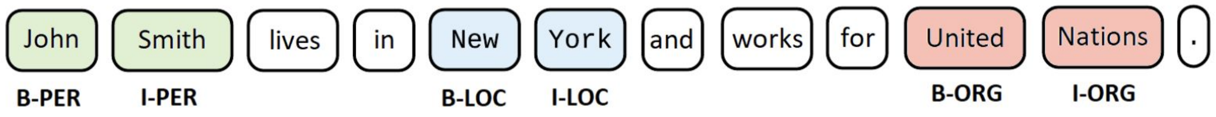
NAMED ENTITY RECOGNITION

Varsha Rajasekar, Parth Shah

Introduction

Information Extraction is the task of automatically extracting structured information from unstructured or semi-structured data. Information Extraction has many applications, including business intelligence, resume harvesting, media analysis, sentiment detection, patent search, and email scanning. Named Entity Recognition (NER) is a subtask of information extraction that seeks to locate and classify entities from text such as person names, organization, locations, monetary values, medical codes etc.

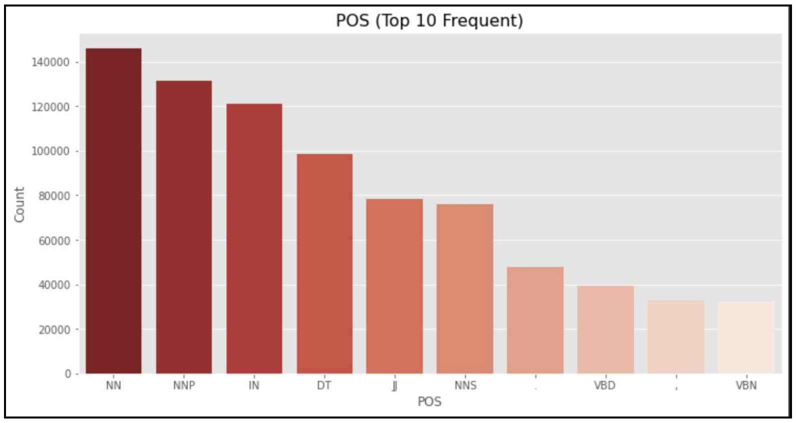


In this project we aim to classify entities from dataset leveraging statistical models, pre-trained models as well as training models from scratch using embeddings. While pre-trained models are robust and have the advantage of being ready to train and test faster, we also aim to train models to extract custom entities.

For our multinomial classification, we used linear SVM optimized with SGD as our baseline model, a bidirectional recurrent neural network and pre trained NER models spaCy and flair, out of which Flair gave the best results.

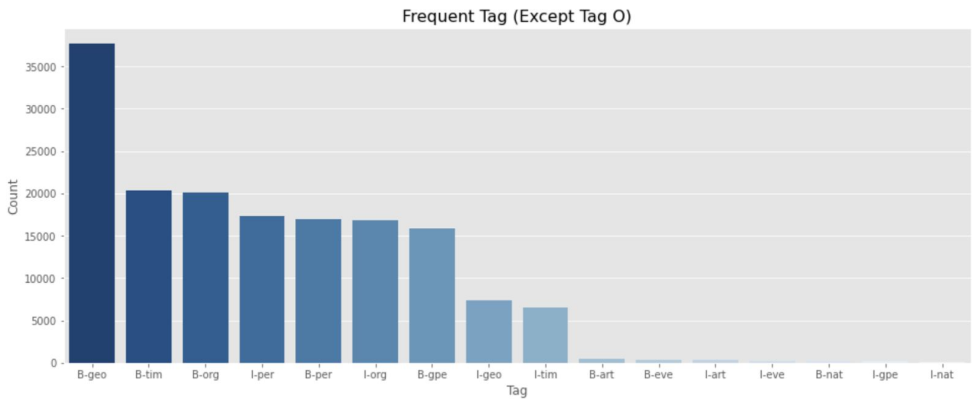
Data Exploration

The dataset is splitted into word tokens and consists of 4 columns: *Sentence #, Word, POS, Tags.* It has ~48k sentences and word count is 1354149. The dataset has 42 different part-of-speech (POS) tags and 17 named-entity tags (Tag). The top 10 frequent POS Tags are shown in the following plot with Noun POS dominating the dataset.



