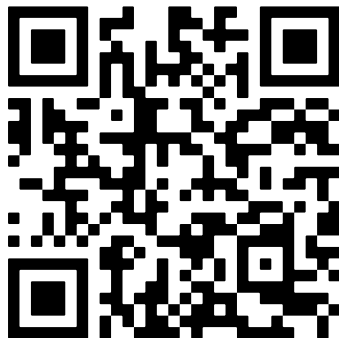


Introduction to the tokenization

Thomas Gerald

October 28, 2025



<https://thomas-gerald.fr/EcAuTAL/index.html>

NLP:

Natural Language Processing has as objective to perform task from natural language:

- Automatic Translation
- Dialogue systems
- Sentiment analysis
- Many other tasks

EN

Madame Roland was a French revolutionary, salonnière and writer.



FR

Madame Roland, était une révolutionnaire, une salonnière et un écrivain français.

NLP, why?

NLP:

Natural Language Processing has as objective to perform task from a natural language:

- Automatic Translation
- Dialogue systems
- Sentiment analysis
- Many other tasks

What brings you here today? SYSTEM

USER

Several things, I have chest pain, fever and sweating

And What else ? SYSTEM

USER

I currently have a cough, yellow sputum and difficulty breathing

Do you have shortness of breath? SYSTEM

USER

Yes

Since When ? SYSTEM

USER

I have shortness of breath since yesterday evening at 23 hours

OK. Do you have a heart condition? SYSTEM

NLP, why?

NLP:

Natural Language Processing has as objective to perform task from a natural language:

- Automatic Translation
- Dialogue systems
- **Sentiment analysis**
- Many other tasks

Stupid storm. No river thats so cool
for us tonight



NEGATIVE



POSITIVE

Applications

- Automatic translation
- Dialogue systems & question-answering
- Sentiment analysis
- Summarization
- Information retrieval
- Topics classification

Applications

- Automatic translation
- Dialogue systems & question-answering
- Sentiment analysis
- Summarization
- Information retrieval
- Topics classification

What kind of data

- **Structured** : Language is sequential
- **Discret** : word/subword/stem/lemma. . .
- **Continuous** : Spoken Language

NLP, why?

Applications

- Automatic translation
- Dialogue systems & question-answering
- Sentiment analysis
- Summarization
- Information retrieval
- Topics classification

What kind of data

- **Structured** : Language is sequential
- **Discret** : word/subword/stem/lemma. . .
- **Continuous** : Spoken Language

Consider only textual information

NLP, why?

Applications

- Automatic translation
- Dialogue systems & question-answering
- Sentiment analysis
- Summarization
- Information retrieval
- Topics classification

What kind of data

- **Structured** : Language is sequential
- **Discret** : word/subword/stem/lemma. . .
- **Continuous** : Spoken Language

Consider only textual information

Some difficulties

- High number of languages (> 6000)
- Ambiguity of the language
(same word with different meaning)
- Language evolution
- Noisy text (social network, text message)

Introduction: How to feed neural networks with text ?

To perform a task on natural language what do we need ?

- Transform to a digital format
- Apply algorithm, train a machine learning model...

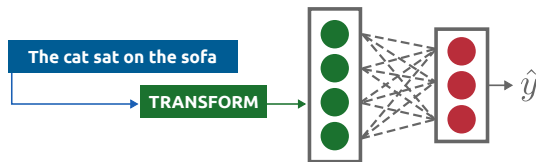
Introduction: How to feed neural networks with text ?

To perform a task on natural language what do we need ?

- Transform to a digital format
- Apply algorithm, train a machine learning model...

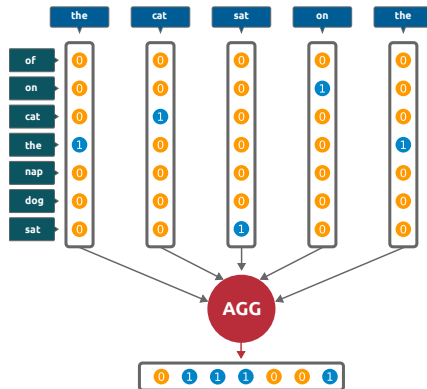
In neural networks

- Take vector(s) (tensor) as input
- Need to transform text to a vector



Bag-of-Word

- An index for each word
- Having a vector with the count of each word



Introduction: BoW and word embeddings

Bag-of-Word

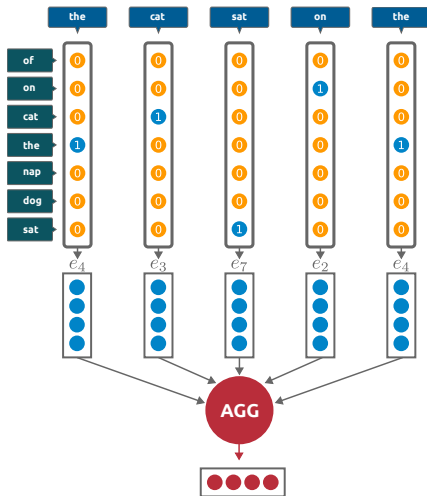
- A vector for each word
- Aggregate vector for text representation

Training vector (Word2vec)

Learn to predict missing word(s) in a context

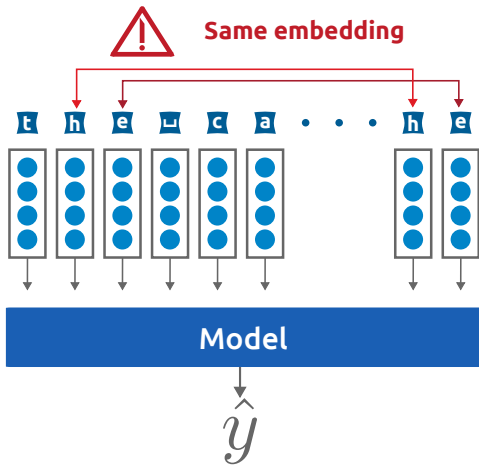
→ cbow, skip-gram

Not the topic of the lecture



Tokenization?

Tokenization: Splitting into characters ?



Splitting words into characters ?

