# Intro to NLP 2022: Assignment 1

In this assignment, we work with a dataset that contains sentences from news articles. It has been collected for a shared task at SemEval 2018 for *Complex Word Identification*.

**Task Description:** <https://sites.google.com/view/cwisharedtask2018/>

**Code for the assignment:** ​​*intro2nlp\_assignment1\_code.zip*

You submit a **pdf** of this document, the format should not be changed.

All floating point numbers should be rounded to **two decimals**.

Your analyses should be conducted using **python 3.8**.   
You submit a **zip**-file containing all your code.

You are allowed to use Python packages (e.g. pandas, sklearn).

Each team member needs to be able to explain the details of the submission. By default, all team members will receive the same grade. If this seems unjust to you, provide an extra statement indicating the workload of each team member.

**Total points**: 20

**Structure:**

* Part A: Linguistic analysis of the dataset using spacy, 6 points
* Part B: Understanding the task of complex word identification, 7 points
* Part C: Modeling the task with an LSTM, 7 points
* Bonus tasks: options for obtaining a grade >8

Fill in your details below:

**Group number:**

**Student 1**

**Name: Nils Breeman**

**Student id:**

**Student 2**

**Name: Julius Wantenaar**

**Student id:**

**Student 3**

**Name: Sebastiaan Bye**

**Student id:**

## PART A: Linguistic analysis using spaCy

In the first part of the assignment, we focus on an analysis of the sentences in the training data.

**File:** *data/preprocessed/train/sentences.txt*

Implement your analyses in *TODO\_analyses.py.*

Note that we are using the most recent spaCy version (3.2) and the model *en\_core\_web\_sm*. Results might vary for other versions. If you cannot use 3.2, clearly explain this to your TA and specify on your submission which version you are using instead.

1. **Tokenization** (1 point)  
   Process the dataset using the spaCy package and extract the following information:

Number of tokens:

Number of types:

Number of words:

Average number of words per sentence:

Average word length:

Provide the definition that you used to determine words:

1. **Word Classes** (1.5 points)

Run the default part-of-speech tagger on the dataset and identify the ten most frequent POS tags. Complete the table below for these ten tags (the tagger in the model *en\_core\_web\_sm* is trained on the PENN Treebank tagset).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Finegrained POS-tag | Universal POS-Tag | Occurrences | Relative Tag Frequency (%) | 3 most frequent tokens with this tag | Example for an infrequent token with this tag |
| NN | Noun | 2099 | 0.14 | \\, year, report | deterioration |
| NN | PRON, NOUN | 2055 | 0.18 | year, report, time | A. |
| NNP | AUX, PROPN | 1793 | 0.16 | US, President, U.S. | - |
| IN | ADP, SCONJ | 1744 | 0.16 | of, in, to | About |
| DT | PRON, DET | 1379 | 0.12 | the, a, The | An |
| JJ | ADJ | 872 | 0.08 | other, Russian, presidential | 21st |
| NNS | NOUN | 781 | 0.07 | ants, troops, people | 1970s |
| , | PUNCT | 699 | 0.06 | , ; … | ; |
| VBD | AUX, VERB | 658 | 0.06 | was, were, said | acknowledged |
| . | PUNCT | 655 | 0.06 | . ? ! | ! |
| VBN | AUX, VERB | 501 | 0.04 | been, accused, killed | - |

1. **N-Grams** (1.5 points)  
   Calculate the distribution of n-grams and provide the 3 most frequent

Token bigrams: of the, in the, to the

Token trigrams: in response to, Republican Party presidential, local time UTC

POS bigrams: **NNP NNP, DT NN, IN DT**

POS trigrams: IN DT NN, NNP NNP NNP, DT NN IN

1. **Lemmatization** (1 point)

Provide an example for a lemma that occurs in more than two inflections in the dataset.

Lemma: challenge

Inflected Forms: challenged, challenging, challenges, challenge

Example sentences for each form:

1. That decision was the one **challenged** unsuccessfully in the High Court .
2. Next week , on January 24 , the Supreme Court is due to deliver a decision in a case **challenging** the government 's right to issue Article 50 — which starts the Brexit negotiations — without the consultation of Parliament .
3. U.S. presidential candidate Mark Everson **challenges** debate exclusion U.S. Republican Party presidential candidate Mark Everson , former commissioner of the Internal Revenue Service ( IRS ) , filed a complaint on Monday with the Federal Election Commission ( FEC ) to challenge his exclusion from Thursday 's first Fox News Republican Party presidential debate .
4. Election law expert Richard Winger , publisher of Ballot Access News , says Everson is " completely correct " in his **challenge** .