Named-entity tags are in IOB (*inside-outside-beginning)* format. The B- prefix before a tag indicates that the tag is the beginning of a chunk, and an I- prefix before a tag indicates that the tag is inside a chunk. The B- tag is used only when a tag is followed by a tag of the same type without O tokens between them. An O tag indicates that a token belongs to no chunk.

The frequency plot for named-entity Tag is shown below (without Tag 'O' for better plot). The most frequent is *geolocation* followed by *time, organization and person.* The least frequent are *art, event or natural-phenomenon.*



Pre-processing

Only the first instance of the beginning of each token for a sentence were labeled in the column *Sentence #.* So a forward fill was done to have each token correspond to the same *Sentence #.* Additionally, a separate class *SentenceTagger* was built to fetch all tokens for a given sentence number for analysis purposes.

The dataset was splitted into *train* and *test* sets before further processing. The first 40,000 sentences are kept into the *training set* and the remaining in the *test set.* The final *training set* has ~875k instances and the *test set* has ~173k instances.

Evaluation

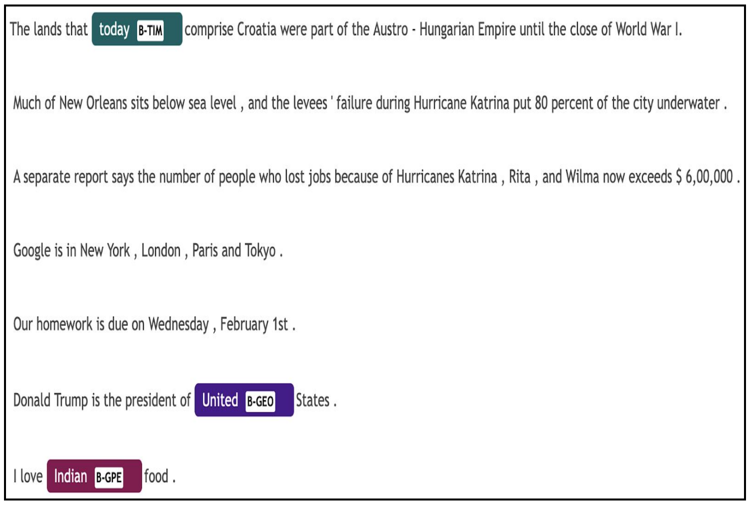
Since *Tags* are imbalanced we use *F1* (micro and macro) to evaluate all models. For a better evaluation score we remove tag *O* during calculation of the metrics as well as the *beginning (B-)* and *inside (I-)* are counted as a single entity. For example, during evaluation *B-gpe and I-gpe → gpe*.

Along with checking the model's overall performance we also check for individual entity's performance with their *precision, recall* and F1-score to improve our models and analysis.

Stochastic Gradient Descent Classifier

For our baseline model, we have used the SVM with SGD training. The function used to achieve this is the SGD-Classifier which is a linear classifier optimized by SGD. SGD allows minibatch learning which is why it is efficient for large scale problems. We can control the model that an SGDClassifier() fits through its loss hyperparameter. For our project we have used the default loss parameter 'hinge' which fits a linear support vector machine.

We first converted our features in the train set to one hot coded vector using DictVectorizer and then split to train and validation sets. After fitting the model on our train set we evaluated its performance on the test set and got a micro F1 score of 0.69 and macro F1 score of 0.49. We can see from the annotated text in figure below, the model does not predict most of the entities.



The lands that today B-TIM comprise Croatia were part of the Austro - Hungarian Empire until the close of World War I.

Much of New Orleans sits below sea level , and the levees ' failure during Hurricane Katrina put 80 percent of the city underwater .

A separate report says the number of people who lost jobs because of Hurricanes Katrina , Rita , and Wilma now exceeds $ 6,00,000 .

Google is in New York , London , Paris and Tokyo .

Our homework is due on Wednesday , February 1st .

Donald Trump is the president of United B-GEO States .