- Very small vocabulary
- ⚠ words (characters) embeddings are meaningless

→ Let the model do all the job !!!

In literature ?

- CharacterBERT: learning sub-module to create word-embedding from characters (CNN)
"CharacterBERT: Reconciling ELMo and BERT for Word-Level Open-Vocabulary Representations From Characters", Boukkouri et al., ICCL 2020

Tokenization : Words ?

We consider as input a “word” (more a unit token)

A word units

- Need to have a semantic meaning ?
- Need to cover all words/meaningfull words ?

→ Choosing a word !!  What is a word? How to retrieve them?

Tokenization : Words ?

We consider as input a “word” (more a unit token)

A word units

- Need to have a semantic meaning ?
- Need to cover all words/meaningfull words ?

→ Choosing a word !!  What is a word? How to retrieve them?

A Vocabulary

- A finite set of “words”
- Serve as base input of algorithms (or models)

Tokenization : Words ?

We consider as input a “word” (more a unit token)

A word units

- Need to have a semantic meaning ?
- Need to cover all words/meaningfull words ?

→ Choosing a word !!  What is a word? How to retrieve them?

A Vocabulary

- A finite set of “words”
- Serve as base input of algorithms (or models)

Do choices of unit tokens impact algorithm ?

Tokenization : Words ?

We consider as input a “word” (more a unit token)

A word units

- Need to have a semantic meaning ?
- Need to cover all words/meaningfull words ?

→ Choosing a word !!  What is a word? How to retrieve them?

A Vocabulary

- A finite set of “words”
- Serve as base input of algorithms (or models)

Do choices of unit tokens impact algorithm ?

→  YES (Spoiler)

Tokenization: splitting into words

Splitting text with simple rules

- Using whitespace

Tokenization: splitting into words

Splitting text with simple rules

- Using whitespace
- Using punctuation

Tokenization: splitting into words

Splitting text with simple rules

- Using whitespace
- Using punctuation

Can use regular expressions on a corpus :

```
text = "Coil Optimization with Quadratic Constraints and Objectives"
word_regex = re.compile(r'(\b[^\s]+\b)')
words = word_regex.findall(text.lower())
# ['coil', 'optimization', 'with', 'quadratic', 'constraints',
#  'and', 'objectives']
```

Inclusive writting

- ☐ Les étudiants sont en grève
- ☐ Les étudiant(e)s sont en grève
- ☐ Les étudiant.e.s sont en grève
- ☐ Ils sont en grève
- ☐ Iels sont en grève

Inclusive writting

- ☐ Les étudiants sont en grève
- ☐ Les étudiant(e)s sont en grève
- ☐ Les étudiant.e.s sont en grève
- ☐ Ils sont en grève
- ☐ Iels sont en grève

Non-standard writting

- ☐ J'm la pizza
- ☐ J' <3 la pizza

Inclusive writting

- ☐ Les étudiants sont en grève
- ☐ Les étudiant(e)s sont en grève
- ☐ Les étudiant.e.s sont en grève
- ☐ Ils sont en grève
- ☐ Iels sont en grève

Non-standard writting

- ☐ J'm la pizza
- ☐ J' <3 la pizza

Typos

- ☐ I lik pizza
- ☐ Do you have informations

Tokenization: Source/document variation

Inclusive writting

- ☐ Les étudiants sont en grève
- ☐ Les étudiant(e)s sont en grève
- ☐ Les étudiant.e.s sont en grève
- ☐ Ils sont en grève
- ☐ Iels sont en grève

Specific domain

- ☐ **P-stalk ribosomes** act as master regulators of **cytokine-mediated** processes
- ☐ Global **Stellarator** Coil Optimization with Quadratic Constraints and Objectives

Non-standard writting

- ☐ J'm la pizza
- ☐ J' <3 la pizza

Typos



- ☐ I lik pizza
- ☐ Do you have informations

Many form for a “word”?

- Conjugation (entraînons, entraînez, entraînes)
- Plurals (stars, star)
- Different spellings (clef, clé)
- Characters case, accents, ...



Tokenization: Problem, unknown words ?

Vocabulary with “words”

- each token is meaningfull
-  Specific to considered resources/corpus !!!
-  Cannot cover all possibilities!!!

Tokenization: Problem, unknown words ?

Vocabulary with “words”



- each token is meaningful
-  Specific to considered resources/corpus !!!
-  Cannot cover all possibilities!!!

Handling out-of-vocabulary


- Use a specific token “unknown”

Tokenization: Problem, unknown words ?

Vocabulary with “words”

- each token is meaningful
-  Specific to considered resources/corpus !!!
-  Cannot cover all possibilities!!!

Handling out-of-vocabulary

- Use a specific token “unknown”
-  Same embedding for all unknown words...

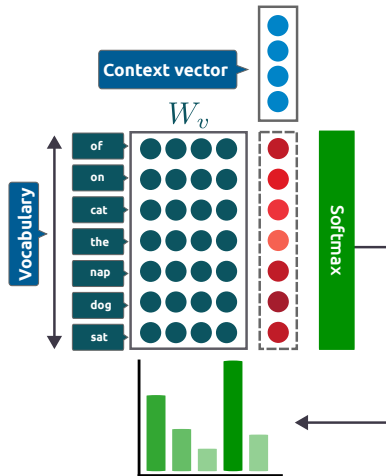
Tokenization: Size of vocabulary, Computational

Vocabulary size

- Must cover all form of the word (conjugation, plurals/singular, ...)
- Quickly increase the size of the vocabulary

Prediction of a words with a MLP

- Size of predictions matrix W_v , is $|V| \times n$
- $n \times |V|$ multiplication !!
- Probability distribution on very large vector ?



Tokenization: Problems, vocabulary's distribution

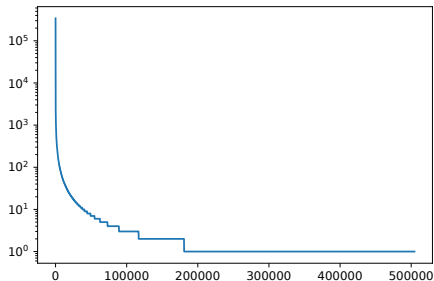


Figure 1: Number of occurrences for a part of wikipedia

Tokens/word distribution

- Follows a zipf's law
- Few tokens observed a lot of times
- A lot of tokens observed only few times

⁰<https://www.kaggle.com/datasets/ltxcmrdata/plain-text-wikipedia-202011>

Objective(s) of the lecture

What vocabulary we want to choose ?

