt | sort | head

A
A
A
A
A
A
A
A
A
...
```

The third step: counting

- Merging upper and lower case

```
tr 'A-Z' 'a-z' < shakes.txt | tr –sc 'A-Za-z' '\n' | sort | uniq –c
```

- Sorting the counts

```
tr 'A-Z' 'a-z' < shakes.txt | tr –sc 'A-Za-z' '\n' | sort | uniq –c | sort –n –r
```

```
23243 the
22225 i
18618 and
16339 to
15687 of
12780 a
12163 you
10839 my
10005 in
8954  d
```

What happened here?

Tokenization issues

- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??

Language issues:

- Chinese and Japanses words are not separated by spaces
- Use maximum matching to find out tokens
- Doesn't generally work in English! - Thetabledownthere

Modern Tokenizers: BPE, Wordpiece

- Word Tokenizers
- Character Tokenizers
- Sub-word Tokenizers
  - Developed for machine translation by Sennrich et al., ACL 2016

    "The main motivation behind this paper is that the translation of some words is transparent in that they are translatable by a competent translator even if they are novel to him or her, based on a translation of known subword units such as morphemes or phonemes."

  - Later used in BERT, T5, RoBERTa, GPT, etc.
  - Relies on a simple algorithm called *byte pair encoding* (Gage, 1994)

Modern Tokenizers: BPE, Wordpiece

Byte pair encoding

- Form base vocabulary (all characters that occur in the training data

word	frequency
hug	10
pug	5
pun	12
bun	4
hugs	5

- Base vocab: b, g, h, n, p, s, u

Example from https://huggingface.co/transformers/tokenizer_summary

Modern Tokenizers: BPE, Wordpiece

Byte pair encoding

- Now, count up the frequency of each character *pair* in the data, and choose the one that occurs most frequently

word	frequency
h+u+g	10
p+u+g	5
p+u+n	12
b+u+n	4
h+u+g+s	5

character pair	frequency
ug	20
pu	17
un	16
hu	15
gs	5

...

Example from https://huggingface.co/transformers/tokenizer_summary.ht

Modern Tokenizers: BPE, Wordpiece

Byte pair encoding

- Now, choose the most common pair (ug) and then merge the characters together into one symbol. Then, retokenize the data

word	frequency
h+*ug*	10
p+*ug*	5
p+u+n	12
b+u+n	4
h+*ug*+s	5

character pair	frequency
un	16
h+ug	15
pu	12
p+ug	5
ug+s	5

...

Example from https://huggingface.co/transformers/tokenizer_summary.ht

Modern Tokenizers: BPE, Wordpiece

Byte pair encoding

- Keep repeating this process! This time we choose *un* to merge, next time we choose *h+ug*, etc.

word	frequency
h+*ug*	10
p+*ug*	5
p+u+n	12
b+u+n	4
h+*ug*+s	5

character pair	frequency
un	16
h+ug	15
pu	12
p+ug	5
ug+s	5

...

Example from https://huggingface.co/transformers/tokenizer_summary.html

Modern Tokenizers: BPE, Wordpiece

Byte pair encoding

- Eventually, after a fixed number of merge steps, we stop

word	frequency
hug	10
p+*ug*	5
p+*un*	12
b+*un*	4
hug + s	5

- new vocab: b, g, h, n, p, s, u, *ug*, *un*, *hug*

Modern Tokenizers: BPE, Wordpiece

Byte pair encoding

- To avoid <UNK>, all possible characters / symbols need to be included in the base vocab. This can be a lot if including all unicode characters!

- GPT-2 uses *bytes* as the base vocabulary (size 256) and then applies BPE on top of this sequence (with some rules to prevent certain types of merges).

- Commonly have vocabulary sizes of 32K to 64K

Modern Tokenizers: BPE, Wordpiece

- WordPiece (Schuster et al., ICASSP 2012): merge by likelihood as measured by language model, not by frequency

- SentencePiece (Kudo et al., 2018): can do subword tokenization without pretokenization *(good for languages that don't always separate words w/ spaces)*, although pretokenization usually improves performance

Text Processing 3. Normalization & Stemming

- We implicitly define equivalence classes of terms
  - deleting periods in a term: U.S.A and USA
  - asymmetric expansion : window → window, windows, Windows
    Windows → Windows
- Case folding: convert every letter to lowercase
  - Exception: uppercase in the middle of sentence – General Motors, SAIL
  - For MT, Sentiment analysis etc, case may be useful (US vs us)

Lemmatization

- Reduce inflections or variant forms to base form
  - *am, are, is* $\rightarrow$ *be*
  - *car, cars, car's, cars'* $\rightarrow$ *car*
- *the boy's cars are different colors* $\rightarrow$ *the boy car be different color*
- have to find correct dictionary headword form

Morphemes

- The small meaningful units that make up words
    - **Stems**: The core meaning--bearing units
    - **Affixes**: Bits and pieces that adhere to stems
        - Often with grammatical functions

Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

→

for exampl compress and compress ar both accept as equival to compress

Porters Stemmer

Step 1a

```
sses → ss   caresses → caress
ies  → i    ponies   → poni
ss   → ss   caress   → caress
s    → ø    cats     → cat
```

Step 1b

```
(*v*)ing → ø  walking   → walk
              sing      → sing
(*v*)ed  → ø  plastered → plaster
…
```

Step 2 (for long stems)

```
ational→ ate relational→ relate
izer→ ize    digitizer → digitize
ator→ ate    operator  → operate
…
```

Step 3 (for longer stems)

```
al   → ø  revival    → reviv
able → ø  adjustable → adjust
ate  → ø  activate   → activ
…
```

```
(*v*)ing → ø  walking → walk
              sing    → sing
```