1. **Named Entity Recognition** (1 point)

Number of named entities:  
Number of different entity labels:

Analyze the named entities in the first five sentences. Are they identified correctly? If not, explain your answer and propose a better decision.

## PART B: Understanding the task of complex word identification

1. **Explore the dataset** (1.5 points)  
   Read the documentation (<https://sites.google.com/view/cwisharedtask2018/datasets>) of the dataset and provide an answer to the following questions:

a) What do the start and offset values refer to? Provide an example.

The start value refers to the first character in the string of the target word. The offset value refers to the last character + 1 of the target word.

Example (stored in **sentence**)

#37-1 Guatemalan Supreme Court approves impeachment of President Molina Yesterday in Guatemala, the Supreme Court approved the attorney general's request to impeach President Otto Pérez Molina.

For this sentence, the start == 31, and offset == 39.The target word ‘approves’ starts at the 31st character, and ends at the 38th character of the string. If you fill in the start and offset values in a list slicing statement:

**sentence**[31:39]

which gives exactly the target word ‘sentence’ (since the value after the colon is not included for list slicing).

b) What does it mean if a target word has a probabilistic label of 0.4?

probabilistic label = the number of annotators who marked the word as difficult / the total number of annotators.

So it is the ratio of annotators who marked it difficult to the total number of annotators.

c) The dataset was annotated by native and non-native speakers. How do the binary and the probabilistic complexity label account for this distinction?

They do not account for this distinction:

1. For binary: if either a native or non-native speaker marks the target word as difficult, the label is set to 1, and zero otherwise. Thus, no distinction is made.
2. For probabilistic: no distinction is made between a word being marked as difficult by native and non-native speakers. Both types of annotators are counted in the numerator (and the total number of annotators is counted for the denominator).
3. **Extract basic statistics** (0.5 point)

Let’s have a closer look at the labels for this task.   
 Use the file *data/original/english/WikiNews\_Train.tsv* and extract the following columns:

Target word, binary label, probabilistic label

Provide the following information:

Number of instances labeled with 0: 4530

Number of instances labeled with 1: 3215

Min, max, median, mean, and stdev of the probabilistic label:

* + Min = 0.0
  + Max = 1.0
  + Median = 0.0
  + Mean = 0.083951
  + Stdev = 0.169690

Number of instances consisting of more than one token: 1086

Maximum number of tokens for an instance: 10

1. **Explore linguistic characteristics** (2 points)  
   For simplicity, we will focus on the instances which consist only of a single token and have been labeled as complex by at least one annotator.   
   Calculate the length of the tokens as the number of characters.

Calculate the frequency of the tokens using the wordfreq package (<https://pypi.org/project/wordfreq/>).

Provide the Pearson correlation of length and frequency with the probabilistic complexity label:

Pearson correlation length and complexity: 0.282

Pearson correlation frequency and complexity: -0.298

Provide 3 scatter plots with the probabilistic complexity on the y-axis.

X-axis: 1) Length 2) Frequency 3) POS tag

Set the ranges of the x and y axes meaningfully.

Plot 1:



Plot 2:

Chart, scatter chart

Description automatically generated

Plot 3:

A picture containing chart

Description automatically generated

Interpret the results in 3-5 sentences:

There is not a clear relationship between word length and probabilistic complexity the observations are evenly distributed over the space. There does appear to be a graphical relationship between word frequency and probabilistic complexity, words that have a lower frequency have a higher probabilistic complexity. Lastly, verbs, nouns, pronpn adjectives and adverbs all range from very simple to very complex. The other tags are only seen scoring low on probabilistic complexity.

1. **Reflection (1 Point)**

Can you think of another linguistic characteristic that might have an influence on the perceived complexity of a word? Propose at least one and explain your choice in 2-4 sentences.

1. **Baselines** (2 Points)

Implement four baselines for the task in *TODO\_baselines.py*.

