CS5740: Project X
https://github.com/cornell-cs5740-20sp/final-Danielczl315

Zhenglun Chen
zc447

1 Introduction (6pt)

In this project, the task is to build a named-entity recognition (NER) model for Twitter text. The data I used for this project is the provided tweet data-set, with each word labeled with either B, I, O. In this setting, B is for beginning of the named entity, I is for continuation of a named entity, and O is for a token that is not part of a named entity.

After reviewing online literature and course material, I decide to use bidirectional long short-term memory (BiLSTM) with conditional random field (CRF) to solve this problem. Also, since tweets are mainly noisy data, I ended up choosing process sentence with ELMo embedding before feeding them into BiLSTM.

In an effort to get the best result, I did several experimentes, including basic BiLSTM vs BiLSTM with conditional random field (CRF), different text data cleaning approach and hyperparameter tuning. In the end, ELMo-BiLSTM-CRF provides the best result and yield f1-score of 0.57461 on test data.

2 Task (5pt)

The formal definition of our task is defined as following statements. V be an universal dictionary and $S = \{B, I, O\}$ be a set of annotations. A word is defined as $w \in V$. From a word, we can define a sentence of length n to be a sequence of words $(w_1, w_2, ..., w_n)$. Given a sentence, our goal is to generate a sequence of annotations $(s_1, s_2, ..., s_n), s_i \in S, \forall 1 \leq i \leq n$ corresponding to each word.

3 Data (4pt)

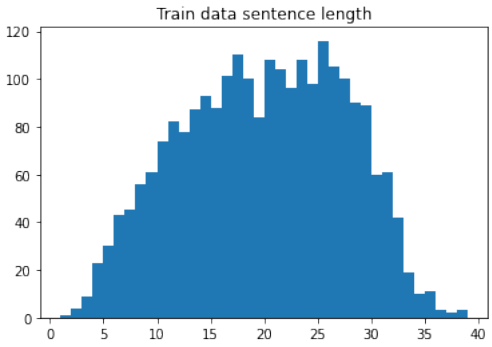
The following are some raw overviews of the data we are working on.

Data	Sentences	Vocab Size
train	2394	10586
dev	959	5168
test	2377	10548

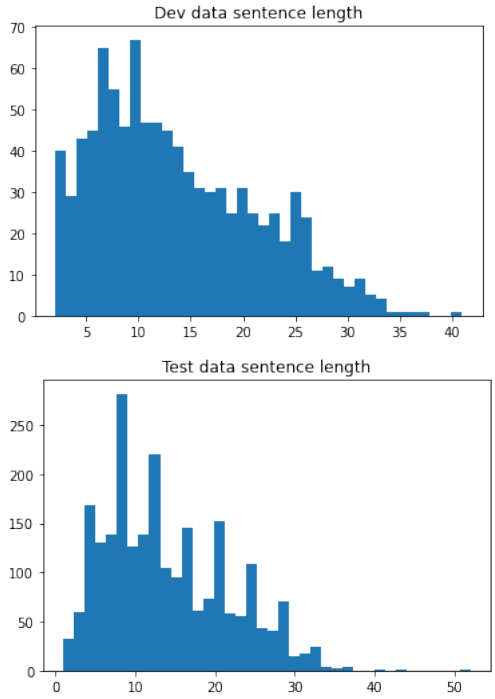
Table 1: Data Overview

For this project, I am using ELMo representation for each word, which is word is represented as a 1024 dimension vector and each sentence of length N is represented as a matrix of size $N \times 1024$. In the ELMo model from AllenNLP[1], sentences are padded in order to match with the sentence with max length in the corpus. This gives me a $N_{maxsentencelength} \times 1024$ matrix for each sentence. Therefore, I did some investigations on the sentences on each data-set.

The max sentence length for train is 39, for dev is 41 and for test is 52. However, to take a better look at the overall data, I visualized the distributions of sentence length in each data-set.



