

# Tokenization



Given a character sequence and a defined document unit,

**tokenization** is the task of chopping it up into pieces, called tokens , perhaps at the same time throwing away certain characters, such as punctuation.

Light TAG

PERSON  
Prime Minister of Denmark, Lars Løkke Rasmussen

COMPANY PLACE  
It was an honor to welcome the Prime Minister of Denmark,  
PERSON  
Lars Løkke Rasmussen {@larsloekke} to the @WhiteHouse yes...  
<https://t.co/N1gOTiVnSp>

PERSON PERSON PERSON  
Lars Løkke Rasmussen

PLACE  
@WhiteHouse

PLACE  
@WhiteHouse

#ConfirmGorsuch

#MakeAmericaGreatAgain

# Tokenization is the most important choice in NLP

That people don't think about

- Tokens are the smallest unit your model can see
- If you make them too small they are meaningless
- If you make them too big
  - You get too many
  - And not enough data to learn
- A change in tokenization can make a task easy or hard

# The difference a space can make

The screenshot shows the Google Translate web interface. At the top, the Google logo is on the left, and a 'Sign in' button is on the right. Below the logo, the word 'Translate' is displayed in red. To the right of 'Translate' is a link that says 'Turn off instant translation' and a star icon. The main interface has two language selection rows. The first row has buttons for 'German', 'English', 'Spanish', and a 'Detect language' dropdown. The second row has buttons for 'English', 'German', 'Spanish', and a 'Translate' button. Below these buttons are two large text areas. The left text area contains a single space character ' '. At the bottom left of the left text area are icons for voice input and a keyboard layout. At the bottom right of the left text area is a character count '0/5000'. The right text area is empty.

Google

Sign in

Translate

Turn off instant translation

German English Spanish Detect language ▼

↔ English German Spanish ▼ Translate

0/5000

# But punctuation and white space are good indicators

President Donald Trump won't pay \$7.73 for his hat ⇒

President

Donald

Trump

won

t

pay

\$7

.73

for

his

hat

# Should everything be a different token ?

- 0. *frog*
- 1. frogs
- 2. toad
- 3. *litoria*
- 4. leptodactylidae
- 5. *rana*
- 6. lizard
- 7. *eleutherodactylus*