- Depends on the source of documents
- Depends on the task
- Depends on the size of corpus (important in Machine Learning approaches)

Other possibilities

- Gather similar words together
- Textual normalisation (removing punctuation)
- ...

How?

Stemming & Lemmatization

Lematization and Tokenization : The course

en_core_web_md

[RELEASE DETAILS](#)

Latest: 3.8.0

English pipeline optimized for CPU. Components: tok2vec, tagger, parser, sender, ner, attribute_ruler, lemmatizer.

LANGUAGE	EN English
TYPE	CORE Vocabulary, syntax, entities, vectors
GENRE	WEB written text (blogs, news, comments)
SIZE	MD 31 MB
COMPONENTS [?]	tok2vec , tagger , parser , sender , attribute_ruler , lemmatizer , ner
PIPELINE [?]	tok2vec , tagger , parser , attribute_ruler , lemmatizer , ner

→ Tokenization, Stemming, Lemmatization,

Text pre-processing ?

- What is a token unit ?
- How I can normalize words ?
- ...

Lemmatization ? Stemming ?

From a word find a root form :

Lemmatization ? Stemming ?

From a word find a root form :

- Stemming (an abstract root, that potentially not exists in the language)

Lemmatization ? Stemming ?

From a word find a root form :

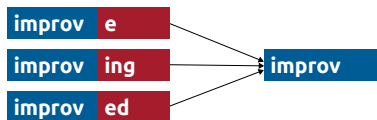
- Stemming (an abstract root, that potentially not exists in the language)
- Lemmatization (a root that is part of the language)

Lemmatization ? Stemming ?

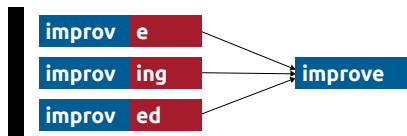
From a word find a root form :

- Stemming (an abstract root, that potentially not exists in the language)
- Lemmatization (a root that is part of the language)

Stemming



Lemmatization

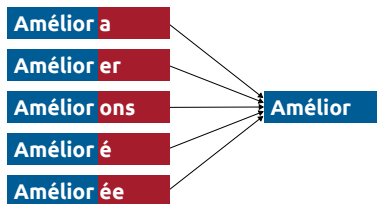


Lemmatization ? Stemming ?

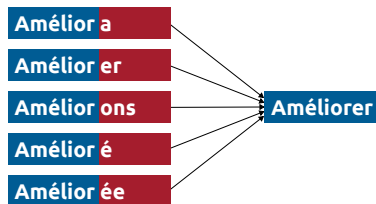
From a word find a root form :

- Stemming (an abstract root, that potentially not exists in the language)
- Lemmatization (a root that is part of the language)

Stemming



Lemmatization



⚠ In most languages, conjugation are different depending on tense and pronoun

Stemming :

The Stemming is the process of transforming a “word” into a stem

- The stem not necessarily exists in the language
- Can use a set of rules
- Rules depend on the language

Stemming :

The Stemming is the process of transforming a “word” into a stem

- The stem not necessarily exists in the language
- Can use a set of rules
- Rules depend on the language

The Porter stemmer

Introduced in 1980 in “An algorithm for suffix stripping” by M.F.Porter

Stemming :

The Stemming is the process of transforming a “word” into a stem

- The stem not necessarily exists in the language
- Can use a set of rules
- Rules depend on the language

The Porter stemmer

Introduced in 1980 in “An algorithm for suffix stripping” by M.F.Porter

- Sequential steps that remove suffix
- Each step is based on “language” rules
- Specific to english (Porter Stemmer)

Stemming-example : the porter stemmer

Some definition

- C one or many consecutive consonants
- V one or many consecutive vowels (A, E, I, U, or Y preceded by consonant)
- A measure m denoted by $C^*(V^+C^+)^mV^*$

Examples for different values of m :

- $m = 0 \rightarrow$ TREE, I, FOE, THE, BY or BE
- $m = 1 \rightarrow$ TREES, GRACE
- ...
- $m = 6 \rightarrow$ CAPITALIZATION

Stemming-example : the porter stemmer

Step-1a, transform the following suffixes:

- $SSES \rightarrow SS$
- $IES \rightarrow I$
- $S \rightarrow \emptyset$

Stemming-example : the porter stemmer

Step-1a, transform the following suffixes:

- SSES \rightarrow SS
- IES \rightarrow I
- S $\rightarrow \emptyset$

- ☐ losses \rightarrow loss
- ☐ abilities \rightarrow abiliti
- ☐ dogs \rightarrow dog

Stemming-example : the porter stemmer

Step-1b, transform the following suffixes:

- $(m > 0)$ EED \rightarrow SS
- A. $(.*V.*)$ ED $\rightarrow \emptyset$
- B. $(.*V.*)$ ING $\rightarrow \emptyset$

Stemming-example : the porter stemmer

Step-1b, transform the following suffixes:

• $(m > 0)$ EED \rightarrow SS

☐ agreed \rightarrow agree

• A. $(.*V.*)$ ED $\rightarrow \emptyset$

☐ troubled \rightarrow troubl

• B. $(.*V.*)$ ING $\rightarrow \emptyset$

☐ filing \rightarrow fil

If A or B, adding e at the end or removing double consonent

• AT \rightarrow ATE

• BL \rightarrow BLE

• IZ \rightarrow IZE

• ...

Stemming-example : the porter stemmer

Step-1b, transform the following suffixes:

• $(m > 0)$ EED \rightarrow SS

• A. $(.*V.*)$ ED $\rightarrow \emptyset$

• B. $(.*V.*)$ ING $\rightarrow \emptyset$

☐ agreed \rightarrow agree

☐ troubled \rightarrow troubl

☐ filing \rightarrow fil

If A or B, adding e at the end or removing double consonant

• AT \rightarrow ATE

• BL \rightarrow BLE

• IZ \rightarrow IZE

• ...

