Deliverable 2 - NLP

Albert Garcia, Jordi Solé, Michael DePass, Pere Gilabert y Alejandro González

1. High Level Summary

Named-entity recognition (NER) is a subtask of information extraction that seeks to tag every word in a document into some predefined categories or tags. For the purpose of this deliverable, our tags are geo (Geographical Entity), org (Organization), per (Person), gpe (Geopolitical Entity), tim (Time indicator), art (Artifact), eve (Event), nat (Natural Phenomenon), and no tag (O). Furthermore, we tested three different models to perform this task: Hidden Markov Model (HMM), Structured Perceptron, and BERT (Bidirectional encoder representations for transformers). In addition, we applied different preprocessing and feature engineering techniques to boost performance. BERT slightly outperformed the Structured Perceptron and HMM models. The test accuracies were 0.961, 0.955, and 0.925 respectively.

2. Methods

2.0 Preprocessing

NaN values in the raw data were replaced via forward fill upon import. The data was grouped by sentence ID and transformed into a pair of lists (one containing the words and one containing the tags). Word2Pos and tag2Pos dictionaries were used to vectorize the sentences. Data was split into train and test sets. TEST_SENTENCES were defined in a list.

2.1 HMM

Hidden Markov Model (HMM) is a model defined by a probability distribution over states and a probability distribution that generates words given the state. In the context of NER systems, the main idea behind the use of HMM models is that it is language independent and we can apply this system for any language domain. The HMM is quick to train and is a common baseline for POS and NER tasks. Posterior decoding was used.

2.2 Structured Perceptron

The second model to be tested was the Structured Perceptron, another commonly used algorithm for structured prediction. Viterbi decoding was used. The training set was further split into training and validation to aid with model selection.

2.2.1 Feature Engineering

11 extra features were implemented in an attempt to boost performance. These features check for: first letter capitalization, all caps, -ed endings, prefixes up to 3 characters in length, no vowels, only vowels, abbreviations, special characters, numeric values, periods, and hyphens. Suffixes up to 3 characters in length were considered as well but this feature was already implemented.

These features were selected since we suspected they would aid in the classification of one or more of the hidden state labels. It is intuitive, for example, that putting extra

emphasis on capitalization would help distinguish the hidden states of words that are typically capitalized from the others (we suspect names of organizations are more likely to be capitalized than natural phenomenon, for example). Similarly, words that contain numeric values and special characters are more likely to be time indicators.

Features were tested one by one for 10 epochs with a stopping criterion of 5 epochs. The "All (specific)" feature involved processing all features implemented in a word or character specific way. With this implementation, the "Capital" feature in the non-specific case, for example, would become many features in the specific case. There would be one added feature for every detected capital letter (that occurred as the first letter in a word). This produced many more parameters.

Extra Features	Train	Validation
None	0.956	0.952
Capital	0.959	0.953
All Caps	0.958	0.938
"-ed" Ending	0.952	0.949
Prefixes	0.955	0.949
No Vowels	0.944	0.947
All Vowels	0.956	0.952
Abbreviation	0.957	0.953
Special Chars	0.955	0.947
Numeric	0.952	0.948
Period	0.948	0.944
Hyphen	0.951	0.950
All (specific)	0.967	0.953
All (non-specific)	0.958	0.955

Several of the extra features yielded small improvements in the performance of the Structured Perceptron. The "All (specific)" implementation yielded the highest training accuracy which is expected due to the huge increase in the number of trainable parameters. The validation accuracy, however, was highest in the "All (non-specific)" case. These features are defined in extended_features.py which can be found in the following Github repo in skeq > sequences > extended_features.py.

2.2.2 Edit Distance

Edit/Levenshtein distance was used to evaluate the distances between words. Out of vocabulary words were replaced with the nearest in-vocabulary word. A BK tree was used to prune the search space to more quickly determine if a given word was part of the corpus. This is much faster than comparing a word to every other word in the corpus.

