

Natural Language Processing

Prodromos Kampouridis MTN2203

University of Piraeus

National Center for Scientific Research “DEMOKRITOS”

Athens, Greece

28 May 2023

Professor

Efstathios Stamatatos,

Dept. of Information and Communication Systems Engineering,

University of the Aegean

ABSTRACT

This project was assigned as part of the coursework for the Natural Language Processing course in the Master’s program in Artificial Intelligence at NCSR “Demokritos” and the University of Piraeus. The assignment focuses on the concepts of Part-of-Speech (POS) tagging, Named Entity Recognition (NER) and Text Chunking using pre-trained models such as BERT and RoBERTa. The behavior and performance of these concepts are analyzed, and their results are interpreted and visually presented. Additionally, the assignment includes using a Transformers based generative language model, ChatGPT for zero-shot and few-shot prompting for all three tasks.

KEYWORDS

Named Entity Recognition; Part-of-Speech tagging; Text Chunking; pre-trained; BERT; RoBERTa; zero-shot prompting; few-shot prompting;

SEQUENCE LABELING

Initially, we modify the given code of the ner-bert.py file, so that we are capable to reuse some of the functionality for multiple questions. The code for building tagset and tag ids is moved under the *build_tags* function which also takes as a parameter the *tags_type* that we want to use, so then we will reuse this method for NER, POS tagging and Chunking. Furthermore, the encode method is also enriched with the *tags_type* parameter, so that we will be able to use it for POS tagging and Chunking tasks later on. An extra parameter *return_concat_results* is introduced in the *EvaluateModel* function to make it possible to retrieve predictions and labels per sentence (without concatenating them), in order to identify wrongly classified sentences in question 2. Moreover, the training process is moved under the *TrainModel* function that takes as parameters the model and the training dataloader, so that we can use it to train different models, such as in questions 1,4,5,6,7,8, with different datasets, such as in question 5.

1. The model’s performance on the test set is presented on the table below

Accuracy	Macro-average Accuracy
0.980	0.908

Table 1: The Model’s performance on the test set

2. Thereafter, we implement the *find_bad_example()* method which given actual labels and predictions of sentence tokens, finds a sentence with at least 10 tokens and at least 30% of its tokens wrongly classified.

The detailed predictions vs. labels comparison is presented in the table below

Token	Label	Prediction
NCAA	B-ORG	B-MISC
AMERICAN	O	B-MISC
FOOTBALL-OHIO	B-MISC	I-MISC
STATE	I-MISC	I-ORG
'S	O	O
PACE	B-PER	O
FIRST	O	O
REPEAT	O	O
LOMBARDI	B-MISC	B-MISC
AWARD	I-MISC	O
WINNER	O	O
.	O	O

Table 2: The labels vs. predictions comparison

The model misclassified 6 out of 12 tokens of the sentence 'NCAA AMERICAN FOOTBALL-OHIO STATE 'S PACE FIRST REPEAT LOMBARDI AWARD WINNER . '

The wrongly classified tokens are the 'NCAA', 'AMERICAN', 'FOOTBALL-OHIO', 'STATE', 'PACE', 'AWARD' and the correctly classified tokens are the following 'S', 'FIRST', 'REPEAT', 'LOMBARDI', 'WINNER' and '.' respectively.

To complete the second part of the 2nd question, we create three new functions that can handle data without annotated labels. The *encode_predict* function encodes the input sentence without requiring or returning any labels. Instead, it also returns the *encoding.word_ids()* so that we can later map the subtoken predictions to the correct words/tokens. Moreover, the *Predict* function performs a forward pass of the model to produce the predictions. Last but not least, *predict_sentence* encodes the input sentence and generates the predictions by using the previous 2 functions. Then, it maps the subtoken predictions to the correct token and prints a table with the predictions per token.

Next, we will give as input to the NER, the following sentence taken from *bbc.com*.

'Local governor Vyacheslav Gladkov said Russian forces were searching for "saboteurs", who he said had attacked Grayvoronsky district by the border.'

The table below contains the results according to the pronunciation.

