WNUT17 Shared Task - Analysis of Language Models on The Task of Named Entity Recognition

Andrew Georgiou
Kings College
apjg4@cam.ac.uk

1 Introduction

Named Entity Recognition (NER) is a valuable task in the field of Natural Language Processing, it is the process of identifying and categorizing key entities within a given piece of text.This can enable the use of these classified entities in tasks such as search engine optimisation, content recommendation systems or pathology report key data extraction.
Current state of the art in NER by Wang (2021) consists of using Neural Architecture Search (NAS) to find better concatenations of embeddings for structured prediction tasks. Previous state-of-the-art approaches would use stand alone pre-trained contextualised embeddings such as BERT, ELMo or XLNet but these new methods utilise concatenation of a mixture of both pretrained contextualised and non-contextualised embeddings (such as GloVe or word2vec) which NAS attempts to search for the best model architecture of this type.
In this paper, we introduce, discuss and compare the performance of deep learning and regression methods for the task of Named Entity Recognition with my proposal for an improvement in performance over non-neural network methods through the use of the stand alone pretrained language models which show excellent results in the task of NER along with supplemental training data from *OntoNotes 5.0*
Unlike the task presented in WNUT-2017 we do not intend to identify each class of Named Entities but identify Beginning, Inside and Outside (BIO) tags of entities which proposes an easier task but allows for more interesting analysis of predictions to identify errors in model classification.

1.1 Embeddings

fastText (Joulin et al., 2016), GloVe (Pennington et al., 2014) and word2vec (Mikolov et al., 2013) are examples of non-contextualised embeddings which encode continuous vector representation of each word. They are encoded in a global vocabulary of words, each word shares a static representation which means a single word is always represented by the same vector regardless of its sentence based context. One such method of computing word embeddings is by using the skip-gram objective function which is seen in word2vec to sum the log probabilities of surrounding *n* words of a given target word w_t.

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T}\sum_{-n\leq j\leq n,\neq 0} \log p\left(w_{j+1} \mid w_t\right)$$

This differs to contextual embeddings such as ELMo (Peters et al., 2018) or BERT (Vaswani et al., 2017) which move past global representation and begin to embed each word based upon its context, allowing for greater knowledge encoding and therefore higher performance in tasks where context matters, such as NER.

1.2 Language Models

ELMo is the precursor to BERT, it uses a bi-directional LSTM model where at each layer it encodes left-to-right and right-to-left context, it then concatenate each of these hidden layers, multiply each vector by a weight based on the task and sum the weighted vectors. This concatenation is the reason ELMo does not have true contextual understanding since it cannot simultaneously account for both left and right context of a given word.
BERT differs in training objective to ELMo, introduced by Vaswani et al. (2017), BERT uses the transformer architecture, an encoder-decoder network that uses self-attention on the encoder and attention on the decoder. These models can contain up to 340 million parameters which when pretrained on the English Wikipedia corpora and the BookCorpus (Zhu et al., 2015) containing 4.4 mil-

Feature Excluded	Precision	Recall	F1
None	23.1	42.3	29.9
POS Tag	23.1	42.3	29.9
Proper Noun	19.3	19.5	19.4
Title Case	21.4	44.8	29.0
Start/End Case	22.9	42.8	29.8
Previous POS	23.2	42.3	30.0
Next POS	23.1	42.3	29.9
Hashtag	23.1	42.4	29.9
Abbreviation	**23.3**	**45.3**	**30.8**
First Char Position	23.1	42.3	29.9
Next Proper Noun	22.8	43.2	29.9

Table 1: Feature Exclusion Table: Displays evaluation metrics of Linear Regression Model with exclusion of each feature on the development data set.

Word Token	Entity	BIO tag	POS tag
You	O	O	PRON
should	O	O	AUX
’	O	O	PUNCT
ve	O	O	NOUN
stayed	O	O	VERB
on	O	O	ADP
Redondo	B-location	B	PROPN
Beach	I-location	I	PROPN
Blvd	I-location	I	PROPN

Table 2: Example Input Data format taken from WNUT17 dataset

lion articles and 74 million sentences respectively which allowed BERT to improve the overall GLUE Benchmark score by 7.7%, making it state-of-the-art.

