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: 17

Student 1
Name: Gergő Pandurcsek
Student id: 2730356

Student 2
Name: Marton Fejer
Student id: 2694002

Student 3
Name: Carol Rameder
Student id: 2747982

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: 15477
   Number of types: 3721
   Number of words: 13242
   Average number of words per sentence: 18.89
   Average word length: 6.54

   Provide the definition that you used to determine words:
   Every token that does not fall under the *punct* or *space* POS category.
   (Speech and Language Processing and Essentials of Linguistics support this for this task)

2. **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
NNP	Propn	2017	0.13	\, US, President	Álvarez
IN	Adp	1600	0.1	of, in, on,	underneath
DT	Det	1313	0.08	the, a, The,	An
JJ	Adj	869	0.05	other, Russian, military	regional
NNS	Noun	779	0.05	ants, police, troops	concentrations
,	Punct	699	0.05	…	…
.	Punct	655	0.04	. ? !	!
VBN	Verb	454	0.03	accused, known, killed	hospitalised

VBD	Verb	451	0.03	said, reported, told	plated

3. **N-Grams** (1.5 points)
   Calculate the distribution of n-grams and provide the 3 most frequent
   Token bigrams: \\ " : 240, of the : 82, , and : 67
   Token trigrams: , \\ " : 40, \\ " . : 28, \\ " , : 13

   Fine-grained POS bigrams: DT NN: 671, NNP NNP: 608, IN DT: 586
   Fine-grained POS trigrams: IN DT NN: 292, NNP NNP NNP: 200, DT NN IN: 195

   Universal POS bigrams: DET NOUN: 780, NOUN PUNCT: 780, NOUN ADP: 710
   Universal POS trigrams: ADP DET NOUN: 302, NOUN ADP DET: 243, VERB DET NOUN: 228

4. **Lemmatization** (1 point)
   Provide an example for a lemma that occurs in more than two inflections in the dataset.
   Lemma: fall
   Inflected Forms: falling, fell, falls, fallen
   Example sentences for each form:
   - falling: The nonfatal accident left an elderly motorcyclist with broken ribs after he came off his bike avoiding the 170 tonnes of falling debris .
   - fell: Bianchi 's helmet became wedged underneath the tractor , causing a diffuse axonal injury , and he fell into a coma .
   - falls: The final decision on lifting Presidential immunity now falls to Congress .
   - fallen: At 9 P.M. EST , INDYCAR released a statement , announcing that Wilson had suffered a severe head injury , was in critical condition and had fallen into a coma .

5. **Named Entity Recognition** (1 point)
   Number of named entities: 1614
   Number of different entity labels: 17

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

   children are thought to be aged three , eight DATE , and ten years DATE , alongside an eighteen-month-old DATE baby .
   We mixed different concentrations of ROS GPE with the spores , plated them out on petridishes with an agar-solution where fungus can grow on .
   They feel they are under-represented in higher education and are suffering in a regional economic downturn .
   Especially as it concerns a third ORDINAL party building up its military presence near our borders .
   Police said three CARDINAL children were hospitalised for \ ORG " severe dehydration \" .

   2nd sentence : ROS not a named entity, no entities in this sentence.
   5th sentence : \ not a named entity

PART B: Understanding the task of complex word identification

6. **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 offset is the number from which character the target word (/sequence) starts.
   The end offset is the number at which character the target word (/sequence) ends.

```
  Both China and the Philippines flexed their muscles on Wednesday.
31 51 flexed their muscles
```

   Starting from B, the 31st character is the f of the flexed word which is part of the target word and the 51st character is after the s of muscles, which is the last part of the target word

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

```
  Both China and the Philippines flexed their muscles on Wednesday.
31 37 flexed 10 10 2 6 1 0.4
```

   # of natives who read the sentence : 10
   # of non-natives who read the sentence : 10
   Total : 10+10=20
   # of natives who marked the target word as complex : 2
   # of non-natives who market the target word as complex : 6
   Total : 2+6=8
   Probability : 8/20 = 0.4

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

   None of them makes distinction between the annotators, the binary value is flipped by any marking, and the probability uses cumulative values.