☐ confl~~at~~ \rightarrow conflate

☐ troubl~~e~~ \rightarrow trouble

☐ siz~~e~~ \rightarrow size

☐ fil~~e~~ \rightarrow file

And other rules \rightarrow Step 1 deals with plurals and past participles ...

Stemming-example : the porter stemmer

Step-2, a set of rules for termination:

- $(m > 0)$ IZATION \rightarrow IZE
- $(m > 0)$ FULNESS \rightarrow FUL
- ...

Stemming-example : the porter stemmer

Step-2, a set of rules for termination:

- $(m > 0)$ IZATION \rightarrow IZE ☐ formalization \rightarrow formalize
- $(m > 0)$ FULNESS \rightarrow FUL ☐ hopeful~~ness~~ \rightarrow hopefull
- ... ☐ ...

Step-3, similar to previous step but refining the stem:

- $(m > 0)$ ALIZE \rightarrow AL
- $(m > 0)$ FUL $\rightarrow \emptyset$
- ...

Stemming-example : the porter stemmer

Step-2, a set of rules for termination:

- $(m > 0)$ IZATION \rightarrow IZE ☐ formalization \rightarrow formalize
- $(m > 0)$ FULNESS \rightarrow FUL ☐ hopeful~~ness~~ \rightarrow hopefull
- ... ☐ ...

Step-3, similar to previous step but refining the stem:

- $(m > 0)$ ALIZE \rightarrow AL ☐ formalize \rightarrow formal
- $(m > 0)$ FUL $\rightarrow \emptyset$ ☐ hopeful \rightarrow hope
- ... ☐ ...

Step-4, Re-refining the stem:

- $(m > 1)$ AL $\rightarrow \emptyset$
- $(m > 1)$ OU $\rightarrow \emptyset$
- ...

Stemming-example : the porter stemmer

Step-2, a set of rules for termination:

- $(m > 0)$ IZATION \rightarrow IZE ☐ formalization \rightarrow formalize
- $(m > 0)$ FULNESS \rightarrow FUL ☐ hopeful~~ness~~ \rightarrow hopefull
- ... ☐ ...

Step-3, similar to previous step but refining the stem:

- $(m > 0)$ ALIZE \rightarrow AL ☐ formalize \rightarrow formal
- $(m > 0)$ FUL $\rightarrow \emptyset$ ☐ hope~~ful~~ \rightarrow hope
- ... ☐ ...

Step-4, Re-refining the stem:

- $(m > 1)$ AL $\rightarrow \emptyset$ ☐ formal \rightarrow form
- $(m > 1)$ OU $\rightarrow \emptyset$ ☐ generou \rightarrow gener
- ... ☐ ...

Stemming-example : the porter stemmer

Other steps ?

- Remove the “e” suffix in some case
- Remove in some cases double consonant

Rule based stemmer :

⚠ Heavily based upon an extensive knowledge of the language to infer the rules

Does two stem always come from semantic close words ?

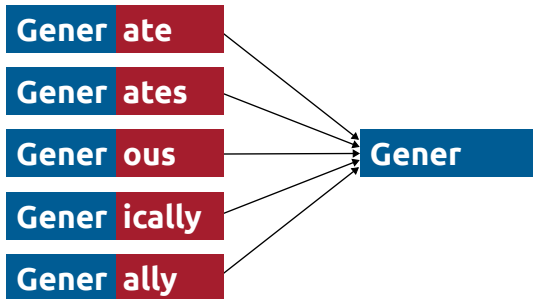
Stemming

An example :

What are the stem of “generous” and “general”

- generous → gener (step1 and 4)
- general → gener (step 4)

Such representation can leads to ambiguity



Does two stem always come from semantic close words ? ⚠ No

Lemmatization:

The Lemmatization is the process of transforming a “word” into a lemma

Lemmatization:

The Lemmatization is the process of transforming a “word” into a lemma

- Contrary to stem lemma is a word existing in the language

Lemmatization:

The Lemmatization is the process of transforming a “word” into a lemma

- Contrary to stem lemma is a word existing in the language
- Lemmatization needs access to a dictionary

Lemmatization:

The Lemmatization is the process of transforming a “word” into a lemma

- Contrary to stem lemma is a word existing in the language
- Lemmatization needs access to a dictionary
- Finding the closest form of text word to dictionary word

Lemmatization, a first example

Example:

```
import spacy
nlp = spacy.load("en_core_web_sm")
text_a = "He is thirty years old"
text_b = "We still are champions"
print(f'Lemmatization A : {[w.lemma_ for w in nlp(text_a)]}')
print(f'Lemmatization B : {[w.lemma_ for w in nlp(text_b)]}')
```

Lemmatization, a first example

Example:

```
import spacy
nlp = spacy.load("en_core_web_sm")
text_a = "He is thirty years old"
text_b = "We still are champions"
print(f'Lemmatization A : {[w.lemma_ for w in nlp(text_a)]}')
print(f'Lemmatization B : {[w.lemma_ for w in nlp(text_b)]}')
```

Results?

Lemmatization, a first example

Example:

```
import spacy
nlp = spacy.load("en_core_web_sm")
text_a = "He is thirty years old"
text_b = "We still are champions"
print(f'Lemmatization A : {[w.lemma_ for w in nlp(text_a)]}')
print(f'Lemmatization B : {[w.lemma_ for w in nlp(text_b)]}')
```

Results?

```
Lemmatization A : ['he', 'be', 'thirty', 'year', 'old']
Lemmatization B : ['we', 'still', 'be', 'champion']
```

Lemmatization, a first example

Example:

```
import spacy
nlp = spacy.load("en_core_web_sm")
text_a = "He is thirty years old"
text_b = "We still are champions"
print(f'Lemmatization A : {[w.lemma_ for w in nlp(text_a)]}')
print(f'Lemmatization B : {[w.lemma_ for w in nlp(text_b)]}')
```

Results?

```
Lemmatization A : ['he', 'be', 'thirty', 'year', 'old']
Lemmatization B : ['we', 'still', 'be', 'champion']
```

→  "is/are" are set to "be", the infinitive

Lemmatization, how does it work?