Token	Prediction
Local	O
governor	O
Vyacheslav	B-PER
Gladkov	I-PER
said	O
Russian	B-MISC
forces	O
were	O
searching	O
for	O

"saboteurs",	O
who	O
he	O
said	O
had	O
attacked	O
Grayvoronsky	B-LOC
district	O
by	O
the	O
border.	O

Table 3: The tokens along with their predictions for our random sentence from *bbc.com*

The model correctly classifies most of the tokens. Two possible misclassifications are 'forces': 'I-MISC' (instead of 'O') and 'district': 'I-LOC' (instead of 'O').

3. The purpose of the *align_label* function is to convert the labels of a sentence, from string representation (e.g. "B_ORG") to label ids (e.g. 3) to create a suitable target format for the model. It also manages to mark the special tokens, and possible subtokens of a word (after the first one) with the special token -100, so that only one label corresponds to each word, and the loss and evaluation metrics are calculated correctly. More detailed:

Firstly, the *tokens.word_ids()* is called. This returns the id of the word that each token of a sentence originates from. This mapping is needed because the BERT tokenizer can split a word into more than one subtokens, and we need a way to know from which word (in the original sentence) each subtoken came from. Note that *tokens.word_ids()* sets the word id of the special tokens (i.e. [CLS]) to None. The *align_label* function sets the None word ids to -100 meaning that the target label id for the special tokens of the sentence is -100. Additionally, for each token that is not a special token, the function assigns the label id by mapping the word id of the token (*labels[word_idx]*) and then using *tagmap* to map the string label to the label id (*tagmap[labels[word_idx]]*). Although, this only happens for the first subtoken of each word (in case a word was split into more than one subtokens). This is achieved by the *elif* clause which checks whether the word id has changed since the previous iteration. For any other subtoken that is not the first subtoken of a word, the function again sets its label id to -100. Finally, according to the *huggingface* documentation, the token classification loss function ignores the special label id -100. Hence, all the special tokens and the subtokens that are not the first subtoken of a word, are ignored during the calculation of the loss, along with the corresponding model predictions. Also, labels and predictions in the positions that are marked with -100 are filtered out in the *EvaluateModel* function, so that the evaluation metrics are correctly calculated.

4. In order to freeze BERT in our model, we iterate through the bert parameters (*freezed_bert_model.bert.parameters()*) and we set the 'requires_grad' attribute of each parameter to False. Consequently, we create a new AdamW optimizer, and we set its params argument to the list of the model parameters that has *requires_grad == True*. This way, the optimizer will only optimize the parameters that have *requires_grad == True* i.e. the classifier layer's parameters, leaving the bert parameters as is (frozen).

The model's performance on the test set is presented on the table below.

Accuracy	Macro-average accuracy
0.853	0.21

Table 4: The Model's performance on the test set with learning rate equal to $1e-5$

It is notable that the model's performance is way worse than the performance of the model in the first task. This is probably due to the small learning rate that we use for both questions ($1e-5$), which although is suitable for the optimizing the BERT+classifier model, delays the optimization of the classification layer (with BERT frozen). A possible solution would be to increase the learning rate to a larger value (e.g. $1e-3$) and/or increase the number of epochs. Indeed, if we change the learning rate from $1e-5$ to $1e-3$, the model performs way better according to the results below.

Accuracy	Macro-average Accuracy
0.965	0.813

Table 5: The Model's performance on the test set by changing the learning rate to $1e-3$

Lastly, the table below contains the number of freezed (bert) parameters and the number of trainable (classification layer) parameters.

Freezed parameters	Trainable parameters
108891648	6921

Table 6: The total number of freezed and trainable parameters respectively

5. To train the model in the concatenation of the training and validation set, we first concatenate the *train_dataset* and the *valid_dataset*. Both datasets are lists of objects, where each object represents a training/validation example. Thus, we achieve this with a simple list concatenation: *train_valid_dataset* = *train_dataset* + *valid_dataset*. Furthermore, we create a new dataloader (*train_valid_loader*) with input dataset, the *train_valid_dataset*. Lastly, we train the model by passing the *train_valid_loader* to the *train_loader* parameter of the *TrainModel* method. It is important to note that validation metrics after each epoch are still calculated on the validation set, which is now also part of the training dataset and thus, these metrics should not be taken into consideration. Since training stopping criterion is fixed to 3 epochs, this does not raise a problem because validation metrics do not affect training in any way. It would however be a problem if we chose to use Early Stopping, where the number of epochs to train the model would be affected by the validation metrics.