7. **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

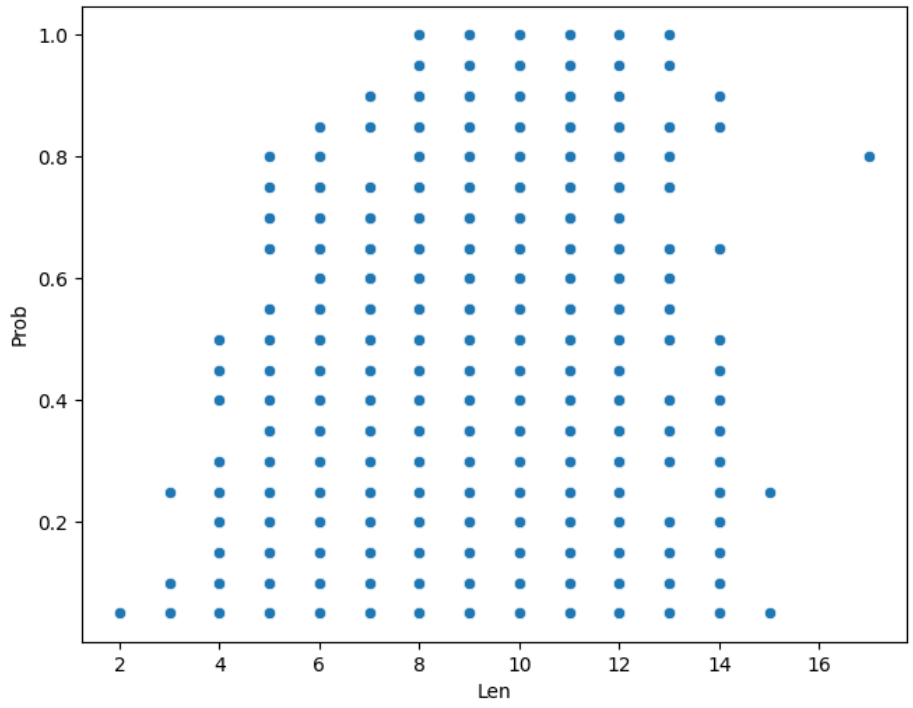
   Provide the following information:
   Number of instances labeled with 0: 4530
   Number of instances labeled with 1: 3216
   Min, max, median, mean, and stdev of the probabilistic label:
   0, 1, 0, 0.084, 0.17

Number of instances consisting of more than one token: 1086
Maximum number of tokens for an instance: 10

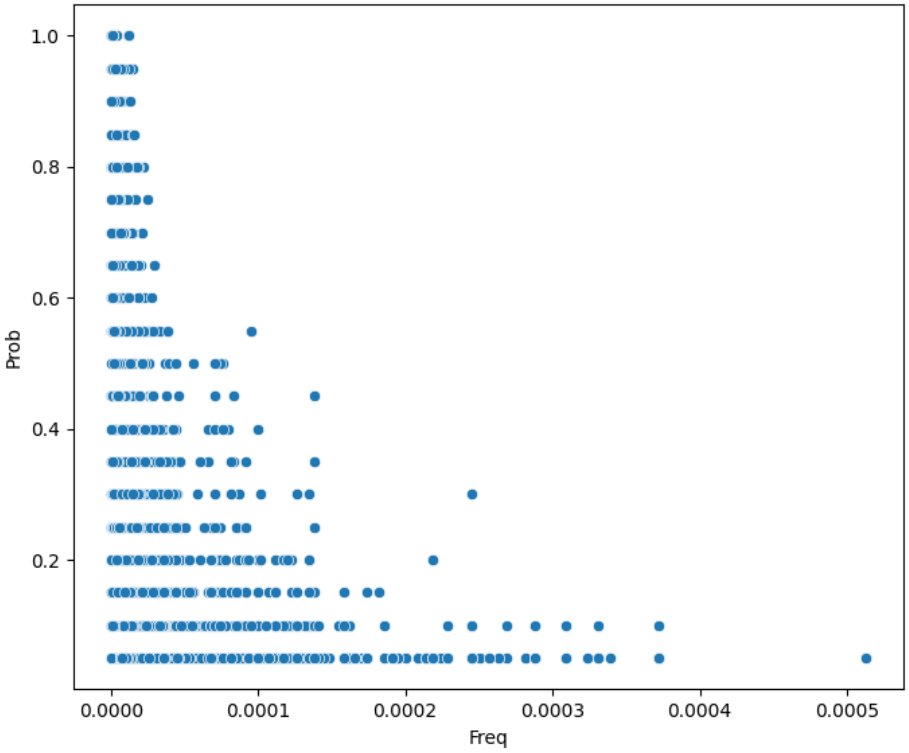
8. **Explore linguistic characteristics** (2 points)
   For simplicity, we will focus on the instances which consist only of a single token and have been labeled as complex by at least one annotator.
   Calculate the length of the tokens as the number of characters.
   Calculate the frequency of the 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: 0.3
   Pearson correlation frequency and complexity: -0.31
   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.