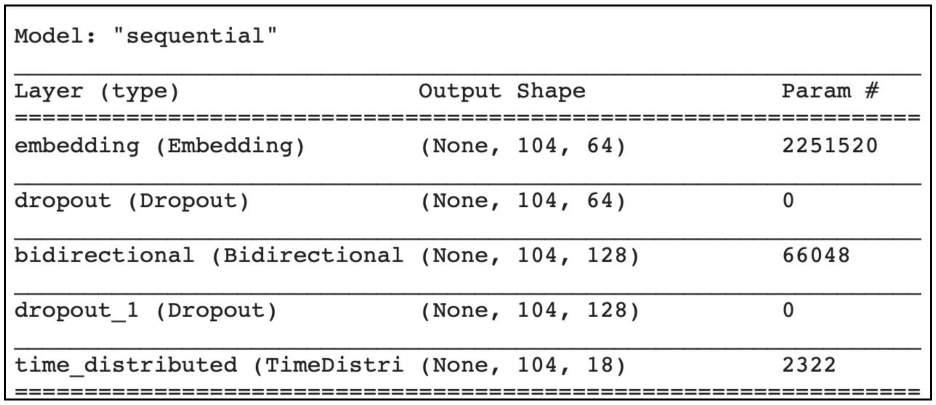
I love Indian B-GPE food .

	Precision	Recall	F1-Score
art	0.50	0.02	0.03
eve	0.81	0.28	0.41
geo	0.64	0.92	0.75
gpe	0.96	0.75	0.82
nat	0.17	0.35	0.23
org	0.90	0.53	0.67
per	0.93	0.67	0.78
tim	1.00	0.55	0.71

Bidirectional LSTM

The SentenceTagger class groups our data by the sentence number and returns the words and their corresponding tags for each sentence. Then we assign an index to every word in our dataset's vocabulary. The first index of 0 is assigned to a generic word 'PAD' that we use for padding later. Each sentence is converted from a list of words to a sequence of indices where each index corresponds to a word.

