Atelier 6 : Language Modeling

1. N-Gram

Le package nltk.lm sera utilisé pour l'implementation du modèle de langage N-gram. La documentation est disponible ICI. Egalement, des exemples sont disponibles ICI

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-
packages (3.9.1)
Requirement already satisfied: click in
/usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in
/usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-
packages (from nltk) (4.66.6)
```

On va considerer le corpus de reuters pour entrainer le. Pour ce faire, choisisez une categorie parmi celles disponibles dans le corpus reuters afin de raliser l'entrainement du modèle.

```
# Recuperation du corpus
import nltk
nltk.download('all')
from nltk.corpus import reuters

job_news=reuters.raw(reuters.fileids("jobs"))
```

```
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package alpino to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/alpino.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger_eng to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
```

```
[nltk_data]     |       taggers/averaged_perceptron_tagger_eng.zip.
[nltk_data]     | Downloading package averaged_perceptron_tagger_ru to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping
[nltk_data]     |       taggers/averaged_perceptron_tagger_ru.zip.
[nltk_data]     | Downloading package averaged_perceptron_tagger_rus to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping
[nltk_data]     |       taggers/averaged_perceptron_tagger_rus.zip.
[nltk_data]     | Downloading package basque_grammars to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping grammars/basque_grammars.zip.
[nltk_data]     | Downloading package bcp47 to /root/nltk_data...
[nltk_data]     | Downloading package biocreative_ppi to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/biocreative_ppi.zip.
[nltk_data]     | Downloading package bllip_wsj_no_aux to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping models/bllip_wsj_no_aux.zip.
[nltk_data]     | Downloading package book_grammars to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping grammars/book_grammars.zip.
[nltk_data]     | Downloading package brown to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/brown.zip.
[nltk_data]     | Downloading package brown_tei to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/brown_tei.zip.
[nltk_data]     | Downloading package cess_cat to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/cess_cat.zip.
[nltk_data]     | Downloading package cess_esp to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/cess_esp.zip.
[nltk_data]     | Downloading package chat80 to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/chat80.zip.
[nltk_data]     | Downloading package city_database to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/city_database.zip.
[nltk_data]     | Downloading package cmudict to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/cmudict.zip.
[nltk_data]     | Downloading package comparative_sentences to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/comparative_sentences.zip.
[nltk_data]     | Downloading package comtrans to /root/nltk_data...
[nltk_data]     | Downloading package conll2000 to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/conll2000.zip.
[nltk_data]     | Downloading package conll2002 to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/conll2002.zip.
[nltk_data]     | Downloading package conll2007 to /root/nltk_data...
[nltk_data]     | Downloading package crubadan to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/crubadan.zip.
[nltk_data]     | Downloading package dependency_treebank to
```

```
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/dependency_treebank.zip.
[nltk_data]     | Downloading package dolch to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/dolch.zip.
[nltk_data]     | Downloading package europarl_raw to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/europarl_raw.zip.
[nltk_data]     | Downloading package extended_omw to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     | Downloading package floresta to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/floresta.zip.
[nltk_data]     | Downloading package framenet_v15 to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/framenet_v15.zip.
[nltk_data]     | Downloading package framenet_v17 to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/framenet_v17.zip.
[nltk_data]     | Downloading package gazetteers to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/gazetteers.zip.
[nltk_data]     | Downloading package genesis to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/genesis.zip.
[nltk_data]     | Downloading package gutenberg to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/gutenberg.zip.
[nltk_data]     | Downloading package ieer to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/ieer.zip.
[nltk_data]     | Downloading package inaugural to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/inaugural.zip.
[nltk_data]     | Downloading package indian to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/indian.zip.
[nltk_data]     | Downloading package jeita to /root/nltk_data...
[nltk_data]     | Downloading package kimmo to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/kimmo.zip.
[nltk_data]     | Downloading package knbc to /root/nltk_data...
[nltk_data]     | Downloading package large_grammars to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping grammars/large_grammars.zip.
[nltk_data]     | Downloading package lin_thesaurus to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/lin_thesaurus.zip.
[nltk_data]     | Downloading package mac_morpho to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/mac_morpho.zip.
[nltk_data]     | Downloading package machado to /root/nltk_data...
[nltk_data]     | Downloading package masc_tagged to /root/nltk_data...
[nltk_data]     | Downloading package maxent_ne_chunker to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data]     | Downloading package maxent_ne_chunker_tab to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping chunkers/maxent_ne_chunker_tab.zip.
```