Majority baseline: always assigns the majority class

Random baseline: randomly assigns one of the classes

Length baseline: determines the class based on a length threshold

Frequency baseline: determines the class based on a frequency threshold

Test different thresholds and choose the one which yields the highest accuracy on the dev\_data:

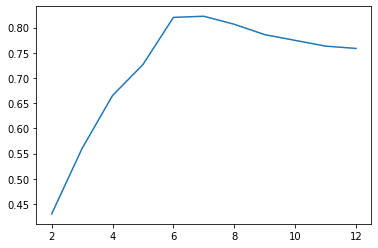
Length threshold:

Frequency threshold:

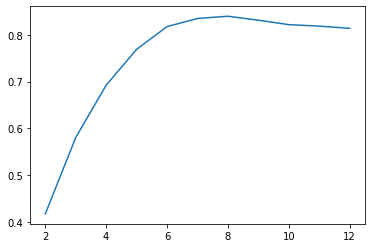
Fill in the table below (round to two decimals!):

|  |  |  |
| --- | --- | --- |
| Baseline | Accuracy on dev | Accuracy on test |
| Majority | 0.81 | 0.75 |
| Random | 0.48 | 0.47 |
| Length | 0.84 | 0.82 |
| Frequency | 0.73 | 0.73 |

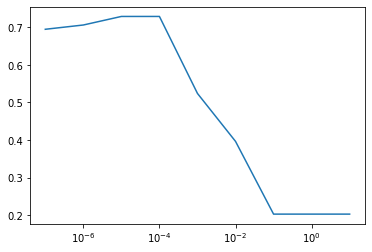
Length: Test



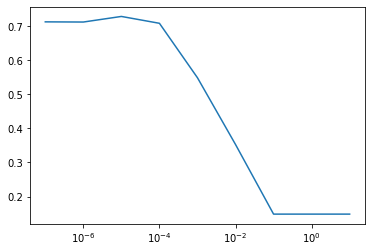
Dev:

b

Frequency: Test



Dev



Interpret the results in 2-3 sentences.

Accuracy is a bad measure. The interesting results come from the plots. Length is the best predictor

Store the predictions in a way that allows you to calculate precision, recall, and F-measure and fill the table in exercise 12.

## PART C: Modeling the task

For part C, we use an implementation for a vanilla LSTM which was originally developed for a named entity recognition project for a Stanford course. You can find more documentation here: <https://github.com/cs230-stanford/cs230-code-examples/tree/master/pytorch/nlp>

1. **Understanding the code** (1.5 Points)  
   Familiarize yourself with our version of the code and try to understand what is going on.

Answer in your own words (1-3 sentences per question)

Run the file *build\_vocab.py*. What does this script do?

This script builds a vocabulary of words and tags from a dataset. It does this by reading in test and training datasets and keeping only the most frequent tokens which are saved to the vocabulary.

Inspect the file *model/net.py.* Which layers are being used and what is their function?

* Embedding layer: Takes each index in the range of vocab\_size and maps them to an embedding dimension vector
* LSTM: applies a long short-term memory on the input and returns an output for each token in a given sentence
* Fully connected layer: Takes the LSTM output for every token to a distribution over NER tags

How could you change the loss function of the model?

The current lose function used is a cross entropy loss function. We would instead use a square loss function. The reason for this is clearly explained by Hui and Belkin (2021) who showed that for standard NLP datasets, square loss produces better results in the majority of NLP experiments. Furthermore, square loss function have the benefit of being less sensitive to randomness in initialization and leads to smaller neural nets as the final softmax layer in cross-entropy needs to be removed.

1. **Detailed evaluation** (2.5 points)

Train the model on the data in *preprocessed/train* and *preprocessed/dev* by running the code in *train.py*.

Evaluate the model on the data in *preprocessed/test*.

The original code only outputs the accuracy and the loss of the model. I adapted the code for you, so that it writes the predictions to *model\_output.tsv*.

Implement calculations for precision, recall, and F1 for each class in *TODO\_detailed\_evaluation.py*. You can use existing functions but make sure that you understand how they work.

Provide the results for the baselines and the LSTM in the table below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Class N | | | Class C | | | Weighted Average |
|  | Precision | Recall | F1 | Precision | Recall | F1 | F1 |
| Random | 0.78 | 0.48 | 0.59 | 0.18 | 0.47 | 0.27 | 0.53 |
| Majority | 0.79 | 1 | 0.89 | 0 | 0 | 0 | 0.71 |
| Length | 0.87 | 0.96 | 0.92 | 0.76 | 0.48 | 0.59 | 0.85 |
| Frequency | 0.84 | 0.82 | 0.82 | 0.34 | 0.38 | 0.36 | 0.73 |
| LSTM | 0.88 | 0.94 | 0.91 | 0.67 | 0.47 | 0.56 | 0.84 |

1. **Interpretation** (1.5 Points)

Compare the performance to the results in the shared task (<https://aclanthology.org/W18-0507.pdf>) and interpret the results in 3-5 sentences. Don’t forget to check the number of instances in the training and test data and integrate this into your reflection.