2.2.3 Early stopping

Early stopping prevented overfitting and hastened training. The perceptron stopped training after 10 epochs without improvement in accuracy. The model with the highest validation accuracy was saved.

2.3 BERT

BERT is a neural network-based technique for natural language processing pre-training. BERT's key technical innovation is applying the bidirectional training of a Transformer to language modelling. This novel neural network architecture automatically detects word

and character level features using a hybrid bidirectional LSTM and CNN architecture, eliminating the need for most feature engineering.

3. Results

3.1 Performance metrics

The first performance metric to consider is overall accuracy:

	HMM	Structured Perceptron	BERT
Train Accuracy	0.972	0.969	0.990
Test Accuracy	0.925	0.955	0.961

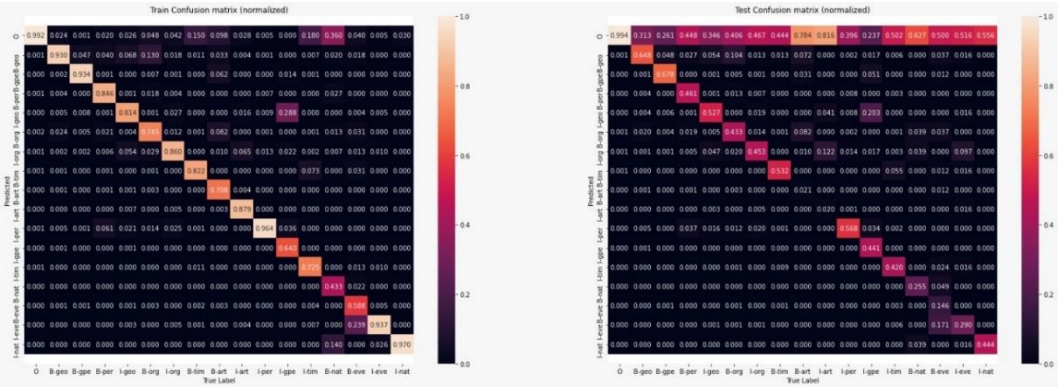
In terms of overall classification accuracy, BERT performed the best with an overall accuracy of 0.961 while the Structured Perceptron performed slightly worse with an accuracy of 0.955 (-0.006 compared to BERT). The HMM performed the worst with an accuracy of 0.925 (-0.036 compared to BERT). The HMM seemed to overfit the most as evidenced by the comparatively large difference between its train and test accuracies.

Another way to evaluate performance is to look at the number of complete sentences that were tagged correctly. There were 35,971 train sentences and 11,988 test sentences:

	HMM	Structured Perceptron	BERT
Train Sentences	24,460	23,244	30,959
Test Sentences	5,964	6,602	8,026