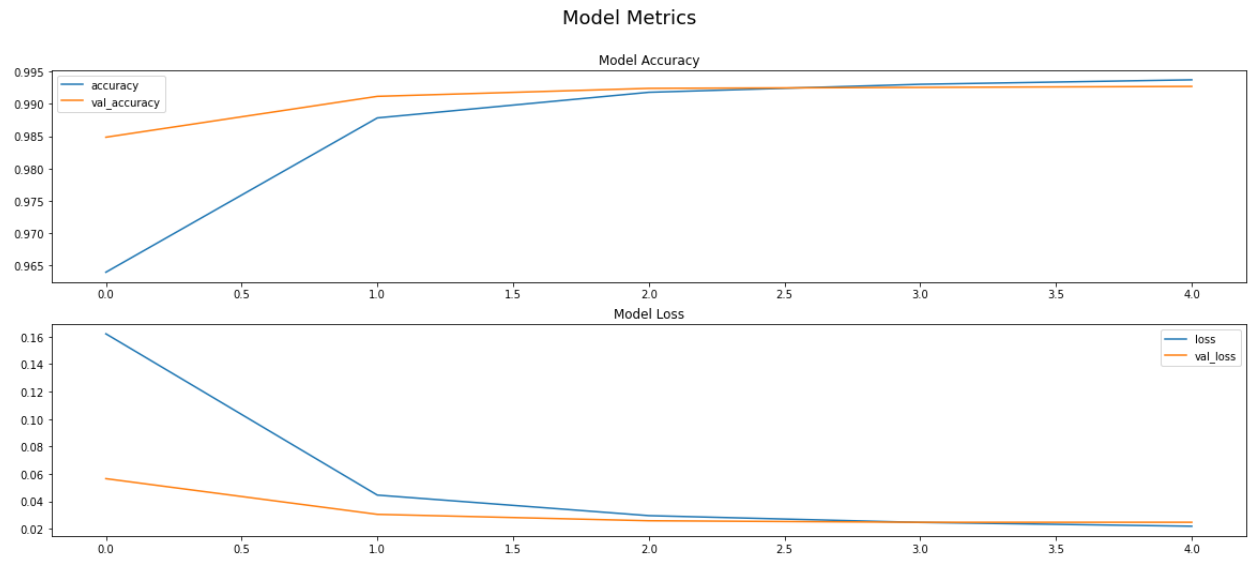
Similarly, we assign an index to every tag where the index 0 is reserved for 'PAD', convert the corresponding list of tags for each sentence into a sequence of indices and then apply one hot encoding on these sequences. In order to feed our sentences into the model, they all need to be of the same length. We took the length of the longest sequence of words and padded all the shorter sequences with the word 'PAD' to achieve this length which for our project is 104.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 104, 64)           2251520
_________________________________________________________________
dropout (Dropout)            (None, 104, 64)           0
_________________________________________________________________
bidirectional (Bidirectional (None, 104, 128)          66048
_________________________________________________________________
dropout_1 (Dropout)          (None, 104, 128)          0
_________________________________________________________________
time_distributed (TimeDistri (None, 104, 18)           2322
=================================================================
```

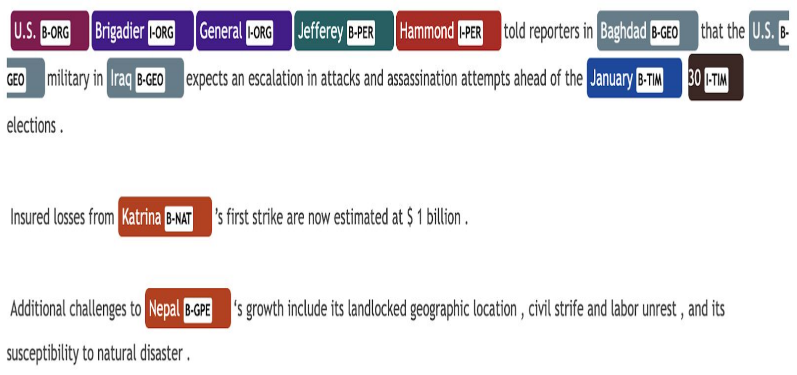
We didn't use pre-trained embeddings for our project. Instead the embeddings are created by the embedding Layer which takes in the padded sequences of length 104 and transforms each token into a vector of n-dimensions. For our project we have taken 64 as our embedding vector dimension. We have used two dropout layers with dropout = 0.5.

The Bi-LSTM layer takes the output of the previous layer and uses 64 units. Since we use the default merge mode 'concat', the forward and backward outputs from each of the 64 units are concatenated which doubles the outputs to the next layer which is 64*2=128. The output layer maintains the many to many RNN architecture and makes sure we get output from every input sequence. The model uses the softmax activation function.



The model is trained by cross-entropy loss coming from each position over 5 epochs. To avoid overfitting we want the validation loss to be slightly higher than training loss which, from figure-- we can evidently see, has been achieved. After evaluation the model's micro F1 score was 0.85 and macro F1 score was 0.54.

Our initial model didn't have any dropout layers and when we ran the model over 10 epochs, our training loss << validation loss. Adding the dropout layers and reducing the number of epochs to 5, improved the overall model accuracy and loss.