Algorithm for lemmatization

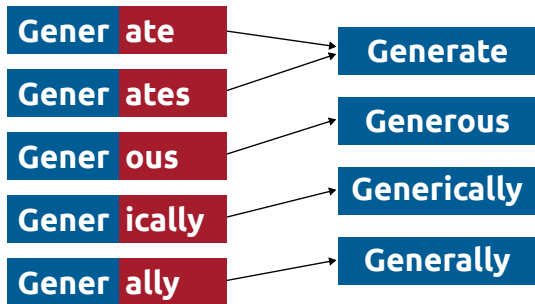
- Rule based approaches
- Dictionary based approaches
- Model based approaches
- A mix of previous approaches

NLTK lemmatizer

- Use synset (set of synonyms)
- Apply rules (removes plurals marks)
- Find the corresponding word in the synset

→ An alternative exists with pos-tag information

Lemmatization



On which kind of tasks

- Information retrieval
- Classification
- Sentiment analysis

→ Particularly when real data is missing

Lemmatization: remaining issues

Lemmatization can depends from context

- I rule on the world
- The rule specify it
- rule is a verb (to rule)
- rule is the noun (a rule)

→ A lemma can be a tuple (word_form, PoS)

 Needs context !!!

→ Some approaches use the POS additionally to the token !!!

Not always easy with small context

Time flies like an arrow

What is the NLP standard input?

What is the NLP standard input?

→ Mostly subwords units

How to build it?

→ Automatically with statistical methods

Using subword units

Tokenization: Splitting into subwords units

- Splitting words into subwords
- Subwords with "semantic" meaning
- Suffix/prefix split

How to ?

- Use a database (dictionary)
- Do it automatically ? → Statistical tokenization

learn **ing**

be **ing**

•
•
•

eat **ing**

sleep **ing**

Tokenization: Splitting into subwords units

**1574 words that
start with "anti"**

anti oxydant

anti biotic

•
•
•

anti viral

anti venoms

Automatic tokenizer ?

Consider a text corpus

- common words are frequent
- common subwords are frequent

→ Algorithms based on the frequency of subwords ?

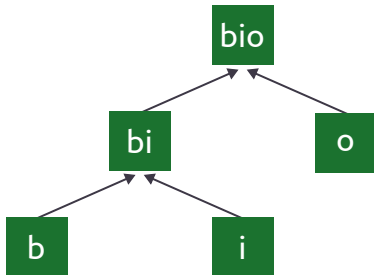
A naive algorithm:

1. Splitting words in all possible subwords
2. Ordering subwords by frequency
3. Add to the vocabulary the top-k most frequent subwords

Tokenization: BPE Algorithm

The Byte Pair Encoding Algorithm

Learn merging rules from characters to subwords



→ “A new algorithm for data compression”, P. Gage, C user Journal, 1994

→ “Neural Machine Translation of Rare Words with Subword Units”, R. Seenrich et al., ACL 2016

Neural Machine Translation of Rare Words with Subword Units

Rico Sennrich and Barry Haddow and Alexandra Birch

School of Informatics, University of Edinburgh

{rico.sennrich,a.birch}@ed.ac.uk,bhaddow@inf.ed.ac.uk

Abstract

Neural machine translation (NMT) models typically operate with a fixed vocabulary, but translation is an open-vocabulary problem. Previous work addresses the translation of out-of-vocabulary words by backing off to a dictionary. In this paper, we introduce a simpler and more effective approach, making the NMT model capable of open-vocabulary translation by encoding rare and unknown words as sequences of subword units. This is based on the intuition that various word classes are translatable via smaller units than words, for instance names (via character copying or transliteration), compounds (via com-

lem, and especially for languages with productive word formation processes such as agglutination and compounding, translation models require mechanisms that go below the word level. As an example, consider compounds such as the German *Abwasser|behandlungs|anlage* ‘sewage water treatment plant’, for which a segmented, variable-length representation is intuitively more appealing than encoding the word as a fixed-length vector.

For word-level NMT models, the translation of out-of-vocabulary words has been addressed through a back-off to a dictionary look-up (Jean et al., 2015; Luong et al., 2015b). We note that such techniques make assumptions that often do not hold true in practice. For instance, there is not always a 1-to-1 correspondence between source and

Algorithm 1 BPE Tokenization

- 1: **Input:** Word w ; Vocabulary \mathcal{V} ; Merge rules \mathcal{M}
- 2: **Output:** Tokenized word \mathcal{W}

Algorithm 1 BPE Tokenization

- 1: **Input:** Word w ; Vocabulary \mathcal{V} ; Merge rules \mathcal{M}
- 2: **Output:** Tokenized word \mathcal{W}
- 3: $\mathcal{W} \leftarrow$ split w into symbols $\in \mathcal{V}$

Algorithm 1 BPE Tokenization

- 1: **Input:** Word w ; Vocabulary \mathcal{V} ; Merge rules \mathcal{M}
- 2: **Output:** Tokenized word \mathcal{W}
- 3: $\mathcal{W} \leftarrow$ split w into symbols $\in \mathcal{V}$
- 4: **for** $(p_1, p_2, v) \in \mathcal{M}$ **do**

Algorithm 1 BPE Tokenization

- 1: **Input:** Word w ; Vocabulary \mathcal{V} ; Merge rules \mathcal{M}
- 2: **Output:** Tokenized word \mathcal{W}
- 3: $\mathcal{W} \leftarrow$ split w into symbols $\in \mathcal{V}$
- 4: **for** $(p_1, p_2, v) \in \mathcal{M}$ **do**
- 5: $i \leftarrow 0$

Algorithm 1 BPE Tokenization

- 1: **Input:** Word w ; Vocabulary \mathcal{V} ; Merge rules \mathcal{M}
- 2: **Output:** Tokenized word \mathcal{W}
- 3: $\mathcal{W} \leftarrow$ split w into symbols $\in \mathcal{V}$
- 4: **for** $(p_1, p_2, v) \in \mathcal{M}$ **do**
- 5: $i \leftarrow 0$
- 6: **while** $i < \text{len}(\mathcal{W})$ **do**