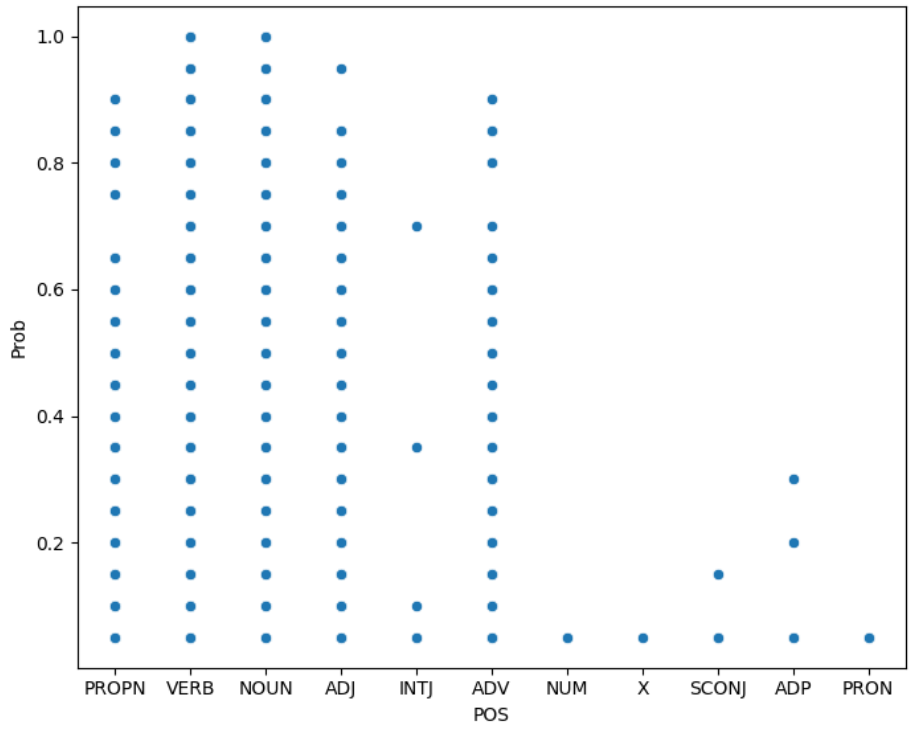
   Plot 1:



Plot 2:



Plot 3:



Interpret the results in 3-5 sentences:
Length – Complexity :
The plot is in line with the correlation, the words can become a lot more complex around length 5, below that there are fewer combinations of characters for a word which makes them easier to memorize and recognize.
Frequency – Complexity :
The plot is again in line with the correlation, there seems to be a negative exponential correlation between the frequency and complexity, which makes sense, since the more frequent a word is, the higher the chance that you are familiar with that word.
POS – Complexity :
We can see that not every tag is similarly complex, in my opinion the cause of this is the number of words in that particular vocabulary, there are a lot less adpositions or numbers than nouns and verbs, therefore they are easier to recognize and memorize.