U.S. B-ORG Brigadier I-ORG General I-ORG Jefferey B-PER Hammond I-PER told reporters in Baghdad B-GEO that the U.S. B-GEO military in Iraq B-GEO expects an escalation in attacks and assassination attempts ahead of the January B-TIM 30 I-TIM elections .

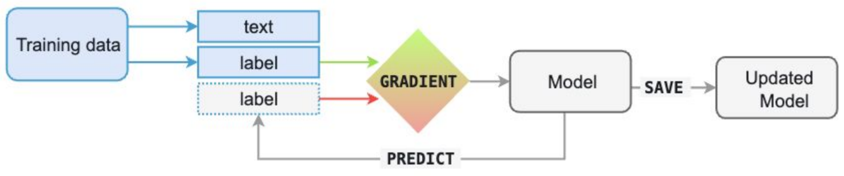
Insured losses from Katrina B-NAT 's first strike are now estimated at $ 1 billion .

Additional challenges to Nepal B-GPE 's growth include its landlocked geographic location , civil strife and labor unrest , and its susceptibility to natural disaster .

	Precision	Recall	F1-Score
art	0.00	0.00	0.00
eve	0.00	0.00	0.00
geo	0.86	0.86	0.86
gpe	0.95	0.91	0.93
nat	0.00	0.00	0.00
org	0.80	0.71	0.76
per	0.90	0.81	0.85
tim	0.93	0.88	0.90

spaCy

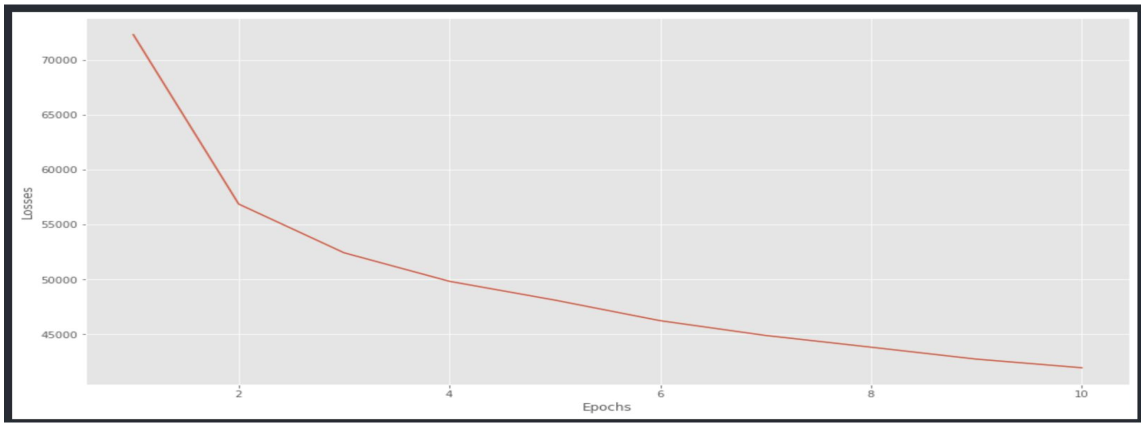
spaCy is an open-source library for NLP written in Python and Cython. It provides a tokenizer, a POS-tagger and named-entity recognizer.



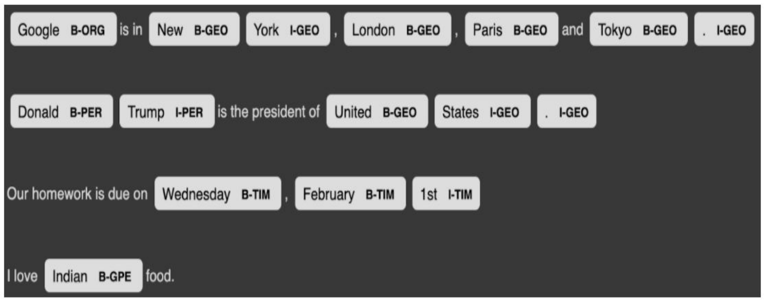
We converted each of our *Sentence* tokens (text) and their corresponding *Tag* (label) into a JSON format required by spaCy. Each *Tag* is stored in the entity's key with their corresponding character indices in the sentence.

spaCy uses Bloom embeddings where every word is mapped to a hash key rather than the word itself which leads to more compact embeddings. Our architecture is 4 Convolutional Neural Nets followed by a feed forward neural network. Each CNN has a max-out activation function with a dropout of 0.35.

Since our dataset had custom entities which were not provided by *out-of-the-box* spaCy NER models, we trained our own custom model. We train our model for 10 epochs with a batch-size of 32. spaCy defines its own loss for training. We observed the training loss decreasing at every epoch on the training set.



spaCy achieved an overall F1 (micro) score of 0.86 and F1 (macro) score of 0.598. In terms of individual entity performance, spaCy wasn't able to identify any *art* entities. It also underperformed in identifying *eve* and *nat* achieving F1-score of 0.39 and 0.51 respectively. spaCy gave a significantly higher score on rest of the entities with the highest score for *gpe* and *time* tags respectively.



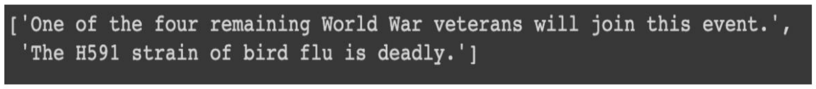
Google B-ORG is in New B-GEO York I-GEO , London B-GEO , Paris B-GEO and Tokyo B-GEO . I-GEO

Donald B-PER Trump I-PER is the president of United B-GEO States I-GEO . I-GEO

Our homework is due on Wednesday B-TIM , February B-TIM 1st I-TIM

I love Indian B-GPE food.

Correctly identified entities.