DistilBert, introduced by Sanh et al. (2020) is a lightweight version of BERT that through distillation has 40% less parameters while still retaining 97% of the language understanding capabilities. This enables much faster prototyping when training the data and performing hyper-parameter tuning.

2 Related Work

Named Entity Recognition is a task that has been around for a long time in the field of Natural Language Processing, therefore we will only discuss previous approaches that are closely related to our own work.

Previous work in Named Entity Recognition relied heavily on feature extraction, Tkatchenko and Simanovsky (2012) showed that with a comprehensive set of features it is possible to achieve an F_1-measure of 91.02% on the CoNLL 2003 dataset (Sang and Meulder, 2003) with a Conditional Random Field based supervised NER system. This system relied heavily on disambiguation pages from Wikipedia to infer Named Entities through something comparable to a lookup system which is why we do not use this approach in our paper as it is a time consuming approach to implement. Other work by Klein et al. (2003) used character-level models to advance state-of-the-art in NER through a Hidden Markov Model and a Maximum-entropy Markov Model to improve previous work surrounding word-level models of similar type. The CoNLL dataset has been since dominated by language models and the use of pretrained embeddings since this paper which has less emphasis on the features extracted during training but on the quality of data these models contextualise.

The WNUT17 shared task dataset (Derczynski et al., 2017a) contains 2,295 texts annotated from Reddit, Twitter, Youtube and Stack Exchange which aim to reduce bias in Named Entity Recognition datasets by introducing the task of detecting and classifying novel and emerging named entities in noisy text. Derczynski et al. (2017b) proposes a surface F_1-measure of 40.24% on this dataset which we will use as a benchmark for our own system. Deep learning based systems have been seen to advance state-of-the-art performance in NER with LSTM-CRF models leading the charge in recent years (Lample et al. (2016)) achieving an F_1-measure of 90+% on CoNLL-2003 and Chiu and Nichols (2016) achieving an additional 0.7% through their Bi-Directional LSTM architecture. Post 2017 we begun to see the domination of BERTology (Chiu and Nichols (2016)) in these tasks where the application of these large pretrained language models begin to dominate state-of-the-art performance. Recently BERT$_{base}$ was used by Nguyen et al. (2020), showing an F_1-measure of 56.5% on the WNUT17 dataset thereby improving the previous state-of-the-art by 14+% through pretraining the language model on 850M tweets which will share a similar noise level to as we see in WNUT17 therefore allowing the model to perform better than the originally proposed model which is accessible through HuggingFace Transformers (Wolf et al. (2020)).

Model	Precision	Recall	F1
LR	15.3	35.6	21.4
LR + MIT	23.1	42.3	29.9
DistilBERT	72.2	**59.4**	65.2
DistilBERT + MIT	72.5	59.1	65.1
BERT	75.2	59.1	66.2
BERT + MIT	**75.7**	59.2	**66.4**

Table 3: Comparison between normal classification and additional manual tagging.
LR: Linear Regression, MIT: Manual Inside Tagging