9. **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.
   Phonetics :
   If you pronounce a word differently than it is written, it is can be harder to recognize it if you haven't seen it in written form much before. For example: coup d'état

10. **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: 8
    Frequency threshold: 0.00004

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

Baseline	Accuracy on dev	Accuracy on test
Majority	0.85	0.8
Random	0.5	0.5
Length	0.88	0.85
Frequency	0.74	0.76

    Interpret the results in 2-3 sentences.
    The reason why the majority baseline is so high because the classes are imbalanced, and the random baseline was generated over 100 cases which could explain the 50% accuracy. It seems like the length is a stronger predictor of complexity than frequency.

    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

11. **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 vocab (i.e. a list with counts for all instances) of all the words used in the training, validation and test files (which have the name 'sentences.txt'). A vocab is also built for all the tags found in the training, validation and test datasets (which have the name 'labels.txt')

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

    There are three main components of the neural network:

- **the embedding layer**, which encodes the input (i.e. the tokens) into a vector of fixed size. This layer should have one dimension of length equal to the size of the vocab.
- **the LSTM layer**, which is a recurrent neural network. The weights of the gates in this layer change over the course of training. These are the trainable parameters.
- **A fully connected linear layer**, which connects the output of the LSTM layer to a final output layer. The number of output nodes of this layer needs to correspond to the number of classes that can be predicted. The loss function (e.g. cross-entropy loss) uses these outputs to calculate the error, which can be backpropagated to update all trainable parameters/weights.

    How could you change the loss function of the model?

    In the file 'model/net.py', the loss function is explicitly defined as a separate function, which is passed as an argument for the training function. This function can be customized of course, but Pytorch also includes some common loss functions.

12. **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* by running *evaluate.py*.

The original code only outputs the accuracy and the loss of the model. I adapted the code, so that it writes the predictions to *experiments/base_model/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	0.82	0.53	0.64	0.23	0.54	0.32	0.58
Majority	0.80	1.00	0.89	NaN	0.00	NaN	NaN
Length	0.85	0.99	0.91	0.85	0.31	0.46	0.77
Frequency	0.91	0.77	0.83	0.44	0.70	0.54	0.82
LSTM (25 epochs)	0.88	0.94	0.91	0.68	0.48	0.57	0.84

13. **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.

	Paper	Our work
Train	27,299 sentences	653 sentences
Test	4,252 sentences	19 sentences
Best performance	0.84 (Camb)	0.84 (overall weighted average)

The performance of the simple LSTM model is similar to the best obtained in the shared task by the Camb team. The difference in the size of the training and test sets is a key factor in explaining the performance similarity between the two models, which are completely different in complexity. The model developed by the Camb team uses a different technique, AdaBoost, alongside with a complex procedure of feature engineering. This explains their high F-1 score of the model, which gained considerably more knowledge than the LSTM, given the large difference of the test sets, 4252 compared to 19.

14. **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: embedding dimension size (and LSTM hidden dimension size in 2 of the experiments), and learning rate