```
['One of the four remaining World War veterans will join this event.',
 'The H591 strain of bird flu is deadly.']
```

No entities found by the model

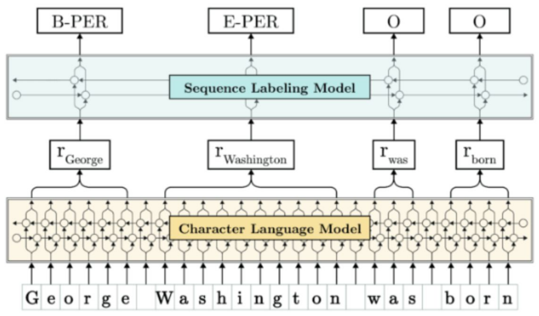
	Precision	Recall	F1-Score
art	0.00	0.00	0.00
eve	0.68	0.28	0.39
geo	0.90	0.88	0.89
gpe	0.97	0.95	0.96
nat	0.81	0.37	0.51
org	0.87	0.77	0.82
per	0.88	0.91	0.89
tim	0.99	0.87	0.93

Flair

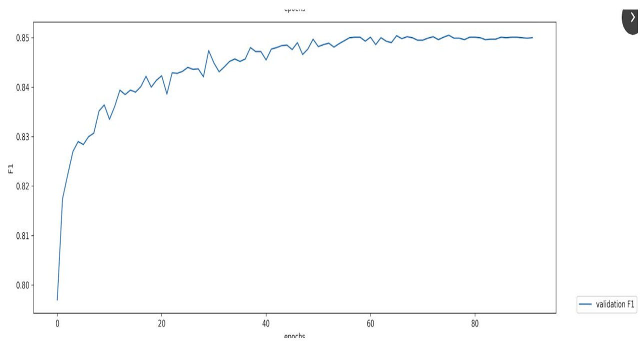
Flair is an open-source library built on top of PyTorch. It provides pre-trained models for classification, named entity recognition and language modeling.

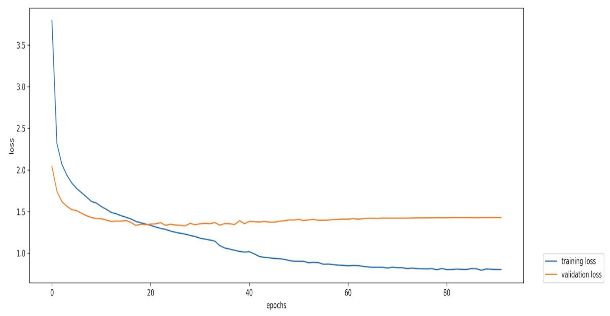
We use Flair embeddings *news-forward* and *news-backward* in this project which are trained on 1 billion word corpus. These are 300 dimensional embeddings which are stacked together using the Flair's StackedEmbedding class.

Our model architecture is a character language model with bi-directional LSTM. The contextual token embeddings outputted by the language model are passed to a bi-directional LSTM to output the entity tags. The LSTM layer has 256 hidden units and a dropout of 0.05. The final layer in the network is a feed forward network that outputs the labels (entities).

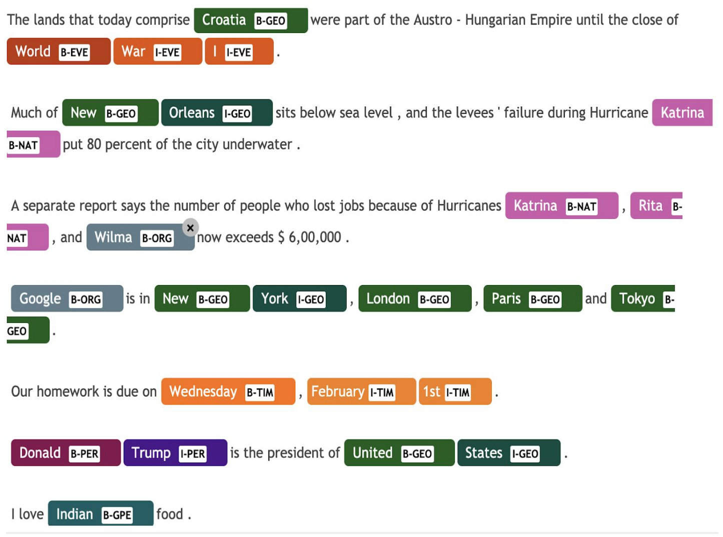


The training data is splitted into a *train* and *dev* set with a 80/20 split and converted into IOB CoNLL format with token and entity (labels). They are then imported as a Sequence Labeling dataset using Flair's Corpus class with train and dev set having 31999 and 8000 sentences respectively. We use an adaptive learning rate starting from 0.1. The maximum number of epochs are set to 150 and batch size is 32. The model trains for 93 epochs until the loss stops improving giving a validation F1 micro and macro score of 0.85 and 0.69 respectively.





Flair achieved an overall F1 (micro) score of 0.86 and F1 (macro) score of 0.68. Flair gives comparatively a better score than previous models in identifying *less-performing* entities *art, eve* and *nat.* While its performance for the other entities is significantly better, we noticed for certain entities its score was slightly lower in comparison with previous models. Similar to previous models Flair gives the highest score for *gpe* and the lowest on *art* entity.


The lands that today comprise Croatia B-GEO were part of the Austro - Hungarian Empire until the close of World B-EVE War I-EVE I I-EVE .

Much of New B-GEO Orleans I-GEO sits below sea level , and the levees ' failure during Hurricane Katrina B-NAT put 80 percent of the city underwater .

A separate report says the number of people who lost jobs because of Hurricanes Katrina B-NAT , Rita B-NAT , and Wilma B-ORG now exceeds $ 6,00,000 .

Google B-ORG is in New B-GEO York I-GEO , London B-GEO , Paris B-GEO and Tokyo B-GEO .

Our homework is due on Wednesday B-TIM , February I-TIM 1st I-TIM .

Donald B-PER Trump I-PER is the president of United B-GEO States I-GEO .

I love Indian B-GPE food .

	Precision	Recall	F1-Score