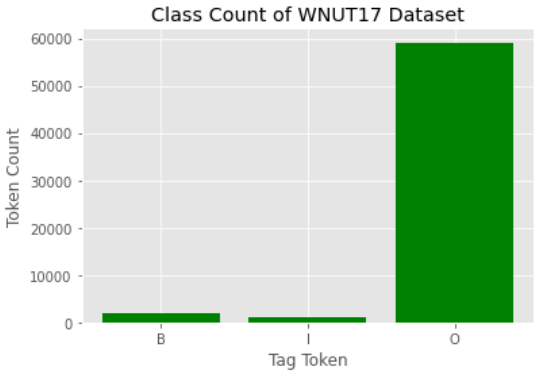

Figure 1: Class count for WNUT17 Dataset

3 Application

3.1 Models

This section will discuss the 3 different models used in this paper.
Throughout this project I made use of 3 models for training and evaluation, that is a Linear Regression Model fit on feature extracted data, BERT and DistilBERT, both language models using the HuggingFace Transformers library. The traditional Linear Regression Model was trained using SKLearn Library (Pedregosa et al., 2011) on a set of extracted features discussed in *Section 3.2*, the model fits the data by Ordinary Least Squares which estimates the coefficients of linear regression $w = (w_1, ..., w_p)$ by minimizing the sum of the squared residuals. solving the problem: $\min_w \|Xw - y\|_2^2$, this means we can fit large amount of data with quite limited resources. The only bounding resource being the memory required to store each word based feature along side its token. For the BERT model I used the standard BERT_{base} cased from HuggingFace, the cased variant of the model is important in NER for identifying entities in the case of abbreviations that may share the same characters with non-entity based words such as *"dog"* and *"DoG" (Difference of Gaussians)*. The model consists of 12-layers, 768-hidden layers, 12-attention heads and 110M parameters, the model was trained with a batch size of 32, 10 epochs, 1000 warm-up steps and early stopping parameters if validation accuracy does not increase over 3 epochs. DistilBERT is the second model that was used, since it allowed for much faster training we could increase the number of epochs from 10 to 20 and our batch size from 32 to 64 and increase the amount of data used to train the model.

3.2 Feature Extraction

This section discusses the features extracted for the Linear Classifier and how each affects performance. Some of the features extracted for the Linear Classifier model are listed below along with a short description of each feature.

1. **POS Tag:** Part-of-speech tag as integer representation for each word.
2. **Proper Noun:** Boolean representation of word's POS tag if is Proper Noun.
3. **Title Case:** Boolean, if first char in word is uppercase.
4. **Start/End Case:** Boolean, if current word is beginning or end of sentence.
5. **Previous POS:** contains previous POS tag integer representation.
6. **Next POS:** contains next POS tag integer representation.
7. **Hashtag:** Boolean, represents if hashtag if first char of given word.
8. **Abbreviation:** Boolean, if all chars in sequence are capitalised.
9. **First Char Position:** Contains integer representation of first char in word's index in sentence.
10. **Next Proper Noun:** Integer of next proper noun in sequence if it exists.

Table 1 shows the effect each feature has on the overall performance and the fragility of feature extraction, we can see the Boolean representation of proper noun's significantly affects the result of the model in terms of precision, recall and f1 measure. Where as the addition of some features can actually inhibit the fitting of the data in the case of feature such as the previous tag and especially the

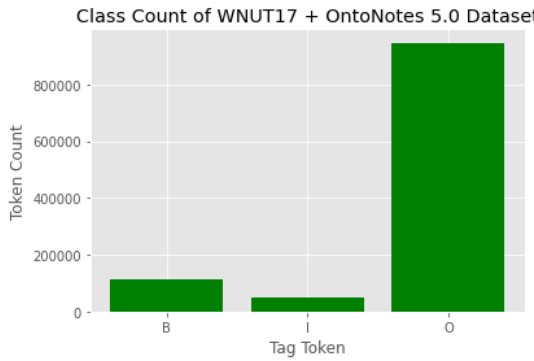

Figure 2: Class count WNUT17 + OntoNotes 5.0

abbreviation feature which when excluded actually increase performance.

3.3 Methods Used

This sub-section covers the methods used in both linear regression and language model models for pre-processing and evaluating data.

3.3.1 Tokens to Sequences

WNUT17 and OntoNotes 5.0 data (Weischedel et al., 2017) is supplied in the format displayed in *Table 2* we therefore require a function to convert these separated words into their sentences. This function handles that by maintaining an index count that iterates once a null value is found (end of sentence token) and then using the *groupby* Dataframe function to group by the index count. Therefore this function is only utilised in both BERT and DistilBERT, since Linear Regression cannot encode for sequences but for single vectors.

