MUNI FI

Utok – The Fast Rule-based Tokenizer

Pavel Rychlý **Samuel Špalek**

NLP Centre, FI MU

Tokenization

- split text into toknes
- tokens = minimal meaningful elements

Example

utok: a fast “tokenizer”

```
utok : a fast " tokenizer "
```

(4 words, 7 tokens)

Applications

- current trend in deep learning: subword tokens
  - learned automatically from training data
  - not much meaningful
- rule-beased tokens still used
  - part of speech tagging
  - searching in text corpora (`"hunt" []{0,4} "kill"`)
  - computing BLEU score for machine translation

Keeping spaces

- spaces are sometimes important
- spaces on correct places make text more readable
- the puctuation meaning depends on surrounding spaces
  - Booking.com, 2022-12-10, 12 - 10
- using empty element <g/> (glue) to mark a connection of tokens without space

Example

utok: a fast "tokenizer"

```
utok <g/> : a fast " <g/> tokenizer <g/> "
```

Subword tokens

- minimal meaningful part could be smaller than whole word
- required for better POS tagging, lemmatization

Word, lemma, tag

didn't ask

```
did     do     VVD
<g/>
n't     not    RB
ask     ask    VV
```

Subword tokens

Query: *do not ask*

(3 tokens)

for archival purposes. </s><s> [4] We	**do not ask**	'functional intactness' of the telescope.
been informed by somebody at , now	**don't ask**	me who cos I don't know, but from one
Yes yes. </s><s> You know if the guy	**doesn't ask**	the ask the price fine. </s><s> Yeah. </
ith her elderly mother-in-law. </s><s> '	**Don't ask**	!' </s><s> Laura groaned. </s><s> 'I co
> If the District Council were slack and	**didn't ask**	them to do it, we, we have no guarante
guy's bound to say erm and even if he	**doesn't ask**	you what the price is you must make su

Unitok

Unitok – a tokenizer from Brno pipelines

- a configuration for each language
- in Python
- configuration: **list** of regular expressions
- method: recurrent splitting of an input line
- not so fast

Unitok configuration

```
HTMLENTITY = ur"&(#x?[0-9A-F]+|\w+);"
HTMLENTITY_RE = re.compile(HTMLENTITY)

DECADE = ur"""
(?<![-\w])
    ( '\d0s | (\d\d)?\d0'?s )
(?![-\w])
"""
DECADE_RE = re.compile(DECADE, re.UNICODE | re.VERBOSE)

CLITIC = ur"""
    ( (?<=\w)('s|'re|'ve|'d|'m|'em|'ll|n't) | (?<=can)not )
(?!\w)
"""
CLITIC_RE = re.compile(CLITIC, re.UNICODE | re.VERBOSE | re.IGNORECASE)

WORD = ur"\w[\w-]*\w|\w"
WORD_RE = re.compile(WORD, re.UNICODE)

ANY_SEQUENCE = ur"(.)\1*"
ANY_SEQUENCE_RE = re.compile(ANY_SEQUENCE)

re_list = [
    ('HTMLENTITY', HTMLENTITY_RE),
    ('DECADE', DECADE_RE),
    ('CLITIC', CLITIC_RE),
    ('WORD', WORD_RE),
    ('ANY_SEQUENCE', ANY_SEQUENCE_RE),
]
```

Utok

- shorter, smaller, faster, simpler
- C++, using re2 library (67 lines)
- congiguration: **set** of regular expressions
- method:
  - find first longest regexp match
  - output token (with optional <g/>)
- handling of subwords:
  - special `*SPLIT` attribute

Utok configuration

```
# XML entity
&(amp|lt|gt|quot|apos);

# Decades
'\d0s
(\d\d)?\d0'?s\b

*SPLIT (?i)(.*[^\d])('s|'re|'ve|'d|'m|'em|'ll|n't)
*SPLIT (?i)(can)(not)

# words
[\pL\pM\pN_]+(-[\pL\pM\pN_]+)*('s|'re|'ve|'d|'m|'em|'ll|n't)?

# puctuation = any non-word, non-space
[?!]+
''
\.+|\*+|:+|=+
[^\pL\pM\pN\pZ]
```

Speed evaluation

Program	English (17 MB)	English (91 MB)	Lorem ipsum with html (96 KB)	Czech (369 MB)
Utok	2.0s	16.7s	56.9ms	25.8s
Utok (no split)	1.0s	7.5s	49.9ms	25.8s
Unitok	12.3s	224.3s	426.3ms	621.6s

Unitok - Utok differences

```
//                                  / <g/> /
word <g/> # <g/> hashtag            word <g/> #hashtag
Title G. A. S <g/> ..               Title G. A. S. <g/> .
```

Conclusion

- utok is shorter, smaller, simpler
- 10-20 times faster
- almost same output as unitok

MUNI
FACULTY
OF INFORMATICS