art	0.27	0.16	0.20
eve	0.45	0.44	0.44
geo	0.86	0.91	0.89
gpe	0.96	0.96	0.96
nat	0.56	0.53	0.54
org	0.77	0.72	0.74
per	0.80	0.80	0.80
tim	0.90	0.87	0.89

Conclusion

The use of embeddings (bi-LSTM self embeddings, bloom and flair) significantly outperforms one-hot vectors (*DictVectorizer to SGD*) in both micro and macro f1-scores.

Flair achieved the highest macro and micro F1-score of 0.68 and 0.86 respectively. The flair model outperformed in identifying entities that were less in frequency such as *art, eve* and *nat* while giving similar scores for other entities. Flair already achieves state-of-the-art NER score on CoNLL data and using the same architecture and embeddings we were able to achieve a higher performance model.

	Micro F1	Macro F1
SVM(SGD)	0.69	0.49
Bi-LSTM	0.85	0.54
spaCy	0.86	0.56
Flair	0.86	0.68

Varsha	SGD Classifier, Bidirectional LSTM
Parth	spaCy, Flair

Research Paper Summary:

FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP

This paper gives an overview of the framework and functionality of Flair. The principal design goal is to allow researchers to build a single model architecture that can then make use of any type of word embedding with no additional engineering effort. FLAIR represents NLP concepts such as tokens, sentences and corpora with simple base classes. It supports a number of embeddings like Glove, ELMo, BERT, Flair embeddings etc. This architecture also lets us mix and match different types of embeddings by passing a list of embeddings to stack into the StackedEmbeddings class. We can produce vector representations of entire documents using two main embedding classes, namely DocumentPoolEmbeddings, which applies a pooling operation to all word embeddings and DocumentLSTMEmbeddings, which applies an LSTM over the word embeddings.The data fetcher in Flair conveniently downloads the dataset if it is not present on the local disk. The dataset is then read into an object of type TaggedCorpus which defines training, testing and development splits.

For model training, the ModelTrainer class implements a host of mechanisms including features such as mini-batching, model checkpointing, evaluation methods etc. The HYPEROPT library supported by Flair, implements a Tree of Parzen Estimators (TPE) approach to hyperparameter optimization. FLAIR includes a model zoo of pre-trained sequence labeling, text classification and language models, allowing users to apply pre-trained models to their own text. It also involves different model variants such as the *default* variant, which typically use embeddings from language models with 2048 hidden states, the *fast* variant use embeddings typically from LMs with 1024 hidden states and the *multilingual* variant that can predict tags for text in multiple languages. Current research focuses on supporting more embeddings such as transformer embeddings and LASER embeddings as well as developing new embedding types and extending the framework to facilitate multi-task learning approaches.