3.3.2 Manual Inside Tags - Appendix A

This function manually sets the inside tags *I* of the prediction sequences before evaluation, it checks each sequence where a *B* tag follows another *B* tag and sets it to an *I* instead which in the evaluation dataset tends to be more common then 2 concurrent *B* tags. initial attempts at fitting the linear classifier found that even with oversampling the *I* tags or under-sampling *B* tags would still result in poor generalisation that *I* tags follow *B* tags. Therefore treating the problem as a binary classification significantly improved classification of full entities. *Table 3* shows the difference manually setting the inside tags makes on the precision, recall and F_1-measure. Something of interest from the data is models that generalise to the task poorly see greater increases score but since we see such a low performance increase in DistilBERT and BERT we can conclude that they are able to identify *I* tags in an entity sequence quite well.

Model	Precision	Recall	F1
DistilBERT	66.8	56.8	61.4
DistilBERT + OntoNotes	73.3	60.1	66.0
BERT	70.1	56.4	63.2
BERT + OntoNotes	**74.4**	61.2	**67.2**

Table 4: Comparison between normal classification and using additional data for training.

3.3.3 Handle Token Offsets - Appendix B

One of the major problems with using language models for token classification is that they are already pretrained on a vocabulary and if words in your input data are not found in the vocabulary then tokenising the data will result in splitting up the word into multiple suffixes found in the vocab. An example of this might be *Tokenisation* which when tokenised is split into the following tokens *Token, #isation*. This requires that we also adjust our labels to match the length of our token sequence, the tokeniser outputs offset maps which we can use to append padding sequences to additional tokens. This means that we are not fine-tuning or predicting our model on additional tokens but the alternative of fine tuning on token suffixes results in a overall performance decrease.

3.3.4 Evaluation Metrics - Appendix C

The method used in this function evaluates whole named entities to produce precision, recall and F_1-measures. We only count a true positive tp if the whole entity phrase is predicted correctly. The calculation of false positives fp is denoted as $fp = n_{pred} - tp$ where n_{pred} is the total number of predictions of entity tag B. False negatives fn are calculated as $fn = n_{gold} - tp$ where n_{gold} is the number of entities in the labelled dataset. Using this data we can calculate the metrics used in our evaluation. Precision, Recall and F_1 are all standard metrics in the field of Named Entity Recognition and can allow us to compare our model's performance to other paper's. It also allows for the generation of confusion matrices and other visualisations which are helpful in evaluating our model.

4 Data

4.0.1 WNUT17

The goal of the WNUT17 dataset is to provide high quality data that intends to enable identifying

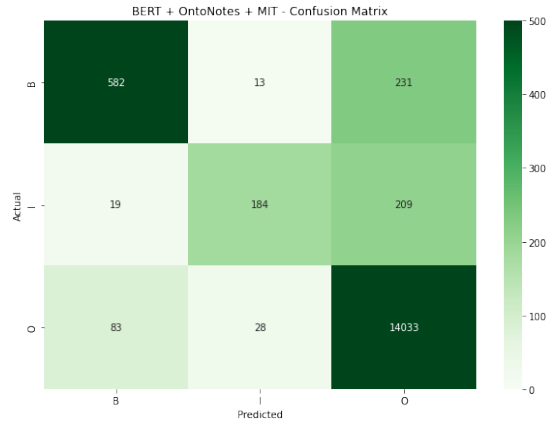

Figure 3: Confusion Matrix: BERT + OntoNotes 5.0 + MIT

unusual, previously-unseen entities unlike other datasets that may all originate from similar sources, these models then perform poorly when faced with data from a different domain. The data is taken from 4 online sources *Twitter, Reddit, Youtube and Stack Exchange, then manually tagged, Figure 1* shows the count of Beginning, Inside and Ending (BIO) tags found in the training dataset. This shows it only contains a very small number of named-entities which is a realistic representation of the frequency of these entities in online text. The data is split into 3 sets: *Training, Development and Testing* which contain 3394, 1009 and 1287 sentences respectively for our evaluation.