Algorithm 1 BPE Tokenization

- 1: **Input:** Word w ; Vocabulary \mathcal{V} ; Merge rules \mathcal{M}
- 2: **Output:** Tokenized word \mathcal{W}
- 3: $\mathcal{W} \leftarrow$ split w into symbols $\in \mathcal{V}$
- 4: **for** $(p_1, p_2, v) \in \mathcal{M}$ **do**
- 5: $i \leftarrow 0$
- 6: **while** $i < \text{len}(\mathcal{W})$ **do**
- 7: **if** $\mathcal{W}_i = p_1 \wedge \mathcal{W}_{i+1} = p_2$ **then**

Algorithm 1 BPE Tokenization

```
1: Input: Word  $w$ ; Vocabulary  $\mathcal{V}$ ; Merge rules  $\mathcal{M}$ 
2: Output: Tokenized word  $\mathcal{W}$ 
3:  $\mathcal{W} \leftarrow$  split  $w$  into symbols  $\in \mathcal{V}$ 
4: for  $(p_1, p_2, v) \in \mathcal{M}$  do
5:    $i \leftarrow 0$ 
6:   while  $i < \text{len}(\mathcal{W})$  do
7:     if  $\mathcal{W}_i = p_1 \wedge \mathcal{W}_{i+1} = p_2$  then
8:        $\mathcal{W}_i \leftarrow v$ 
9:        $\mathcal{W}_{i+1:\text{len}(\mathcal{W})} \leftarrow \mathcal{W}_{i+2:\text{len}(\mathcal{W})}$ 
```


Tokenization: BPE Algorithm

Algorithm 1 BPE Tokenization

```
1: Input: Word  $w$ ; Vocabulary  $\mathcal{V}$ ; Merge rules  $\mathcal{M}$ 
2: Output: Tokenized word  $\mathcal{W}$ 
3:  $\mathcal{W} \leftarrow$  split  $w$  into symbols  $\in \mathcal{V}$ 
4: for  $(p_1, p_2, v) \in \mathcal{M}$  do
5:    $i \leftarrow 0$ 
6:   while  $i < \text{len}(W)$  do
7:     if  $W_i = p_1 \wedge W_{i+1} = p_2$  then
8:        $W_i \leftarrow v$ 
9:        $W_{i+1:\text{len}(W)} \leftarrow W_{i+2:\text{len}(W)}$ 
10:    else
11:       $i \leftarrow i + 1$ 
12:    end if
13:  end while
14: end for
```

Tokenization: BPE Algorithm

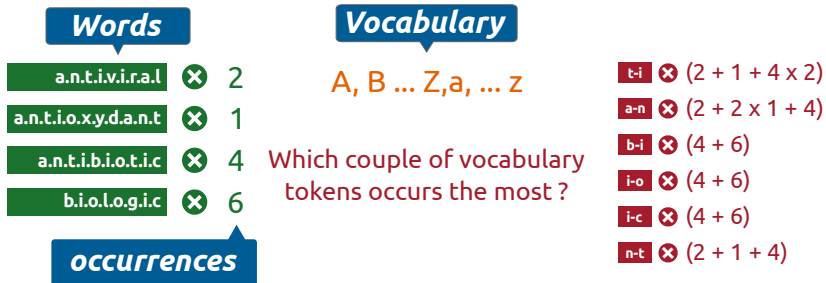
Algorithm 1 BPE Tokenization

```
1: Input: Word  $w$ ; Vocabulary  $\mathcal{V}$ ; Merge rules  $\mathcal{M}$ 
2: Output: Tokenized word  $\mathcal{W}$ 
3:  $\mathcal{W} \leftarrow$  split  $w$  into symbols  $\in \mathcal{V}$ 
4: for  $(p_1, p_2, v) \in \mathcal{M}$  do
5:    $i \leftarrow 0$ 
6:   while  $i < \text{len}(W)$  do
7:     if  $W_i = p_1 \wedge W_{i+1} = p_2$  then
8:        $W_i \leftarrow v$ 
9:        $W_{i+1:\text{len}(W)} \leftarrow W_{i+2:\text{len}(W)}$ 
10:    else
11:       $i \leftarrow i + 1$ 
12:    end if
13:  end while
14: end for
15: return  $\mathcal{W}$ 
```

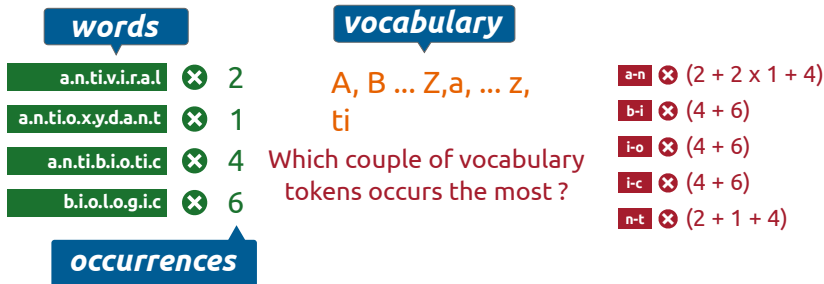
Merge rules

t-i	a.n.t.i.o.x.y.d.a.n.t
a-n	a.n.ti.o.x.y.d.a.n.t
an-ti	an.ti.o.x.y.d.an.t
bi-o	an.ti.o.x.y.d.an.t