The model's performance on the test it is presented on the next table.

Accuracy	Macro-average Accuracy
0.979	0.911

Table 7: The Model's performance on the test set by changing the learning rate to $1e-3$

Comparing to the model of the first question which was trained only on the original training set, the test accuracy is slightly lower, 0.979 (vs 0.980), but the test macro-accuracy has slightly improved 0.911 (vs 0.908).

6. To train the model for the POS tagging task, we first need to create new *tagmap* and *tagset* variables on the *pos_tags* labels. We achieve this by calling the *build_tags* method that we have constructed in the first cell, that takes as input the type of tags for which we want to create the *tagmap* and the *tagset* (i.e. pos tags). We then create new train, valid and test datasets. Again, we achieve this by using the modified encode method that takes the tag type (i.e. pos_tags) as a parameter and constructs the labels tensor based on that. Next, we create a new model with *num_labels=len(tagset)*, where the tagset has now changed to contain the pos tagging labels. Lastly, we create data loaders on the new pos tagging datasets and we train the model using the *TrainModel* method.

Now, the model's performance on the test set is presented on the table below.

Accuracy	Macro-average Accuracy
0.941	0.853

Table 8: The Model's performance on the test set for task 6

The model misclassified 3 out of 10 tokens of the sentence 'SOCCER – LATE GOALS GIVE JAPAN WIN OVER SYRIA .' according to the next table.

Token	Label	Prediction
SOCCER	NN	NN
-	:	:
LATE	JJ	NNP
GOALS	NNS	NNP
GIVE	VBP	NNP
JAPAN	NNP	NNP
WIN	NNP	NNP
OVER	IN	IN
SYRIA	NNP	NNP
.	.	.

Table 9: The labels vs predictions comparison

The wrongly classified tokens are the 'LATE', 'GOALS', 'GIVE' and the correctly classified tokens are the following 'SOCCER', '-', 'JAPAN', 'WIN', 'OVER', 'SYRIA' and '.' respectively.

Next, we will give as input to the POS tagging, again the following sentence taken from *bbc.com*.

'Local governor Vyacheslav Gladkov said Russian forces were searching for "saboteurs", who he said had attacked Grayvoronsky district by the border.'

The table below contains the results.

Token	Prediction
Local	JJ
governor	NN
Vyacheslav	NNP
Gladkov	NNP
said	VBD
Russian	JJ
forces	NNS
were	VBD
searching	VBG
for	IN
"saboteurs",	"
who	WP
he	PRP
said	VBD
had	VBD
attacked	VCN
Grayvoronsky	NNP
district	NN
by	IN
the	DT
border.	NN

Table 10: The tokens along with their predictions for the POS tagging task

The model classifies correctly all the tokens, except for the token “saboteurs”, which is classified as “ (quote) instead of NNS (Noun, plural).

It is important to note that although BERT’s tokenizer is able to split difficult tokens like “saboteurs”, and make correct predictions, in this example the model’s prediction is mostly affected by the first character ‘’, which can be explained by the cleaner data that the model was trained on. Better initial tokenization of the word “saboteurs”, into 4 separate tokens [‘’, ‘saboteurs’, ‘’, ‘,’], would probably yield better predictions, although our sentence of words was picked randomly for education purposes. Similarly for the token ‘border.’.

7. To train the model for the text chunking task, we first need to create new tagmap and tagset variables on the chunk_tags labels. We achieve this by calling the *build_tags* method that we constructed in the first cell, that takes as input the type of tags for which we want to create the tagmap and tagset (i.e. chunk_tags). Next, we create new train, valid and test datasets. Again, we achieve this by using the modified encode method that takes the tag type (i.e. chunk_tags) as a parameter and constructs the labels tensor based on that. Then, we create a new model with *num_labels=len(tagset)*, where the tagset has now changed to contain the chunking labels. Lastly, we create data loaders on the new data chunking datasets and we train the model using the *TrainModel* method. The model’s performance on the test set is presented on the table below, respectively.