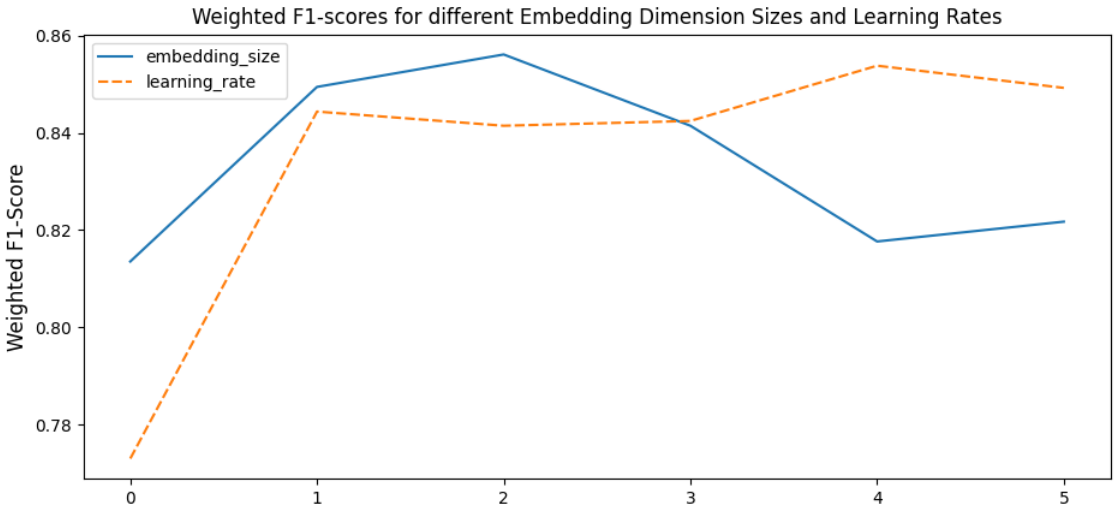
Plot: weighted f1-scores

25 Epochs, batch size = 5

1st picture:

blue line: hidden lstm and embedding dimension varied, learning rate = 0.001

orange line: hidden lstm dim. = embedding dimension = 50, learning rate varied

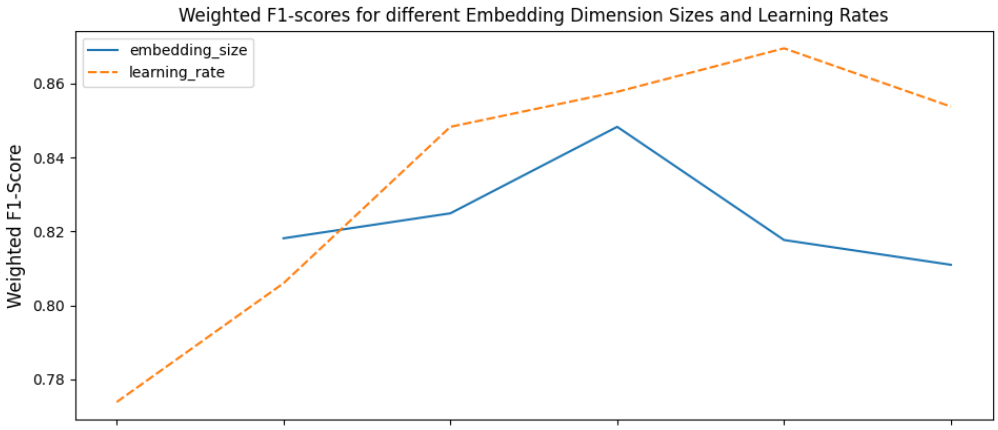


Emb. dim. size	3	10	25	50	100	250
LSTM hid. dim.	3	10	25	50	100	250
Learning rate	0.0001	0.0003	0.001	0.003	0.01	0.03

2nd picture: LSTM hidden dimension fixed at 100

blue line: embedding dimension varied, learning rate = 0.001

orange line: embedding dimension = 50, learning rate varied



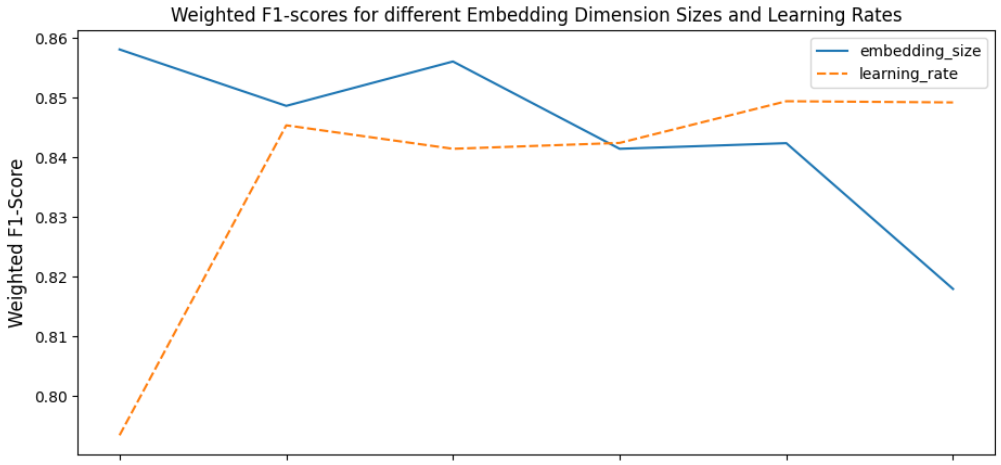
Emb. dim. size	3	10	25	50	100	250
LSTM hid. dim.	100	100	100	100	100	100
Learning rate	0.0001	0.0003	0.001	0.003	0.01	0.03