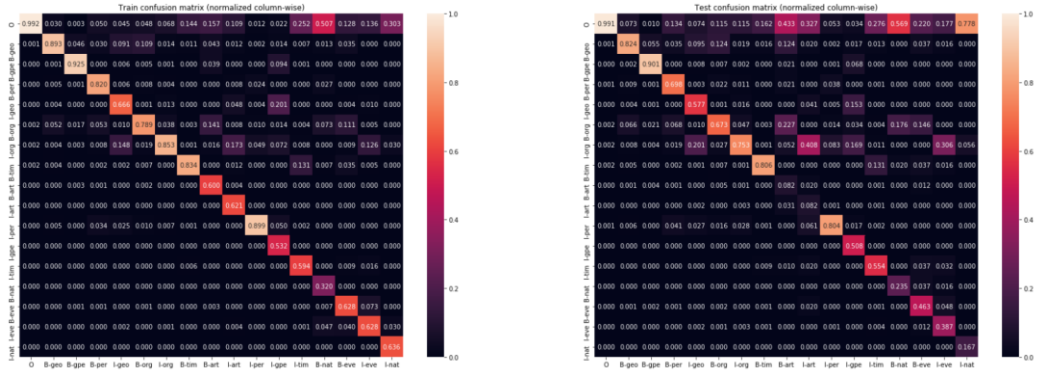
3.2 Confusion matrices

3.2.1 HMM



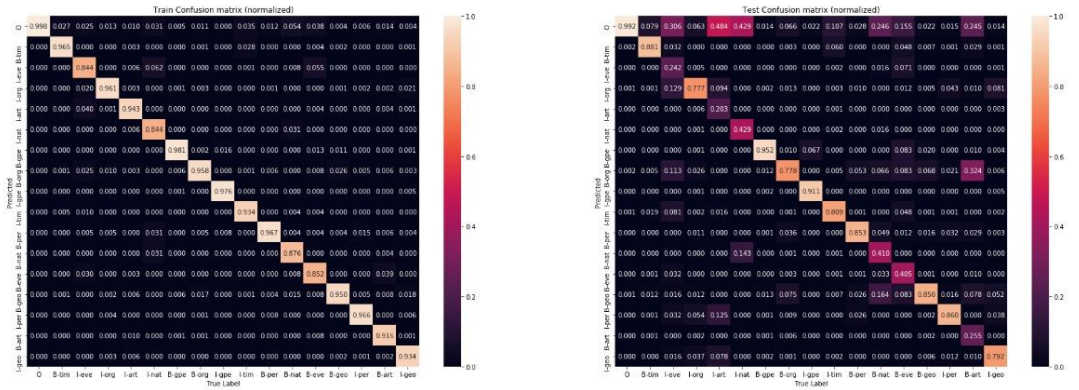
Notice how in the test set a lot of tags have been incorrectly predicted as ‘O’. From this we can say that HMM tends to predict ‘O’ in cases it should not. This could be due to the way HMM is trained and due to the huge imbalance in the classes (the number of words with tag ‘O’ is way larger than any other tag).

3.2.2 Structured Perceptron



Compared to the HMM model, the test confusion matrix no longer shows a large amount of incorrectly classified 'O' tags. The tags 'I-art' and 'B-art' are still not correctly identified (in both HMM and SP). We can also detect some tags in which there is some confusion with the tag 'O' like the tags 'B-nat' and 'I-nat'. Additionally, the pairs of tags 'I-art'-'I-org' as well as the pair 'I-eve'-'I-org' are confused to some extent.

3.2.3 BERT



Compared to the HMM and Structured Perceptron models, the test confusion matrix for the BERT model shows the least amount of incorrectly classified 'O' tags. In addition, we can detect some tags in which there is some confusion with the tag 'O' like the tags 'I-nat' and 'I-art'. Additionally, the pairs of tags 'B-art'-'B-org' are confused to some extent.

3.3 Test Sentences Check

We will now test each of the algorithms on each of the test sentences.

3.3.1 HMM

It is clear that the unbalanced dataset is posing significant difficulty for the HMM. Since the vast majority of tags are 'O', the model is overpredicting 'O', tag in an attempt to maximize accuracy.

```
The\O programmers\O from\O Barcelona\B-geo might\O write\O a\O sentence\O without\O a\O spell\O checker\O .\O
The\O programmers\O from\O Barchelona\O cannot\O write\O a\O sentence\O without\O a\O spell\O checker\O .\O
Jack\B-per London\B-geo went\O to\O Parris\O .\O
Jack\B-per London\B-geo went\O to\O Paris\B-geo .\O
We\O never\O though\O Microsoft\B-org would\O become\O such\O a\O big\O company\O .\O
We\O never\O though\O Microsof\O would\O become\O such\O a\O big\O company\O .\O
The\O president\O of\O U.S.A\B-org though\O they\O could\O win\O the\O war\O .\O
The\O president\O of\O the\O United\B-geo States\I-geo of\I-geo America\I-geo though\O they\O could\O win\O the\O war\O .\O
The\O king\O of\O Saudi\B-geo Arabia\I-geo wanted\O total\O control\O .\O
Robin\O does\O not\O want\O to\O go\O to\O Saudi\O Arabia\O .\O
Apple\B-org is\O a\O great\O company\O .\O
I\O really\O love\O apples\O and\O oranges\O .\O
```