```
[nltk_data]     | Downloading package maxent_treebank_pos_tagger to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping taggers/maxent_treebank_pos_tagger.zip.
[nltk_data]     | Downloading package maxent_treebank_pos_tagger_tab to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping
[nltk_data]     |       taggers/maxent_treebank_pos_tagger_tab.zip.
[nltk_data]     | Downloading package moses_sample to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping models/moses_sample.zip.
[nltk_data]     | Downloading package movie_reviews to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/movie_reviews.zip.
[nltk_data]     | Downloading package mte_teip5 to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/mte_teip5.zip.
[nltk_data]     | Downloading package mwa_ppdb to /root/nltk_data...
[nltk_data]     |   Unzipping misc/mwa_ppdb.zip.
[nltk_data]     | Downloading package names to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/names.zip.
[nltk_data]     | Downloading package nombank.1.0 to /root/nltk_data...
[nltk_data]     | Downloading package nonbreaking_prefixes to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/nonbreaking_prefixes.zip.
[nltk_data]     | Downloading package nps_chat to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/nps_chat.zip.
[nltk_data]     | Downloading package omw to /root/nltk_data...
[nltk_data]     | Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]     | Downloading package opinion_lexicon to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/opinion_lexicon.zip.
[nltk_data]     | Downloading package panlex_swadesh to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     | Downloading package paradigms to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/paradigms.zip.
[nltk_data]     | Downloading package pe08 to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/pe08.zip.
[nltk_data]     | Downloading package perluniprops to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping misc/perluniprops.zip.
[nltk_data]     | Downloading package pil to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/pil.zip.
[nltk_data]     | Downloading package pl196x to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/pl196x.zip.
[nltk_data]     | Downloading package porter_test to /root/nltk_data...
[nltk_data]     |   Unzipping stemmers/porter_test.zip.
[nltk_data]     | Downloading package ppattach to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/ppattach.zip.
[nltk_data]     | Downloading package problem_reports to
[nltk_data]     |     /root/nltk_data...
```

```
[nltk_data]     |   Unzipping corpora/problem_reports.zip.
[nltk_data]     | Downloading package product_reviews_1 to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/product_reviews_1.zip.
[nltk_data]     | Downloading package product_reviews_2 to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/product_reviews_2.zip.
[nltk_data]     | Downloading package propbank to /root/nltk_data...
[nltk_data]     | Downloading package pros_cons to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/pros_cons.zip.
[nltk_data]     | Downloading package ptb to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/ptb.zip.
[nltk_data]     | Downloading package punkt to /root/nltk_data...
[nltk_data]     |   Unzipping tokenizers/punkt.zip.
[nltk_data]     | Downloading package punkt_tab to /root/nltk_data...
[nltk_data]     |   Unzipping tokenizers/punkt_tab.zip.
[nltk_data]     | Downloading package qc to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/qc.zip.
[nltk_data]     | Downloading package reuters to /root/nltk_data...
[nltk_data]     | Downloading package rslp to /root/nltk_data...
[nltk_data]     |   Unzipping stemmers/rslp.zip.
[nltk_data]     | Downloading package rte to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/rte.zip.
[nltk_data]     | Downloading package sample_grammars to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping grammars/sample_grammars.zip.
[nltk_data]     | Downloading package semcor to /root/nltk_data...
[nltk_data]     | Downloading package senseval to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/senseval.zip.
[nltk_data]     | Downloading package sentence_polarity to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/sentence_polarity.zip.
[nltk_data]     | Downloading package sentiwordnet to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/sentiwordnet.zip.
[nltk_data]     | Downloading package shakespeare to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/shakespeare.zip.
[nltk_data]     | Downloading package sinica_treebank to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/sinica_treebank.zip.
[nltk_data]     | Downloading package smultron to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/smultron.zip.
[nltk_data]     | Downloading package snowball_data to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     | Downloading package spanish_grammars to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping grammars/spanish_grammars.zip.
[nltk_data]     | Downloading package state_union to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/state_union.zip.
```