Accuracy	Macro-average Accuracy
0.951	0.586

Table 11: The Model’s performance on the test set for task 7

The model misclassified 4 out of 12 tokens of the sentence ‘Aly Ashour 7 , 56 penalty , Mohamed Ouda 24 , 73’ according to the next table.

Token	Label	Prediction
Aly	B-ADVP	B-NP
Ashour	B-NP	I-NP
7	I-NP	I-NP
,	O	I-NP
56	B-NP	I-NP
penalty	I-NP	I-NP
,	O	O
Mohamed	B-NP	B-NP
Ouda	I-NP	I-NP
24	I-NP	I-NP
,	I-NP	I-NP
73	I-NP	I-NP

Table 12: The labels vs predictions comparison

The wrongly classified tokens are the ‘Aly’, ‘Ashour’, ‘,’’, ‘56’ and the correctly classified tokens are the following ‘7’, ‘penalty’, ‘,’’, ‘Mohamed’, ‘Ouda’, ‘24’, ‘,’ And ‘73’ respectively.

Next, we will give as input to the text chunking tagging, again the same sentence from *bbc.com*.

‘Local governor Vyacheslav Gladkov said Russian forces were searching for “saboteurs”, who he said had attacked Grayvoronsky district by the border.’

The next table contains the requested results.

Token	Prediction
Local	B-NP

governor	I-NP
Vyacheslav	I-NP
Gladkov	I-NP
said	B-VP
Russian	B-NP
forces	I-NP
were	B-VP
searching	I-VP
for	B-PP
"saboteurs",	O
who	B-NP
he	B-NP
said	B-VP
had	B-VP
attacked	I-VP
Grayvoronsky	B-NP
district	I-NP
by	B-PP
the	B-NP
border.	I-NP

Table 13: The tokens along with their predictions for the text chunking task

The model classifies correctly all the tokens except for the token “'saboteurs'”, which is classified as ‘O’ instead of B-NP. Of course, with better initial tokenization, the model would probably classify this token better.

8. To change the model from BERT to *roberta-base*, we first have to import the model *RobertaForTokenClassification* and its tokenizer *RobertaTokenizerFast*. Next, we just initialize the tokenizer and the model by specifying the Roberta version to be ‘*roberta_base*’, as follows:

```
roberta_version = 'roberta-base'
```

```
tokenizer = RobertaTokenizerFast.from_pretrained(roberta_version)
```

```
model = RobertaForTokenClassification.from_pretrained(roberta_version, num_labels = len(tagset))
```

The rest of the code remains the same. Now, the model’s performance on the test set appears on the table below.

Accuracy	Macro-average Accuracy
0.983	0.926

Table 14: The Model’s performance on the test set for question 8

Comparing to the bert-base-uncased model of the first question, the roberta-base model performs better both in accuracy: 0.983 (vs 0.980) and in macro-average accuracy as well: 0.926 (vs 0.908).

9. Based on the findings of the questions 2, 6 and 7, we will use the following 3 sentences to test the performance of ChatGPT in zero-shot token classification:

NER: ‘NCAA AMERICAN FOOTBALL-OHIO STATE 'S PACE FIRST REPEAT LOMBARDI AWARD WINNER.’

POS tagging: ‘SOCCER - LATE GOALS GIVE JAPAN WIN OVER SYRIA.’

Chunking: *'Aly Ashour 7 , 56 penalty , Mohamed Ouda 24 , 73'*

We will also test the following **sentence from BBC** in **all tasks**:

'Local governor Vyacheslav Gladkov said Russian forces were searching for "saboteurs", who he said had attacked Grayvoronsky district by the border.'