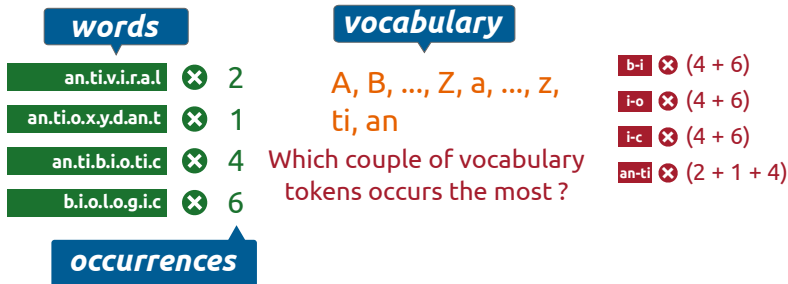
Tokenization: BPE Algorithm



Tokenization: BPE Algorithm



Tokenization: BPE Algorithm



Tokenization: BPE Algorithm

<i>words</i>		
an.ti.v.i.r.a.l	✕	2
an.ti.o.x.y.d.an.t	✕	1
an.ti.bi.o.ti.c	✕	4
bi.o.l.o.g.i.c	✕	6
<i>occurrences</i>		

vocabulary

A, B, ..., Z, a, ..., z,
ti, an, bi

Which couple of vocabulary
tokens occurs the most ?

bi-o ✕ (4 + 6)

i-c ✕ (4 + 6)

an-ti ✕ (2 + 1 + 4)



Tokenization: BPE Algorithm

words		
an.ti.v.i.r.a.l	✕	2
an.ti.o.x.y.d.an.t	✕	1
an.ti.bio.ti.c	✕	4
bio.l.o.g.i.c	✕	6
occurrences		

vocabulary

A, B, ..., Z, a, ..., z,
ti, an, bi, bio

Which couple of vocabulary
tokens occurs the most ?

i-c ✕ (4 + 6)

an-ti ✕ (2 + 1 + 4)



Principle:

Initialize vocabulary with characters

1. Tokenize the text (with current rules)
2. Finding tokens that appears the most consecutively
3. Add a merge rule between the two tokens
4. Repeat the procedure

Algorithm 2 Training BPE (big picture)

- 1: **Input:** Corpus C ; Vocabulary \mathcal{V} ; size S
- 2: **Output:** \mathcal{V}, \mathcal{M}

Principle:

Initialize vocabulary with characters

1. Tokenize the text (with current rules)
2. Finding tokens that appears the most consecutively
3. Add a merge rule between the two tokens
4. Repeat the procedure

Algorithm 2 Training BPE (big picture)

- 1: **Input:** Corpus C ; Vocabulary \mathcal{V} ; size S
- 2: **Output:** \mathcal{V}, \mathcal{M}
- 3: $W \leftarrow \text{tokenize}(C)$
- 4: $\mathcal{M} \leftarrow \emptyset$

Tokenization: BPE Algorithm

Principle:

Initialize vocabulary with characters

1. Tokenize the text (with current rules)
2. Finding tokens that appears the most consecutively
3. Add a merge rule between the two tokens
4. Repeat the procedure

Algorithm 2 Training BPE (big picture)

- 1: **Input:** Corpus C ; Vocabulary \mathcal{V} ; size S
- 2: **Output:** \mathcal{V}, \mathcal{M}
- 3: $W \leftarrow \text{tokenize}(C)$
- 4: $\mathcal{M} \leftarrow \emptyset$
- 5: **while** $|\mathcal{V}| < S$ **do**

Tokenization: BPE Algorithm

Principle:

Initialize vocabulary with characters

1. Tokenize the text (with current rules)
2. Finding tokens that appears the most consecutively
3. Add a merge rule between the two tokens
4. Repeat the procedure

Algorithm 2 Training BPE (big picture)

- 1: **Input:** Corpus C ; Vocabulary \mathcal{V} ; size S
- 2: **Output:** \mathcal{V}, \mathcal{M}
- 3: $W \leftarrow \text{tokenize}(C)$
- 4: $\mathcal{M} \leftarrow \emptyset$
- 5: **while** $|\mathcal{V}| < S$ **do**
- 6: $F \leftarrow \text{compute_pairs_freq}(W)$
- 7: $P \leftarrow \text{best_pairs}(F)$

Tokenization: BPE Algorithm

Principle:

Initialize vocabulary with characters

1. Tokenize the text (with current rules)
2. Finding tokens that appears the most consecutively
3. Add a merge rule between the two tokens
4. Repeat the procedure

Algorithm 2 Training BPE (big picture)

- 1: **Input:** Corpus C ; Vocabulary \mathcal{V} ; size S
- 2: **Output:** \mathcal{V}, \mathcal{M}
- 3: $W \leftarrow \text{tokenize}(C)$
- 4: $\mathcal{M} \leftarrow \emptyset$
- 5: **while** $|\mathcal{V}| < S$ **do**
- 6: $F \leftarrow \text{compute_pairs_freq}(W)$
- 7: $P \leftarrow \text{best_pairs}(F)$
- 8: $\mathcal{V} \leftarrow \mathcal{V} \cup (P_0.P_1)$
- 9: $m \leftarrow (P_0, P_1, \{j | V_j = P_0.P_1\})$
- 10: $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$

Tokenization: BPE Algorithm

Principle:

Initialize vocabulary with characters

1. Tokenize the text (with current rules)
2. Finding tokens that appears the most consecutively
3. Add a merge rule between the two tokens
4. Repeat the procedure

Algorithm 2 Training BPE (big picture)

- 1: **Input:** Corpus C ; Vocabulary \mathcal{V} ; size S
- 2: **Output:** \mathcal{V}, \mathcal{M}
- 3: $W \leftarrow \text{tokenize}(C)$
- 4: $\mathcal{M} \leftarrow \emptyset$
- 5: **while** $|\mathcal{V}| < S$ **do**
- 6: $F \leftarrow \text{compute_pairs_freq}(W)$
- 7: $P \leftarrow \text{best_pairs}(F)$
- 8: $\mathcal{V} \leftarrow \mathcal{V} \cup (P_0.P_1)$
- 9: $m \leftarrow (P_0, P_1, \{j | V_j = P_0.P_1\})$
- 10: $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
- 11: $W \leftarrow \text{merge_pair}(m, W)$
- 12: **end while**

Tokenization: BPE Algorithm

Principle:

Initialize vocabulary with characters

1. Tokenize the text (with current rules)
2. Finding tokens that appears the most consecutively
3. Add a merge rule between the two tokens
4. Repeat the procedure

Algorithm 2 Training BPE (big picture)