```
[nltk_data]     | Downloading package stopwords to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/stopwords.zip.
[nltk_data]     | Downloading package subjectivity to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/subjectivity.zip.
[nltk_data]     | Downloading package swadesh to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/swadesh.zip.
[nltk_data]     | Downloading package switchboard to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/switchboard.zip.
[nltk_data]     | Downloading package tagsets to /root/nltk_data...
[nltk_data]     |   Unzipping help/tagsets.zip.
[nltk_data]     | Downloading package tagsets_json to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping help/tagsets_json.zip.
[nltk_data]     | Downloading package timit to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/timit.zip.
[nltk_data]     | Downloading package toolbox to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/toolbox.zip.
[nltk_data]     | Downloading package treebank to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/treebank.zip.
[nltk_data]     | Downloading package twitter_samples to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/twitter_samples.zip.
[nltk_data]     | Downloading package udhr to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/udhr.zip.
[nltk_data]     | Downloading package udhr2 to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/udhr2.zip.
[nltk_data]     | Downloading package unicode_samples to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/unicode_samples.zip.
[nltk_data]     | Downloading package universal_tagset to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping taggers/universal_tagset.zip.
[nltk_data]     | Downloading package universal_treebanks_v20 to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     | Downloading package vader_lexicon to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     | Downloading package verbnet to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/verbnet.zip.
[nltk_data]     | Downloading package verbnet3 to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/verbnet3.zip.
[nltk_data]     | Downloading package webtext to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/webtext.zip.
[nltk_data]     | Downloading package wmt15_eval to /root/nltk_data...
[nltk_data]     |   Unzipping models/wmt15_eval.zip.
[nltk_data]     | Downloading package word2vec_sample to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping models/word2vec_sample.zip.
[nltk_data]     | Downloading package wordnet to /root/nltk_data...
```

```
[nltk_data]    | Downloading package wordnet2021 to /root/nltk_data...
[nltk_data]    | Downloading package wordnet2022 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet2022.zip.
[nltk_data]    | Downloading package wordnet31 to /root/nltk_data...
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet_ic.zip.
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/words.zip.
[nltk_data]    | Downloading package ycoe to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ycoe.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection all
```

Relaiser une segmentation des documents du corpus. Commencer au début par une segmentation à base de phrases et realiser pour chacune un nettoyage en effacant juste les cateres speciaux et la ponctuation. en sortie on doit avoir le vocabulaire du corpus (tous les mots) et la liste des bigrams (une liste de liste telques les sous listes comportes deux mots chacune)/

```
from nltk.tokenize import RegexpTokenizer
from nltk.tokenize import sent_tokenize
from nltk.lm.preprocessing import pad_both_ends
from nltk.util import bigrams, ngrams


def Segmentation_bigrams(corpus):
  Vocabulary=['<s>','</s>'] # pour les debuts et fins des phrases
  Bigrams=[] #[[mot1,mot2],[mot2,mot3]....]
  Sentences=sent_tokenize(corpus)
  tokenizer = RegexpTokenizer(r'\w+')
  for sentence in Sentences:
    word_tokens = [word.lower() for word in
tokenizer.tokenize(sentence)]
    Vocabulary=Vocabulary+word_tokens
    Bigrams=Bigrams+[list(bigrams(pad_both_ends([word for word in
word_tokens],n=2)))]
  return Vocabulary,Bigrams

Vocabulary,Bigrams=Segmentation_bigrams(job_news)

Vocabulary[:10]
```

```
['<s>',
 '</s>',
 'german',
 'industrial',
 'employment',
 'seen',
 'stagnating',
 'the',
```

```
 'number',
 'of']

Bigrams[:2]

[[('<s>', 'german'),
  ('german', 'industrial'),
  ('industrial', 'employment'),
  ('employment', 'seen'),
  ('seen', 'stagnating'),
  ('stagnating', 'the'),
  ('the', 'number'),
  ('number', 'of'),
  ('of', 'workers'),
  ('workers', 'employed'),
  ('employed', 'in'),
  ('in', 'the'),
  ('the', 'west'),
  ('west', 'german'),
  ('german', 'industrial'),
  ('industrial', 'sector'),
  ('sector', 'stagnated'),
  ('stagnated', 'in'),
  ('in', 'the'),
  ('the', 'last'),
  ('last', 'quarter'),
  ('quarter', 'of'),
  ('of', '1986'),
  ('1986', 'as'),
  ('as', 'a'),
  ('a', '50'),
  ('50', '000'),
  ('000', 'increase'),
  ('increase', 'in'),
  ('in', 'overall'),
  ('overall', 'employment'),
  ('employment', 'benefited'),
  ('benefited', 'only'),
  ('only', 'the'),
  ('the', 'services'),
  ('services', 'branch'),
  ('branch', 'the'),
  ('the', 'diw'),
  ('diw', 'economic'),
  ('economic', 'institute'),
  ('institute', 'said'),
  ('said', '</s>')],
 [('<s>', 'a'),
  ('a', 'diw'),
  ('diw', 'report'),
```