For the NER task, we will include a brief explanation of each class to the prompt. In POS tagging and chunking tasks which have a large number of labels, we will let ChatGPT understand the meaning of each label solely by their name and the context of the task (POS tagging or text chunking)

NER / zero-shot

Prompt:

Act as a Named Entity Recognition model (NER) and classify each token of the input sentence, into one of the following classes:

["B-ORG", "I-ORG", "B-PER", "I-PER", "B-LOC", "I-LOC", "B-MISC", "I-MISC", "O"]

Explanation of the classes:

B-ORG: Beginning token of an organization entity

I-ORG: Inside token of an organization entity

B-PER: Beginning token of a person entity

I-PER: Inside token of a person entity

B-LOC: Beginning token of a location entity

I-LOC: Inside token of a location entity

B-MISC: Beginning token of a miscellaneous entity

I-MISC: Inside token of a miscellaneous entity

O: Outside of any entity

Split the input sentence into tokens with space as delimiter.

The output should have the following format:

```
{  
  "token": "class"  
}
```

input sentence: NCAA AMERICAN FOOTBALL-OHIO STATE 'S PACE FIRST REPEAT LOMBARDI AWARD WINNER .

Response:

```
{ "NCAA": "B-ORG", "AMERICAN": "I-ORG", "FOOTBALL-OHIO": "I-ORG", "STATE": "I-ORG", "S": "O", "PACE": "O", "FIRST": "O", "REPEAT": "O", "LOMBARDI": "B-MISC", "AWARD": "I-MISC", "WINNER": "I-MISC", ".": "O" }
```

Evaluation:

ChatGPT misclassified the following tokens (5 out of 12):

Token	Label	Prediction
AMERICAN	O	I-ORG
FOOTBALL-OHIO	B-MISC	I-ORG
STATE	I-MISC	I-ORG
PACE	B-PER	O
WINNER	O	I-MISC

Table 15: The misclassification of ChatGPT for NER / zero-shot

Response with BBC sentence as input:

```
{ "Local": "O", "governor": "O", "Vyacheslav": "B-PER", "Gladkov": "I-PER", "said": "O", "Russian": "B-MISC", "forces": "I-MISC", "were": "O", "searching": "O", "for": "O", "saboteurs": "O", "who": "O", "he": "O", "said": "O", "had": "O", "attacked": "O", "Grayvoronsky": "B-LOC", "district": "I-LOC", "by": "O", "the": "O", "border": "O" }
```

Evaluation:

Most classifications seem correct. “district” could also be “O” instead of “I-LOC”.

POS tagging / zero-shot**Prompt:**

Act as a POS tagging model and classify each token of the input sentence, into one of the following classes:

```
["NNP", "NN", "CD", "IN", "DT", "JJ", "NNS", "VBD", ".", ",", "VB", "VBN", "RB", "CC", "TO", "PRP", "(", ")", "VBG", "VBZ", ":", "''", "POS", "PRP$", "VBP", "MD", "NNPS", "RP", "WP", "WDT", "SYM", "$", "WRB", "JJR", "JJS", "FW", "RBR", "EX", "''", "RBS", "PDT", "UH", "WP$", "LS", "NN|SYM"]
```

Split the input sentence into tokens with space as delimiter.

The output should have the following format:

```
{
  "token": "class"
}
```

input sentence: SOCCER - LATE GOALS GIVE JAPAN WIN OVER SYRIA .

Response:

```
{ "SOCCER": "NN", "-": "SYM", "LATE": "JJ", "GOALS": "NNS", "GIVE": "VB",
  "JAPAN": "NNP", "WIN": "VB", "OVER": "IN", "SYRIA": "NNP", ".": "." }
```

Evaluation:

ChatGPT misclassified the following tokens (3 out of 10):

Token	Label	Prediction
-	:	SYM
GIVE	VBP	VB
WIN	NNP	VB

Table 16: The misclassification of ChatGPT for POS / zero-shot

Response with BBC sentence as input:

```
{ "Local": "NNP", "governor": "NN", "Vyacheslav": "NNP", "Gladkov": "NNP", "said":
  "VBD", "Russian": "JJ", "forces": "NNS", "were": "VBD", "searching": "VBG", "for": "IN",
  ""saboteurs"": "NN", "who": "WP", "he": "PRP", "said": "VBD", "had": "VBD", "attacked":
  "VBN", "Grayvoronsky": "NNP", "district": "NN", "by": "IN", "the": "DT", "border": "NN",
  ".": "." }
```