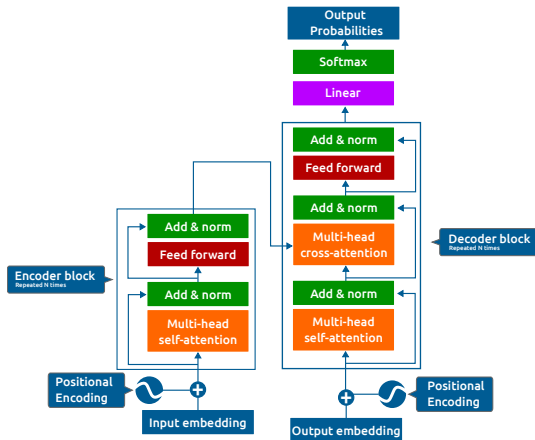
```
1: Input: Corpus  $C$ ; Vocabulary  $\mathcal{V}$ ; size  $S$ 
2: Output:  $\mathcal{V}, \mathcal{M}$ 
3:  $W \leftarrow \text{tokenize}(C)$ 
4:  $\mathcal{M} \leftarrow \emptyset$ 
5: while  $|\mathcal{V}| < S$  do
6:    $F \leftarrow \text{compute\_pairs\_freq}(W)$ 
7:    $P \leftarrow \text{best\_pairs}(F)$ 
8:    $\mathcal{V} \leftarrow \mathcal{V} \cup (P_0.P_1)$ 
9:    $m \leftarrow (P_0, P_1, \{j | V_j = P_0.P_1\})$ 
10:   $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$ 
11:   $W \leftarrow \text{merge\_pair}(m, W)$ 
12: end while
13: return  $\mathcal{V}, \mathcal{M}$ 
```

Tokenization: BPE Algorithm

→ Used in a lot of transformer models

Tokenization: BPE Algorithm

→ Used in a lot of transformer models



What are the models ?

- GPT, GPT2 (on byte level)
- BART
- RoBERTa
- ...

When to stop learning merging rules?

When to stop learning merging rules?

- Threshold on the size of the vocabulary

When to stop learning merging rules?

- Threshold on the size of the vocabulary
- Empirical size (depending on the languages/tasks/models)

When to stop learning merging rules?

- Threshold on the size of the vocabulary
- Empirical size (depending on the languages/tasks/models)

→ 32 000, 64 000 mostly (threshold between size and token expressivness)

Tokenization: Wordpiece Algorithm

Principle (similar to BPE)

- Initialize vocabulary with characters
- Compute for each token pairs t_1, t_2 $score(t_1, t_2) = \frac{p(t_1, t_2)}{p(t_1) \times p(t_2)}$
- Add a merge rule for couple of tokens with highest score

When to stop ?

- Threshold on the size of the vocabulary
- Threshold on the frequency

What are the models?

- BERT
- Most of transformer based on BERT

Unigram

- Different tokenization
- From large collection of subwords, reduce the vocabulary

S u b w o r d s (_ means spaces)	Vocabulary id sequence
_Hell/o/_world	13586 137 255
_H/ello/_world	320 7363 255
_He/llo/_world	579 10115 255
_/He/l/l/o/_world	7 18085 356 356 137 255
H/e/l/l/o//world	320 585 356 137 7 12295

Table 1: Multiple subword sequences encoding the same sentence “Hello World”

→ “Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates”,
Taku Kudo, ACL 2018

Hypothesis : Training with different tokenization would help (regularization)

Tokenization: Unigrams

Unigram tokenization

Considering an input sentence X and $S(X)$ all the possible decomposition of X into subwords :

- The probability of the sequence $x = x_1, x_2, \dots, x_m \in S(x)$ is given by $P(x) = \prod_{i=1}^m p(x_i)$
- $p(x_i)$ is the probability of a subword in the corpus

For the sequence X the segmentation is

$$x^* = \arg \max_{x \in S(X)} P(x)$$

→ It is also possible to sample $x \sim S(x)$

Building vocabulary

1. Start from a large vocabulary
2. Compute how removing tokens affect the likelihood over the datasets
3. Remove tokens that affect (drop in likelihood) the most the model

Unigrams summary

- Different segmentation possible for an example
- Sampling most likely split

Do models use this approach ?

- T5
- ...

⚠ Same sentence can have a different tokenization

Tokenization: Lossless Tokenization

Example :

Let consider the following vocabulary :

Voc	id
the	1
cat	2
is	3
sit	4
t	5
ing	6
on	7
couch	8

What is the tokenization of (considering BPE):

“The cat is sitting on the couch”



the-cat-is-sit-t-ing-on-the-couch ([1, 2, 3, 4, 5, 6, 7, 1, 8])

How to detokenize ?



“thecatissittingonthecouch”

⚠ We loss some information ont the space

Tokenization: Lossless Tokenization

Detokenization ?

1. Add a special character for inner-word tokens
→ cannot encode multiple space
2. Add a special token for space

With **wordpiece** implementation

→ Using the token “*##token*” meaning it is an “inside word” token

With **sentencepiece** implementation

→ Using the token “_” for space char

Tokenization: Some remaining issues

anti oxydant

anti biotic

•
•
•

anti viral

anti venoms

*1574 words that
start with "anti"*

Not always meaningfull



anti cs

Tokenization: Some remaining issues

anti oxydant

anti biotic

•
•
•

anti viral

anti venoms

*1574 words that
start with "anti"*

Not always meaningfull



anti cs

- Tokens are not always relevant

Tokenization: Some remaining issues

anti oxydant

anti biotic

•
•
•

anti viral

anti venoms

*1574 words that
start with "anti"*

Not always meaningfull



anti cs

- Tokens are not always relevant
- Mostly depends on the size of the vocabulary or the training corpus

Conclusion:

- Principle and problems of tokenization
- Different approaches (BPE, Wordpiece, Unigram)

Conclusion:

- Principle and problems of tokenization
- Different approaches (BPE, Wordpiece, Unigram)

Exercise ?

- **Build your lemmatizer:** Construct the lemmatizer for french (you also can create one for your favorite language)
- **Implement the BPE algorithm:** Implement the BPE tokenizer