```
  ('report', 'added'),
  ('added', 'the'),
  ('the', 'general'),
  ('general', 'downturn'),
  ('downturn', 'in'),
  ('in', 'the'),
  ('the', 'economy'),
  ('economy', 'since'),
  ('since', 'last'),
  ('last', 'autumn'),
  ('autumn', 'had'),
  ('had', 'had'),
  ('had', 'a'),
  ('a', 'negative'),
  ('negative', 'effect'),
  ('effect', 'on'),
  ('on', 'the'),
  ('the', 'willingness'),
  ('willingness', 'of'),
  ('of', 'firms'),
  ('firms', 'to'),
  ('to', 'take'),
  ('take', 'on'),
  ('on', 'workers'),
  ('workers', '</s>')]]
```

Pour l'entrainement du modèle nous utilisons la librairie nltk.lm qui permet de realiser des modèles de langage ngram. La librairie fournit plusieurs modeles d'implementation (MLE, Laplace,Lidstone,....) pour calculer les differentes probabilités.

```
from nltk.lm import Laplace
from nltk.lm import MLE

model = MLE(2)
model.fit(Bigrams, Vocabulary)
```

Ci-dessous quelques exemples pour l'exploitation du modèle

```
print(model.counts)
```

```
<NgramCounter with 2 ngram orders and 14020 ngrams>
```

```
print(model.vocab)
```

```
<Vocabulary with cutoff=1 unk_label='<UNK>' and 1989 items>
```

```
model.counts[['kind']]['of']
```

```
2
```

```
#recuperer la probabilité d'avoir un terme en fonction d'un contexte
model.score('of','kind'.split())
#ou bien
model.score("of", ["kind"])
```

```
1.0
```

Génération des termes via le modèle

```
# mots=model.generate(10,text_seed=['kind']) # generation de 10 termes qui suivent "kind"
# " ".join(mots)
```

Le code precedent retourne une generation aléatoire des mots parmis ceux les plus probables. On peut utiliser un random_seed pour fixer la generation des termes via le modèle

```
mots=model.generate(10,text_seed=['kind'],random_seed=0)
" ".join(mots)
```

```
{"type":"string"}
```

```
mots=model.generate(10,text_seed=['french'],random_seed=5)
" ".join(mots)
```

```
{"type":"string"}
```

```
mots=model.generate(10,text_seed=['sweden'],random_seed=0)
" ".join(mots)
```

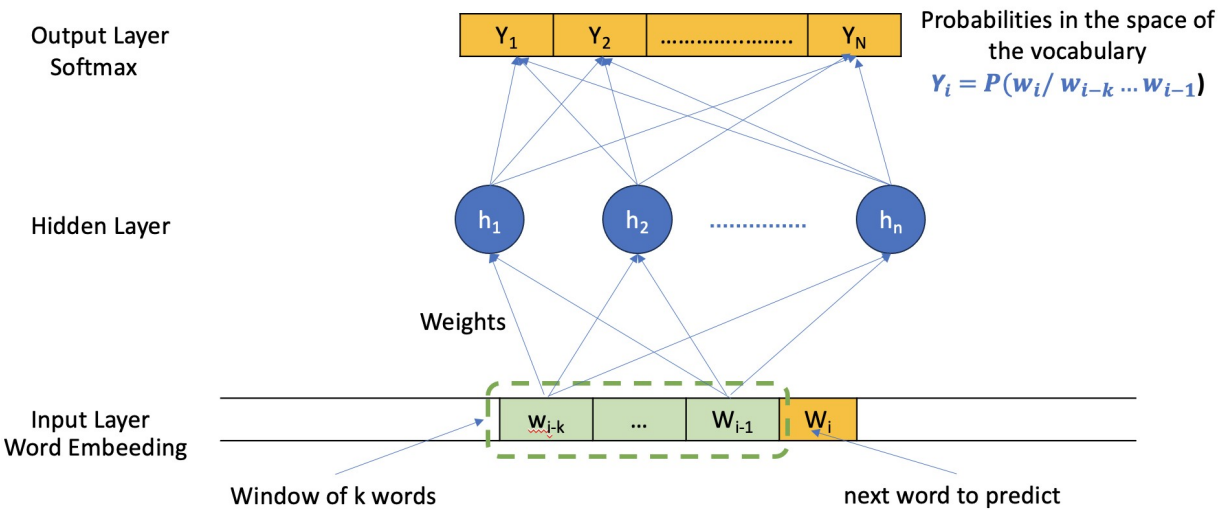
```
{"type":"string"}
```

2. Neural Language Model

Pour concevoir un modèle de langage neuronal , nous considérons un reseau de neurones à trois couche. une couche d'entrée, une couche cachée et une couhe de sortie.

Les mots doivent être dotés d'une structure vectorielle avant de les introduire dans le processus d'apprentissage.

La couche de sortie comportera un vecteur contenant les probabilités des generation des defifferents mots du vocabulaire.



2.1 Preparation du dataset d'apprentissage:

Nous consederant toujour le meme datset que la première section.

Pour la recuperation des embeedings, la librairie Gensim fourni une implémentation de l'algorithme Word2Vec :