4.0.2 OntoNotes 5.0

OntoNotes is a multi-lingual dataset that uses 18 entity types, we do not require the entity classes for our task therefore we take the first character of each class to extract its BIO tag. *Figure 2* shows the combination of both datasets and how the OntoNotes dataset contains many more entities per sequence thereby significantly reducing the entity disparity found in *Figure 1.*

5 Results and Discussion

5.1 Results

We can observe in *Table 5*, the best BERT model performs 4.7% better than the best DistilBERT model and 231.8% better than a Linear Regression model. This is to be expected as BERT is a large model that has already encoded a large corpora from Wikipedia, this means that the model has the ability to encode for often unseen words throughout its 4.4 million articles. DistilBert trades

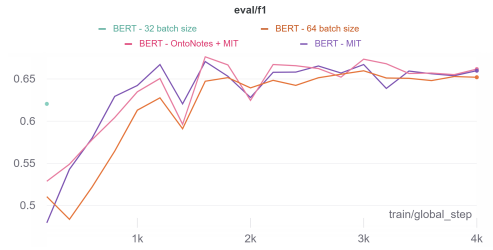

Figure 4: F1 Score plotted against training steps for BERT models with different parameters

off performance for training time, it did perform slightly worse in precision, recall and f1 score although it did train 61.4% faster. The confusion matrix displayed on *Figure 3* shows that the BERT model correctly identifies beginning and outside tags with higher accuracy than inside tags, while still over predicting the outside class which in future work could be mitigated by oversampling full entities to ensure beginning and outside predictions aren't skewed. In our discussion look at the results of our analysis into the specific entities our model failed to predict and entities we predicted that were not labelled as such.
Figure 4 shows how parameter tuning on BERT did not significantly affect performance on our evaluation dataset, which is why it does not feature much discussion in this paper as long training times couple with hyperparameter tweaking is an intensive task that hasn't affected results in a meaningful way.

5.2 Discussion

When evaluating our results and performing analysis of the data we analysed some of the tags BERT classified as Entities *B or I* that were manually tagged as not entities *O* by WNUT17 dev test set are as follows:

Kremlin
North
Zora
screwattack
Fallon
Silver Falls
soviet
Peanut Butter Jelly

Analysing these *"non-entities"* it could be objectively stated that some of these are tags that have been labelled incorrectly, human error is bound to occur in manually tagged datasets,

Model Architecture	ML Library	Training Time (s)	Precision	Recall	F_1 Score
Logistic Regression	Scikit-Learn	**10.4**	15.3	35.6	21.4
Logistic Regression + MIT	Scikit-Learn	10.5	23.1	42.3	29.9
DistilBERT	HuggingFace	385	72.2	59.4	65.2
DistilBERT + MIT	HuggingFace	**381**	72.5	59.1	65.1
DistilBERT + MIT + OntoNotes	HuggingFace	522	73.3	60.1	66.0
BERT	HuggingFace	**615**	70.1	56.4	63.2
BERT + MIT	HuggingFace	628	74.4	61.2	67.2
BERT + MIT + OntoNotes	HuggingFace	761	**76.5**	**63.3**	**69.3**

Table 5: Precision, Recall and F1 score on Logistic Regression, DistilBert and BERT, with Manual Inside Tagging and additional data from OntoNotes 5.0.

such issues such as ”Fallon” which given the full context of its text ”This is why Fallon is better. Not just shitty Trump jokes. Actual content.” refers to American Comedian ”Jimmy Fallon”, which in fact BERT classified correctly.
As a result of this subjectivity it classified 83 entities that were not regarded as entities. For this task we subjectively went through these entities and identified 36 that were not labelled correctly in the original dataset. Although in the reverse we could not identify many words that had been labelled as entities that were in fact not such.

emma
ryan
trump
martin
skittles
rick
trump

What we found as a result of this data is that BERT struggles with non-titled named entities, therefore a way we might approach this in a future case is either by oversampling our entities and removing their title capitalisation or by using an uncased model in that title case will not be a feature of the model at all.
Simple tests on utilising an uncased model showed f1 reduction of 2+% on BERT therefore it will need more work to become a viable solution to this problem.