3. *litoria*



4. leptodactylidae



5. *rana*



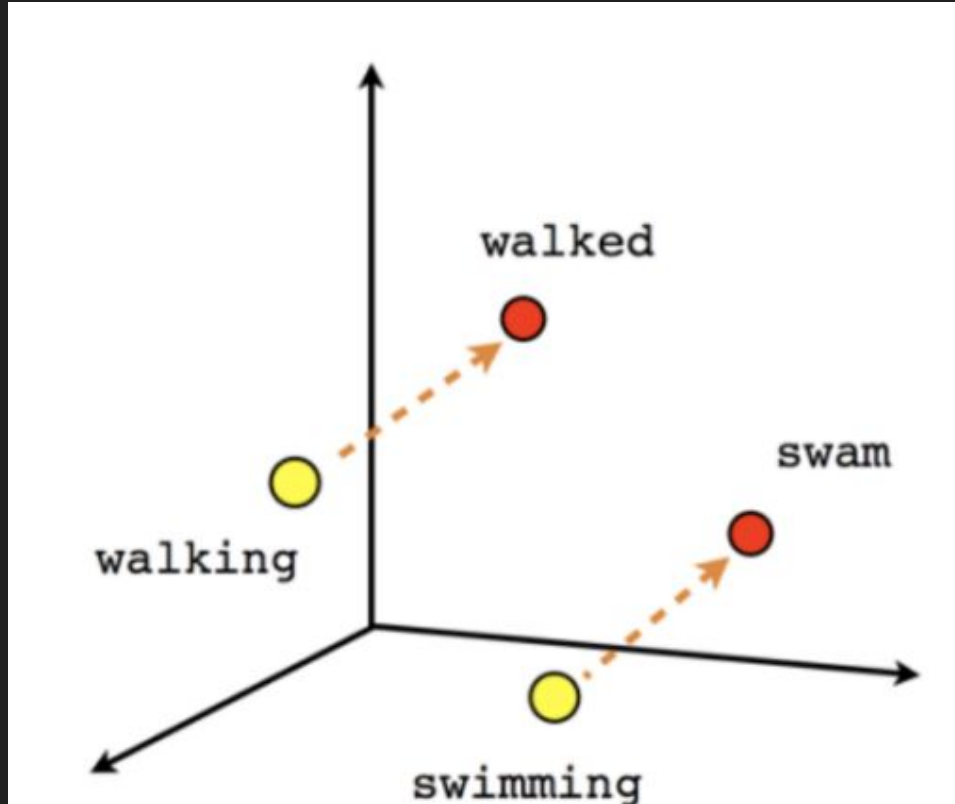
7. *eleutherodactylus*

# Should everything be a different token ?

Type	Occurrences	Rank
the	3789654	1st
he	2098762	2nd
[...]		
king	57897	1,356th
boy	56975	1,357th
[...]		
stringify	5	34,589th
[...]		
transducionalify	1	123,567th

01/01/1970	01/02/1970	01/03/1970	01/04/1970	01/05/1970
01/01/1971	01/02/1971	01/03/1971	01/04/1971	01/05/1971
01/01/1972	01/02/1972	01/03/1972	01/04/1972	01/05/1972
01/01/1973	01/02/1973	01/03/1973	01/04/1973	01/05/1973
01/01/1974	01/02/1974	01/03/1974	01/04/1974	01/05/1974
01/01/1975	01/02/1975	01/03/1975	01/04/1975	01/05/1975
01/01/1976	01/02/1976	01/03/1976	01/04/1976	01/05/1976
01/01/1977	01/02/1977	01/03/1977	01/04/1977	01/05/1977
01/01/1978	01/02/1978	01/03/1978	01/04/1978	01/05/1978
01/01/1979	01/02/1979	01/03/1979	01/04/1979	01/05/1979
01/01/1980	01/02/1980	01/03/1980	01/04/1980	01/05/1980
01/01/1981	01/02/1981	01/03/1981	01/04/1981	01/05/1981
01/01/1982	01/02/1982	01/03/1982	01/04/1982	01/05/1982
01/01/1983	01/02/1983	01/03/1983	01/04/1983	01/05/1983
01/01/1984	01/02/1984	01/03/1984	01/04/1984	01/05/1984
01/01/1985	01/02/1985	01/03/1985	01/04/1985	01/05/1985
01/01/1986	01/02/1986	01/03/1986	01/04/1986	01/05/1986
01/01/1987	01/02/1987	01/03/1987	01/04/1987	01/05/1987

# What about Morphology



כטובל ושרץ בידו

Light TAG



# You can write clever functions that tokenize

```
import re
dateReg=re.compile(r'(\d{2}[-\\]){2}\d{4}')
def tokenize(sentence):
    sentence = dateReg.sub("_DATE_",sentence)
    tokens = sentence.split()
    return tokens
tokenize(u"I went to the store on 12\\12\\1984 and 12\\24\\1976")

['I', 'went', 'to', 'the', 'store', 'on', '_DATE_', 'and', '_DATE_']
```

And you should, in light of the problem your solving

```
import re
dateReg=re.compile(r'(\d{2}[-\\]){2}\d{4}')
def tokenize(sentence):
    sentence = dateReg.sub("_DATE_",sentence)
    tokens = sentence.split()
    return tokens
tokenize(u"I went to the store on 12\\12\\1984 and 12\\24\\1976")

['I', 'went', 'to', 'the', 'store', 'on', '_DATE_', 'and', '_DATE_']
```

# What to consider ?

- Do I know what is relevant in the text ?
- Do I know what can be thrown away ?
- Does my model calculate interactions between tokens ?
- Each token is a feature, I need ~10 examples per feature ?
- Each token is a feature, will my model fit in memory ?
- Each token is a feature, will my model overfit ?
- **What representation of the data best suits my task ?**

# Resources

- [Stanford NLP Group](#)
- Snowball [Stemmer](#)
- [How Spacy Tokenizes](#)