```
*fasttext-wiki-news-subwords-300
*conceptnet-numberbatch-17-06-300
*word2vec-ruscorpora-300
*word2vec-google-news-300
*glove-wiki-gigaword-50
*glove-wiki-gigaword-100
*glove-wiki-gigaword-200
*glove-wiki-gigaword-300
*glove-twitter-25
*glove-twitter-50
*glove-twitter-100
*glove-twitter-200

import gensim.downloader
glove_vectors = gensim.downloader.load('word2vec-google-news-300')

[==================================================] 100.0%
1662.8/1662.8MB downloaded
```

Dans un premier temps nous allons commencer par des inputs de type bigrams. Recuperer la liste des bigrams crée précédemment et exploiter la dans la generation des vecteurs X et Y avec X[i] et Y[j] coresspondent au bigram [wi,wj], sachant que X[i] est l'embedding du mot wi et yi=1 (les autres yk=0)

```
import numpy
```

```
'''pour un bigrame <Wi,Wj>, il doit generer le vecteur
y=[...0...pj=1,..0...] pour
une entree x[i] aui corrependant à l'embeeding de wi
'''

def data_preparation(corpus):
  Vocabulary,Bigrams=Segmentation_bigrams(corpus) #Assurez vous que
Bigrams est une liste de bigrams
  X=[]
  Y=[]
  for lb in Bigrams:
    for bg in lb:
      if bg[0] in glove_vectors and bg[1] in glove_vectors:
        X=X+[glove_vectors[bg[0]]]
        y=numpy.array([0]*len(Vocabulary ))
        y[Vocabulary.index(bg[1])]=1
        Y=Y+[y]
  return X,Y

X,Y=data_preparation(job_news)
print(numpy.array(X).shape)  # Should be (num_samples, num_features)
print(numpy.array(Y).shape)  # Should be (num_samples, num_classes)
```

```
(9715, 300)
(9715, 13501)
```

2.1 Entrainement du modèle:

Le package keras sera utilisé pour l'implementation du langage de modèle neuronal. La documentation est disponible ICI

```
!pip install keras
```

```
Requirement already satisfied: keras in
/usr/local/lib/python3.10/dist-packages (3.5.0)
Requirement already satisfied: absl-py in
/usr/local/lib/python3.10/dist-packages (from keras) (1.4.0)
Requirement already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (from keras) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-
packages (from keras) (13.9.4)
Requirement already satisfied: namex in
/usr/local/lib/python3.10/dist-packages (from keras) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-
packages (from keras) (3.12.1)
Requirement already satisfied: optree in
/usr/local/lib/python3.10/dist-packages (from keras) (0.13.1)
Requirement already satisfied: ml-dtypes in
```