50 Epochs, batch size = 5

1st picture:

blue line: hidden lstm and embedding dimension varied, learning rate = 0.001

orange line: hidden lstm dim. = embedding dimension = 50, learning rate varied

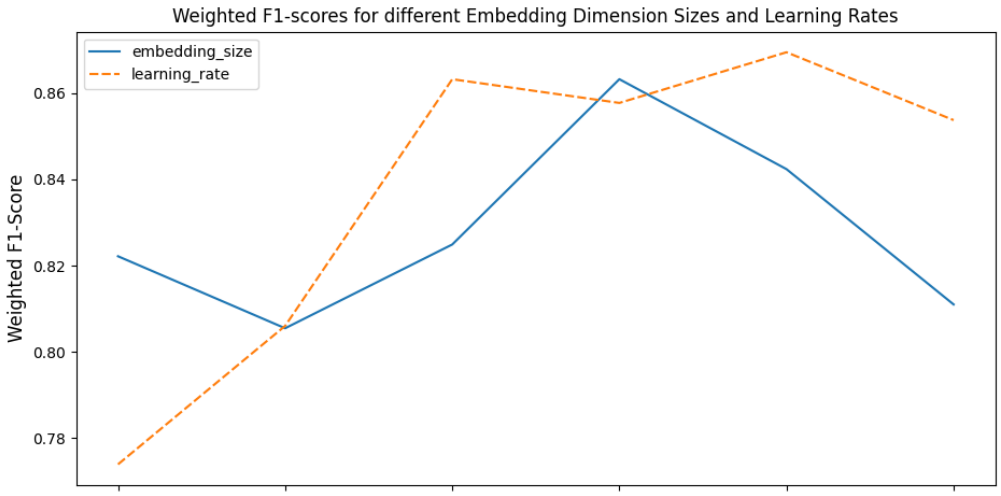


Emb. dim. size	3	10	25	50	100	250
LSTM hid. dim.	3	10	25	50	100	250
Learning rate	0.0001	0.0003	0.001	0.003	0.01	0.03

2nd picture: LSTM hidden dimension fixed at 100

blue line: embedding dimension varied, learning rate = 0.001

orange line: embedding dimension = 50, learning rate varied



Emb. dim. size	3	10	25	50	100	250
LSTM hid. dim.	100	100	100	100	100	100
Learning rate	0.0001	0.0003	0.001	0.003	0.01	0.03

Interpret the result (2-4 sentences):

The most optimal value for the learning rate is 0.01 when the trend for both types of epochs and embedding size reaches the peak value. When looking at the models with larger hidden dimensions of the LSTM, the highest F1_scores are reached for the pre-set embedding size of 50. The variation of the F1-Score with the respect to the changing learning rate is similar to the LSTM models of the same size. Overall, the most optimal configuration of the model is with a hidden dimension set to 100 and a learning rate of 0.01, as in both measured cases the F1-Score is the highest at almost 0.87.

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

1. Example 1, Label at Value 1, Label at Value 2
2. Example 2, Label at Value 1, Label at Value 2
3. Example 3, Label at Value 1, Label at Value 2

Learning rate was changed from 0.01 to 0.0001

	Gold standard	Output of model - lr=0.01	Output of model - lr=0.0001
House (row 38)	N	C	N
gunfire (row 49)	C	C	N
upwards (row 142)	C	C	N

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.