3.3.2 Structured Perceptron

The results obtained for the Structured Perceptron seem a bit better than the HMM. Implementing the edit distance for words out of corpus we have been able to correctly tag “Barchelona” with the same tag as the corresponding closest word, Barcelona. The same goes for “Parris” and “Microsof”. Overall, they have all been correctly tagged except for some specific words like 'Robin' word being tagged as 'O'.

```
The/O programmers/O from/O Barcelona/B-geo might/O write/O a/O sentence/O without/O a/O spell/O checker/O ./O
The/O programmers/O from/O Barchelona/B-geo cannot/O write/O a/O sentence/O without/O a/O spell/O checker/O ./O
Jack/B-per London/I-per went/O to/O Parris/B-geo ./O
Jack/B-per London/I-per went/O to/O Paris/B-geo ./O
We/O never/O though/O Microsoft/B-org would/O become/O such/O a/O big/O company/O ./O
We/O never/O though/O Microsof/B-org would/O become/O such/O a/O big/O company/O ./O
The/O president/O of/O U.S.A/B-org though/O they/O could/O win/O the/O war/O
The/O president/O of/O the/O United/B-geo States/I-geo of/O America/B-geo though/O they/O could/O win/O the/O war/O
The/O king/O of/O Saudi/B-org Arabia/I-org wanted/O total/O control/O ./O
Robin/O does/O not/O want/O to/O go/O to/O Saudi/B-org Arabia/I-org ./O
Apple/B-org is/O a/O great/O company/O ./O
I/O really/O love/O apples/O and/O oranges/O ./O
```

3.3.3 BERT

We can see that all the results are correct except that U.S.A should be B-geo I-geo as is also the case with the Structured Perceptron. But now, we are correctly tagging “Robin” as a name.

```
The\O programmers\O from\O Barcelona\B-geo might\O write\O a\O sentence\O without\O a\O spell\O checker\O .\O
The\O programmers\O from\O Barchelona\B-geo cannot\O write\O a\O sentence\O without\O a\O spell\O checker\O .\O
Jack\B-per London\I-per went\O to\O Parris\B-geo .\O
Jack\B-per London\I-per went\O to\O Paris\B-geo .\O
We\O never\O though\O Microsoft\B-org would\O become\O such\O a\O big\O company\O .\O
We\O never\O though\O Microsof\B-org would\O become\O such\O a\O big\O company\O .\O
The\O president\O of\O U\B-org .\B-org S\B-org .\B-org A\B-org though\O they\O could\O win\O the\O war\O
The\O president\O of\O the\O United\B-geo States\I-geo of\I-geo America\I-geo though\O they\O could\O win\O the\O war\O
The\O king\O of\O Saudi\B-geo Arabia\I-org wanted\O total\O control\O .\O
Robin\B-per does\O not\O want\O to\O go\O to\O Saudi\B-geo Arabia\I-geo .\O
Apple\B-org is\O a\O great\O company\O .\O
I\O really\O love\O apples\O and\O oranges\O .\O
```

4. Conclusion

From the previous results we can conclude that the best model as evidenced by the test accuracy is the BERT model with 0.961. The Structured Perceptron is not far off, at 0.955. Whereas for the simpler model, the HMM, we obtained the lowest test accuracy (0.925) as we initially thought it would happen. It, however, trained much quicker than the more sophisticated models.

These results are partly because we did not implement any new features for the HMM, whereas for the Structured Perceptron we used BK-Trees to detect words out of the corpus and Edit Distances to get the closest word from the unseen word in training. Moreover, we used a set of extended features to further boost the accuracy for the Structured Perceptron. Hence, some misclassifications by the HMM were classified correctly by the Structured Perceptron and by BERT.

Despite investigating and implementing a large set of extra features for the Structured Perceptron, BERT still managed to outperform it. This speaks to the power and utility of BERT and other neural network-based models.