Evaluation:

ChatGPT misclassified the following tokens(1 out of 21):

Token	Label	Prediction
Local	JJ	NNP

Table 17: The misclassification of ChatGPT for POS / zero-shot for BBC sentenc

Chunking / zero-shot**Prompt:**

Act as a text chunking model and classify each token of the input sentence, into one of the following classes:

```
["I-NP", "B-NP", "O", "B-PP", "B-VP", "I-VP", "B-ADVP", "B-SBAR", "B-ADJP", "B-PRT",
  "I-ADJP", "I-ADVP", "I-PP", "B-INTJ", "I-CONJP", "B-LST", "B-CONJP", "I-SBAR", "I-
  LST", "I-INTJ"]
```

Split the input sentence into tokens with space as delimiter.

The output should have the following format:

```
{
  "token": "class"
}
```

input sentence: Aly Ashour 7 , 56 penalty , Mohamed Ouda 24 , 73

Response:

{ "Aly": "I-NP", "Ashour": "I-NP", "7": "O", ",": "O", "56": "O", "penalty": "O", ",": "O", "Mohamed": "I-NP", "Ouda": "I-NP", "24": "O", ",": "O", "73": "O" }

Evaluation:

ChatGPT misclassified the following tokens (9 out of 12):

Token	Label	Prediction
Aly	B-ADVP	I-NP
Ashour	B-NP	I-NP
7	I-NP	O
56	B-NP	O
penalty	I-NP	O
Mohamed	B-NP	I-NP
24	I-NP	O
,	I-NP	O
73	I-NP	O

Table 18: The misclassification of ChatGPT for Chunking / zero-shot

Response with BBC sentence as input:

{ "Local": "B-NP", "governor": "I-NP", "Vyacheslav": "I-NP", "Gladkov": "I-NP", "said": "O", "Russian": "B-NP", "forces": "I-NP", "were": "O", "searching": "B-VP", "for": "B-PP", "": "O", "saboteurs": "O", "": "O", ",": "O", "who": "B-SBAR", "he": "O", "said": "O", "had": "O", "attacked": "B-VP", "Grayvoronsky": "B-NP", "district": "I-NP", "by": "B-PP", "the": "B-NP", "border": "I-NP", ".": "O" }

Evaluation:

ChatGPT misclassified the following tokens (9 out of 21):

Token	Label	Prediction
said	B-VP	O
were	B-VP	O
searching	I-VP	B-VP
saboteurs	B-NP	O
who	B-NP	B-SBAR
he	B-NP	O
said	B-VP	O
had	B-VP	O
attacked	I-VP	B-VP

Table 19: The misclassification of ChatGPT for Chunking / zero-shot for BBC sentence

10. We will use the same sentences as in Question 9, but this time we will include 2 examples from the training set in the prompt to help ChatGPT gain a better understanding of the task (few-shot).

NER / few-shot

Prompt:

Act as a Named Entity Recognition model (NER) and classify each token of the input sentence, into one of the following classes:

["B-ORG", "I-ORG", "B-PER", "I-PER", "B-LOC", "I-LOC", "B-MISC", "I-MISC", "O"]

Explanation of the classes:

B-ORG: Beginning token of an organization entity

I-ORG: Inside token of an organization entity

B-PER: Beginning token of a person entity

I-PER: Inside token of a person entity

B-LOC: Beginning token of a location entity

I-LOC: Inside token of a location entity

B-MISC: Beginning token of a miscellaneous entity

I-MISC: Inside token of a miscellaneous entity

O: Outside of any entity

Split the input sentence into tokens with space as delimiter.

The output should have the following format:

```
{  
  "token": "class"  
}
```

Example 1: The European Commission said on Thursday it disagreed with German advice to consumers to shun British lamb until scientists determine whether mad cow disease can be transmitted to sheep .