6 Conclusion

In this work, we evaluated both deep and non-deep methodology on the WNUT17 shared task of Named Entity Recognition. We found that manual tagging still aids identifying inside tags of entities even when using advanced language models showing there is still room for improvement in classifying full entities. We discussed the problems with subjectivity of manual tagging data for datasets, how there is still room for improvement in building high quality objective data for classification. We observed how although large amounts of additional data from external sources do give improvements to our models performance it is not proportional to the amount of data added.
We observed through our findings, it still remains that NER is a knowledge demanding task and the greater sources of knowledge and how we encode knowledge in our model vastly improves the performance of entity recognition in comparison to standard feature extraction approaches.
More effective construction and application of identifying non-title cased entities through tweaking the sampling and how we fine tune the training data for our models. We would also like to see the extension of our model onto the full task of Named Entity Recognition.

References

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017a. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

Leon Derczynski, Eric Nichols, Marieke Erp, and Nut Limsopatham. 2017b. Results of the wnut2017 shared task on novel and emerging entity recognition. pages 140–147.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and

Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher Manning. 2003. Named entity recognition with character-level models.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Maksim Tkatchenko and Andrey Simanovsky. 2012. Named entity recognition: Exploring features. In *KONVENS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Ralph M. Weischedel, Eduard H. Hovy, Mitchell P. Marcus, and Martha Palmer. 2017. Ontonotes : A large training corpus for enhanced processing.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

7 Appendices

Appendix A:

```
def manual_inside_tags(predictions):
  output_arr = []
  for i in range(len(predictions)):
    for j in range(1, len(predictions[i])):
      if(predictions[i][j-1] == 'B' and predictions[i][j] == 'B'):
        predictions[i][j] = 'I'
  return predictions
```

Appendix B:

```
def encode_tags(tags, encodings):
    labels = [[int(tag) for tag in doc] for doc in tags]
    encoded_labels = []
    for lab, off in zip(labels, encodings.offset_mapping):
        encoded_lab_positions = np.ones(len(off), dtype=int) * -1
        arr_offset = np.array(off)

        try:
          get_first_pos = arr_offset[:,0] == 0
          get_second_pos = arr_offset[:,1] != 0
          adjusted_label_positions = get_first_pos & get_second_pos
          encoded_lab_positions[adjusted_label_positions] = lab
          encoded_labels.append(encoded_lab_positions.tolist())
        except:
          for i in range(len(doc_enc_labels)):
            if(first_pos[i] == True and second_pos[i] == True):
              encoded_lab_positions[i] = lab.pop(0)
          encoded_labels.append(encoded_lab_positions.tolist())
    return encoded_labels
```

Appendix C:

```
def f1_metric(txt):
  npred = 0; ngold = 0; tp = 0
  nrows = len(txt)
  for i in txt.index:
    if txt['prediction'][i]=='B' and txt['bio_only'][i]=='B':
      npred += 1
      ngold += 1
      for predfindbo in range((i+1),nrows):
        if txt['prediction'][predfindbo]=='O' or txt['prediction'][predfindbo]=='B':
          break
      for goldfindbo in range((i+1),nrows):
        if txt['bio_only'][goldfindbo]=='O' or txt['bio_only'][goldfindbo]=='B':
          break
      if predfindbo==goldfindbo:
        tp += 1
    elif txt['prediction'][i]=='B'and txt['bio_only'][i] !='[PAD]':
      npred += 1
    elif txt['bio_only'][i]=='B':
      ngold += 1

  fp = npred - tp
  fn = ngold - tp
  if(tp+fp == 0 or tp+fn == 0):
    return 0.0
  prec = tp / (tp+fp)
  rec = tp / (tp+fn)
  f1 = (2*(prec*rec)) / (prec+rec)
  return f1, prec, rec
```