```
/usr/local/lib/python3.10/dist-packages (from keras) (0.4.1)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from keras) (24.2)
Requirement already satisfied: typing-extensions>=4.5.0 in
/usr/local/lib/python3.10/dist-packages (from optree->keras) (4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in
/usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0-
>rich->keras) (0.1.2)
```

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dense, Embedding, Flatten, Input, LSTM
import numpy as np

# Create the model
X = np.array(X)
X = np.reshape(X, (X.shape[0], 1, X.shape[1]))
model = Sequential([
    Input(shape=(X.shape[1], X.shape[2])),  # Input layer
    Dense(10, activation='relu'),   # First layer with 10 units
    LSTM(10,activation='relu' ),
    Dense(len(Vocabulary), activation='softmax')  # Output layer with
Softmax
])

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(numpy.array(X), numpy.array(Y), epochs=500, batch_size=100,
verbose=0)

# Evaluate the model (using the same data for simplicity)
loss, accuracy = model.evaluate(numpy.array(X), numpy.array(Y),
verbose=0)
print(f'Loss: {loss}, Accuracy: {accuracy}')
```

```
Loss: 3.0060665607452393, Accuracy: 0.3163149654865265
```

De la meme facons aue la premier section, donner le code exploitant le modèle dans la generation du texte à partir d'un mot de départ

```
def generate (start,n):
  sentence=""
```

```
  next_word=start
  for i in range(n):
    sentence=sentence+" "+next_word

    input_vector =
np.expand_dims(np.expand_dims(glove_vectors[next_word], axis=0),
axis=1)
    predictions = model.predict(input_vector)
    predictions=predictions.tolist()[0]
    next_word = Vocabulary[predictions.index(max(predictions))]
  return sentence

generate("employers",10)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 569ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
```

```
{"type":"string"}
```