In the conclusion of the paper they reported that mostly, traditional feature engineering approaches based on length and frequency outperform neural networks. We find a similar result in our analysis for the dataset with except for the frequency based approach for class c. However, summing the sentences of the test, training and validation set we get 652+19+85 = 756, which shows that the test/training split is not well balanced which may influence performance measures as test set should be larger.

1. **Experiments** (2 points)  
   Vary a hyperparameter of your choice and plot the F1-results (weighted average) for at least 5 different values. Examples for hyperparameters are embedding size, learning rate, number of epochs, random seed,

Hyperparameter: lstm\_hidden\_dim

Plot:

Chart, line chart

Description automatically generated

Interpret the result (2-4 sentences):

Increasing the amount of nodes does not lead to a better F1 score. This is probably due to the fact that the network needs more time to train with a larger hidden layer. However, when there are too little nodes, the network is unable to learn the patterns that make a word complex.

Provide 3 examples for which the label changes when the hyperparameter changes:

1. officers, Label at Value 1:1, Label at Value 2:0
2. here, Label at Value 1:0, Label at Value 2:1
3. met, Label at Value 1:1, Label at Value 2:0

## Bonus Tasks

The maximum grade you can get for the assignment is an 8. If you want to obtain a better grade, you need to individually send results for one of the bonus tasks to intro2nlp@googlegroups.com. If the group project grade is less than an 8, we do not check the bonus task submission. If the group project grade is an 8 and you submitted an answer for a bonus task, you might still only receive an 8, if the quality of the bonus task submission is not sufficient.

Task options:

* Provide answers for exercises 8 and 12-14 for at least one of the other languages of the CWI task.
* Improve the model by making a substantial change. Varying a hyperparameter or simply adding another layer **is not** a substantial change. Motivate your modification and interpret the findings.
* Identifying complex words is only the first step for lexical simplification. Read up on related work and explain potential architectures for contextualized lexical simplification in detail.

Spanish:

Question 8.

1. For simplicity, we will focus on the instances which consist only of a single token and have been labeled as complex by at least one annotator.   
   Calculate the length of the tokens as the number of characters.

Calculate the frequency of the tokens using the wordfreq package (<https://pypi.org/project/wordfreq/>).

Provide the Pearson correlation of length and frequency with the probabilistic complexity label:

Pearson correlation length and complexity: 0.22

Pearson correlation frequency and complexity: -0.16

Provide 3 scatter plots with the probabilistic complexity on the y-axis.

X-axis: 1) Length 2) Frequency 3) POS tag

Set the ranges of the x and y axes meaningfully.

Scatter chart

Description automatically generated

A picture containing text

Description automatically generated

Chart, scatter chart

Description automatically generated

Question 12

Train the model on the data in *preprocessed/train* and *preprocessed/dev* by running the code in *train.py*.

Evaluate the model on the data in *preprocessed/test*.

The original code only outputs the accuracy and the loss of the model. I adapted the code for you, so that it writes the predictions to *model\_output.tsv*.

Implement calculations for precision, recall, and F1 for each class in *TODO\_detailed\_evaluation.py*. You can use existing functions but make sure that you understand how they work.

Provide the results for the baselines and the LSTM in the table below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Class N | | | Class C | | | Weighted Average |
|  | Precision | Recall | F1 | Precision | Recall | F1 | F1 |
| Random | 0.50 | 0.88 | 0.64 | 0.50 | 0.12 | 0.19 | 0.41 |
| Majority | 1 | 0.88 | 0.93 | 0.0 | 0 | 0 | 0.94 |
| Length | 0.82 | 0.94 | 0.87 | 0.60 | 0.31 | 0.41 | 0.77 |
| Frequency | 0.98 | 0.19 | 0.31 | 0.44 | 0.99 | 0.61 | 0.43 |
| LSTM | 0.87 | 0.95 | 0.91 | 0.67 | 0.43 | 0.53 | 0.83 |

Question 13

We see once again that the LSTM doesn’t outperform simple statistical models on F1 score. However, it is the model that works best without having a 0 in any category so I would prefer it when I would have to choose the best model for a general purpose.

Question 14.

Increasing the amount of nodes leads to a better F1 score. This suggests that we should use large hidden layers to capture the patterns that decide if a word is complex

Chart, line chart

Description automatically generated