Answer: {"The": "O", "European": "B-ORG", "Commission": "I-ORG", "said": "O", "on": "O", "Thursday": "O", "it": "O", "disagreed": "O", "with": "O", "German": "B-MISC", "advice": "O", "to": "O", "consumers": "O", "shun": "O", "British": "B-MISC", "lamb": "O", "until": "O", "scientists": "O", "determine": "O", "whether": "O", "mad": "O", "cow": "O", "disease": "O", "can": "O", "be": "O", "transmitted": "O", "sheep": "O", ".": "O"}

Example 2: Australian coach Geoff Marsh said he was impressed with the competitiveness of the opposition .

input sentence: NCAA AMERICAN FOOTBALL-OHIO STATE 'S PACE FIRST REPEAT LOMBARDI AWARD WINNER .

```
{ "NCAA": "B-ORG", "AMERICAN": "I-ORG", "FOOTBALL-OHIO": "I-ORG", "STATE":  
"I-ORG", "S": "O", "PACE": "B-PER", "FIRST": "O", "REPEAT": "O", "LOMBARDI": "O",  
"AWARD": "O", "WINNER": "O", ".": "O" }
```

ChatGPT misclassified the following tokens (5 out of 12):

Token	Label	Prediction
AMERICAN	O	I-ORG
FOOTBALL-OHIO	B-MISC	I-ORG
STATE	I-MISC	I-ORG
LOMBARDI	B-MISC	O
AWARD	I-MISC	O

Response with BBC sentence as input:

Evaluation:

POS tagging / few-shot

Prompt:

["NNP", "NN", "CD", "IN", "DT", "JJ", "NNS", "VBD", ".", ",", "VB", "VBN", "RB", "CC", "TO", "PRP", "(", ") ", "VBG", "VBZ", ":", "''", "POS", "PRP\$", "VBP", "MD", "NNPS", "RP", "WP", "WDT", "SYM", "\$", "WRB", "JJR", "JJS", "FW", "RBR", "EX", "''", "RBS", "PDT", "UH", "WP\$", "LS", "NN|SYM"]

Split the input sentence into tokens with space as delimiter.

The output should have the following format:

 $\{$

```
"token": "class"
}
```

Example 1: The European Commission said on Thursday it disagreed with German advice to consumers to shun British lamb until scientists determine whether mad cow disease can be transmitted to sheep .

Answer: {"The": "DT", "European": "NNP", "Commission": "NNP", "said": "VBD", "on": "IN", "Thursday": "NNP", "it": "PRP", "disagreed": "VBD", "with": "IN", "German": "JJ", "advice": "NN", "to": "TO", "consumers": "NNS", "shun": "VB", "British": "JJ", "lamb": "NN", "until": "IN", "scientists": "NNS", "determine": "VBP", "whether": "IN", "mad": "JJ", "cow": "NN", "disease": "NN", "can": "MD", "be": "VB", "transmitted": "VBN", "sheep": "NN", ".": "."}

Example 2: Australian coach Geoff Marsh said he was impressed with the competitiveness of the opposition .

Answer: {"Australian": "JJ", "coach": "NN", "Geoff": "NNP", "Marsh": "NNP", "said": "VBD", "he": "PRP", "was": "VBD", "impressed": "VBN", "with": "IN", "the": "DT", "competitiveness": "NN", "of": "IN", "opposition": "NN", ".": "."}

input sentence: SOCCER - LATE GOALS GIVE JAPAN WIN OVER SYRIA .

Response:

{"SOCCER": "NN|SYM", "-": ".", "LATE": "JJ", "GOALS": "NNS", "GIVE": "VBP", "JAPAN": "NNP", "WIN": "NNP", "OVER": "IN", "SYRIA": "NNP", ".": "."}

Evaluation:

ChatGPT misclassified the following token (1 out of 10):

Token	Label	Prediction
SOCCER	NN	NN SYM

Table 21: The misclassification of ChatGPT for POS / few-shot

Response with BBC sentence as input:

{"Local": "JJ", "governor": "NN", "Vyacheslav": "NNP", "Gladkov": "NNP", "said": "VBD", "Russian": "JJ", "forces": "NNS", "were": "VBD", "searching": "VBG", "for": "IN", "saboteurs": "NN|SYM", "who": "WP", "he": "PRP", "said": "VBD", "had": "VBD", "attacked": "VBN", "Grayvoronsky": "JJ", "district": "NN", "by": "IN", "the": "DT", "border": "NN", ".": "."}

Evaluation:

ChatGPT misclassified the following token (1 out of 21):

Token	Label	Prediction
"saboteurs"	NNS	NN SYM

Grayvoronsky	NNP	JJ
--------------	-----	----

Table 22: The misclassification of ChatGPT for POS / few-shot for BBC sentence

Chunking / few-shot

Prompt:

Act as a chunking model and classify each token of the input sentence, into one of the following classes:

["I-NP", "B-NP", "O", "B-PP", "B-VP", "I-VP", "B-ADVP", "B-SBAR", "B-ADJP", "B-PRT", "I-ADJP", "I-ADVP", "I-PP", "B-INTJ", "I-CONJP", "B-LST", "B-CONJP", "I-SBAR", "I-LST", "I-INTJ"]

Split the input sentence into tokens with space as delimiter.

The output should have the following format:

```
{
  "token": "class"
}
```

Example 1: The European Commission said on Thursday it disagreed with German advice to consumers to shun British lamb until scientists determine whether mad cow disease can be transmitted to sheep .

Answer: {"The": "B-NP", "European": "I-NP", "Commission": "I-NP", "said": "B-VP", "on": "B-PP", "Thursday": "B-NP", "it": "B-NP", "disagreed": "B-VP", "with": "B-PP", "German": "B-NP", "advice": "I-NP", "to": "B-PP", "consumers": "B-NP", "shun": "I-VP", "British": "B-NP", "lamb": "I-NP", "until": "B-SBAR", "scientists": "B-NP", "determine": "B-VP", "whether": "B-SBAR", "mad": "B-NP", "cow": "I-NP", "disease": "I-NP", "can": "B-VP", "be": "I-VP", "transmitted": "I-VP", "sheep": "B-NP", ".": "O" }

Example 2: Australian coach Geoff Marsh said he was impressed with the competitiveness of the opposition .

Answer: {"Australian": "B-NP", "coach": "I-NP", "Geoff": "I-NP", "Marsh": "I-NP", "said": "B-VP", "he": "B-NP", "was": "B-VP", "impressed": "I-VP", "with": "B-PP", "the": "B-NP", "competitiveness": "I-NP", "of": "B-PP", "opposition": "I-NP", ".": "O" }

input sentence: Aly Ashour 7 , 56 penalty , Mohamed Ouda 24 , 73

Response:

{ "Aly": "B-NP", "Ashour": "I-NP", "7": "O", ",": "O", "56": "O", "penalty": "O", ",": "O", "Mohamed": "B-NP", "Ouda": "I-NP", "24": "O", ",": "O", "73": "O" }

Evaluation:

ChatGPT misclassified the following tokens (8 out of 12). For:

Token	Label	Prediction
Aly	B-ADVP	B-NP
Ashour	B-NP	I-NP
7	I-NP	O
56	B-NP	O
penalty	I-NP	O
24	I-NP	O
,	I-NP	O
73	I-NP	O

Table 23: The misclassification of ChatGPT for Chunking / few-shot

Response with BBC sentence as input:

{ "Local": "B-NP", "governor": "I-NP", "Vyacheslav": "I-NP", "Gladkov": "I-NP", "said": "B-VP", "Russian": "B-NP", "forces": "I-NP", "were": "B-VP", "searching": "I-VP", "for": "B-PP", "": "O", "saboteurs": "B-NP", "": "O", ",": "O", "who": "B-NP", "he": "B-NP", "said": "B-VP", "had": "B-VP", "attacked": "I-VP", "Grayvoronsky": "B-NP", "district": "I-NP", "by": "B-PP", "the": "B-NP", "border": "I-NP", ".": "O" }

Evaluation:

ChatGPT classified all the tokens correctly.