Exercice

1.Tri grams

Refaire le meme travail avec le tri-grams

```
from nltk.util import bigrams, ngrams

def Segmentation_trigrams(corpus):
  Vocabulary=['<s>','</s>'] # pour les debuts et fins des phrases
  Trigrams=[] #[[mot1,mot2,mot3],[mot2,mot3,mot4]....]
  Sentences=sent_tokenize(corpus)
  tokenizer = RegexpTokenizer(r'\w+')
  for sentence in Sentences:
    word_tokens = [word.lower() for word in
tokenizer.tokenize(sentence)]
    Vocabulary=Vocabulary+word_tokens
    Trigrams=Trigrams+[list(ngrams(pad_both_ends([word for word in
word_tokens],n=3),3))]
```

```
  return Vocabulary,Trigrams

Vocabulary,Trigrams=Segmentation_trigrams(job_news)

Trigrams[:2]
```

```
[[('<s>', '<s>', 'german'),
  ('<s>', 'german', 'industrial'),
  ('german', 'industrial', 'employment'),
  ('industrial', 'employment', 'seen'),
  ('employment', 'seen', 'stagnating'),
  ('seen', 'stagnating', 'the'),
  ('stagnating', 'the', 'number'),
  ('the', 'number', 'of'),
  ('number', 'of', 'workers'),
  ('of', 'workers', 'employed'),
  ('workers', 'employed', 'in'),
  ('employed', 'in', 'the'),
  ('in', 'the', 'west'),
  ('the', 'west', 'german'),
  ('west', 'german', 'industrial'),
  ('german', 'industrial', 'sector'),
  ('industrial', 'sector', 'stagnated'),
  ('sector', 'stagnated', 'in'),
  ('stagnated', 'in', 'the'),
  ('in', 'the', 'last'),
  ('the', 'last', 'quarter'),
  ('last', 'quarter', 'of'),
  ('quarter', 'of', '1986'),
  ('of', '1986', 'as'),
  ('1986', 'as', 'a'),
  ('as', 'a', '50'),
  ('a', '50', '000'),
  ('50', '000', 'increase'),
  ('000', 'increase', 'in'),
  ('increase', 'in', 'overall'),
  ('in', 'overall', 'employment'),
  ('overall', 'employment', 'benefited'),
  ('employment', 'benefited', 'only'),
  ('benefited', 'only', 'the'),
  ('only', 'the', 'services'),
  ('the', 'services', 'branch'),
  ('services', 'branch', 'the'),
  ('branch', 'the', 'diw'),
  ('the', 'diw', 'economic'),
  ('diw', 'economic', 'institute'),
  ('economic', 'institute', 'said'),
  ('institute', 'said', '</s>'),
  ('said', '</s>', '</s>')],
 [('<s>', '<s>', 'a'),
```

```
  ('<s>', 'a', 'diw'),
  ('a', 'diw', 'report'),
  ('diw', 'report', 'added'),
  ('report', 'added', 'the'),
  ('added', 'the', 'general'),
  ('the', 'general', 'downturn'),
  ('general', 'downturn', 'in'),
  ('downturn', 'in', 'the'),
  ('in', 'the', 'economy'),
  ('the', 'economy', 'since'),
  ('economy', 'since', 'last'),
  ('since', 'last', 'autumn'),
  ('last', 'autumn', 'had'),
  ('autumn', 'had', 'had'),
  ('had', 'had', 'a'),
  ('had', 'a', 'negative'),
  ('a', 'negative', 'effect'),
  ('negative', 'effect', 'on'),
  ('effect', 'on', 'the'),
  ('on', 'the', 'willingness'),
  ('the', 'willingness', 'of'),
  ('willingness', 'of', 'firms'),
  ('of', 'firms', 'to'),
  ('firms', 'to', 'take'),
  ('to', 'take', 'on'),
  ('take', 'on', 'workers'),
  ('on', 'workers', '</s>'),
  ('workers', '</s>', '</s>')]]

model = MLE(3)
model.fit(Trigrams, Vocabulary)

print(f"model.counts {model.counts}")

model.counts <NgramCounter with 2 ngram orders and 14541 ngrams>

print(f"model.vocab {model.vocab}")

model.vocab <Vocabulary with cutoff=1 unk_label='<UNK>' and 1989
items>

model.counts[['kind']]['of']

0

#recuperer la probabilité d'avoir un terme en fonction d'un contexte
model.score('of','kind'.split())
#ou bien
model.score("of", ["kind"])

0
```

```
# mots=model.generate(10,text_seed=['kind']) # generation de 10 termes
qui suivent "kind"
# " ".join(mots)

model.fit(Trigrams, Vocabulary)

import numpy


'''pour un bigrame <Wi,Wj>, il doit generer le vecteur
y=[...0...pj=1,..0...] pour
une entree x[i] aui corrependant à l'embeeding de wi
'''

def data_preparation(corpus):
  Vocabulary,Trigrams=Segmentation_trigrams(corpus) #Assurez vous que
Bigrams est une liste de bigrams
  X=[]
  Y=[]
  for lb in Trigrams:
    for bg in lb:
      if bg[0] in glove_vectors and bg[1] in glove_vectors:
        X=X+[glove_vectors[bg[0]]]
        y=numpy.array([0]*len(Vocabulary ))
        y[Vocabulary.index(bg[1])]=1
        Y=Y+[y]
  return X,Y

X,Y=data_preparation(job_news)
print(numpy.array(X).shape)  # Should be (num_samples, num_features)
print(numpy.array(Y).shape)  # Should be (num_samples, num_classes)

(9715, 300)
(9715, 13501)

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dense, Embedding, Flatten, Input, LSTM
import numpy as np

# Create the model
X = np.array(X)
X = np.reshape(X, (X.shape[0], 1, X.shape[1]))
model = Sequential([
    Input(shape=(X.shape[1], X.shape[2])),  # Input layer
    Dense(10, activation='relu'),   # First layer with 10 units
    LSTM(10,activation='relu' ),
    Dense(len(Vocabulary), activation='softmax')  # Output layer with
Softmax
])
```

```
# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(numpy.array(X), numpy.array(Y), epochs=500, batch_size=100,
verbose=0)

# Evaluate the model (using the same data for simplicity)
loss, accuracy = model.evaluate(numpy.array(X), numpy.array(Y),
verbose=0)
print(f'Loss: {loss}, Accuracy: {accuracy}')
```

```
Loss: 3.00656795501709, Accuracy: 0.32166752219200134
```

```
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout, Input

# Improved Model
model = Sequential([
    Input(shape=(X.shape[1], X.shape[2])),  # Input layer
    LSTM(128, activation='tanh', return_sequences=True),  # First LSTM
layer
    Dropout(0.4),  # Regularization
    LSTM(128, activation='tanh'),  # Second LSTM layer
    Dropout(0.4),  # Regularization
    Dense(32, activation='relu'),  # Fully connected layer
    Dense(len(Vocabulary), activation='softmax')  # Output layer with
Softmax
])

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(numpy.array(X), numpy.array(Y), epochs=500, batch_size=100,
verbose=0)

# Evaluate the model (using the same data for simplicity)
loss, accuracy = model.evaluate(numpy.array(X), numpy.array(Y),
verbose=0)
print(f'Loss: {loss}, Accuracy: {accuracy}')
```

```
Loss: 2.306885004043579, Accuracy: 0.3815748989582062
```