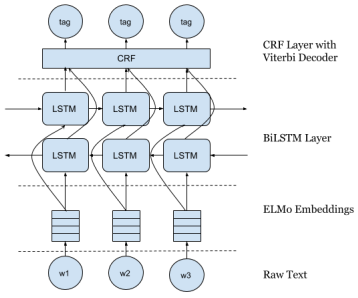
[1] https://allennlp.org/elmo



Intuitively, I can pad every sentence into 52 words into to achieve best performance on test set. However, as we can tell from the graph, most of the sentences are below 40 words. Obviously, there is a trade-off between the padding length, performance and run-time: lower length gives us less run-time but worse performance while higher length gives us more run-time but better performance. Experiment on this will be discussed in the later section.

4 Model (25pt)

Overall, the architecture of my model is ELMo-BiLSTM-CRF. The use of BiLSTM-CRF is inspired by Limsopatham and Collier (2016). [2]



[2]Bidirectional LSTM for Named Entity Recognition in Twitter Messages `https://noisy-text.github.io/2016/pdf/WNUT20.pdf`

4.1 ELMo [3]

ELMo, unlike classic word embeddings like GloVe and Word2Vec which assign each word in the trained vocabulary a fixed word vector, is a deep contextualized word representation based on word use and linguistic context. With the use of character-based encoding in ELMo implementation, it is needless to handle the unknown words and feature extraction.

4.2 BiLSTM

Bidirectional LSTM allows the network to retain both backward and forward information about the input sentence. LSTMs are common units in NLP applications.

4.3 CRF

CRF is a discriminative sequence model that performs well on tasks like NER. In my architecture, CRF takes the output emissions from LSTM $\psi_{EMIT}(x, s)$ where $x \in BiLSTM(ELMo(w))$ and $w \in V^n$ and $s \in S^n$, infers the conditional probability $P(s|x)$ and the path with highest score.

4.3.1 Log-Linear Model

The conditional probability are inferred by a softmax function where Φ denotes the feature vector

$$P(s|x) = \frac{exp(\Phi(x, s))}{\sum_{s' \in S^n} exp(\Phi(x, s'))}$$

4.3.2 Feature Vector

One key component in the definition is the feature vector:

$$\Phi(x, s) \in \mathbb{R}^d$$

which maps any pair of input feature sequence and a state sequence to a d-dimensional feature vector. In this specific task,

$$\Phi(x, s) \in \mathbb{R}^d = \sum^{i} log\psi(x_i, s_i)$$

$$= \sum^{i} log\psi_{EMIT}(x_i|s_i) + log\psi_{TRANS}(s_i|s_{i-1})$$

Emission scores are output from BiLSTM and transition scores are randomly initialized. CRF is then decoded with Viterbi algorithm to produce the produce the most likely sequence s.

[3]Deep contextualized word representations `https://arxiv.org/pdf/1802.05365.pdf`

5 Learning (10pt)

The learning objective for this architecture is similar to common CRF, which is to minimize the negative log-likelihood

$$-log(Z(x, s_{pred}) - GOLDEN(x, s_{true})))$$

where $Z(x, s_{pred})$ is the partition function computed by forward algorithm and $GOLDEN$ refers to the golden score of the sequence.

Algorithm 1: Forward Algorithm

Data: $\psi_{EMIT}(x_i, s_i)$ emissions output by BiLSTM
Result: α_n
for $t \leftarrow 1$ **to** n **do**
  $\alpha_t = \psi_{EMIT}(x_t, s_t) +$
  $\sum \psi_{TRANS}(s_t|s_{t-1}) + \alpha_{t-1}$

6 Implementation Details (3pt)

I used an ADAM optimizer with 0.001 learning rate. With in my architecture, I added dropout layers between ELMO and BiLSTM, BiLSTM and CRF to avoid possible overfitting on the training data. Also, I padded / truncated every sentence to 40 length.

7 Experimental Setup (4pt)

Nothing fancy here, basically I just did two experiement, for which I compared ELMo-BiLSTM against ELMo-BiLSTM-CRF. Then I compared with 40 against 52 sentence length when generating embeddings.

8 Results

Test Results (3pt) The result I got from leaderboard is 0.57461, with ELMo-BiLSTM-CRF architecture and 40 sentence length.

Development Results (7pt) Here are the results I got from two experiments

	F1 Score
With CRF	0.513
Without CRF	0.386

Table 2: Exp. 1

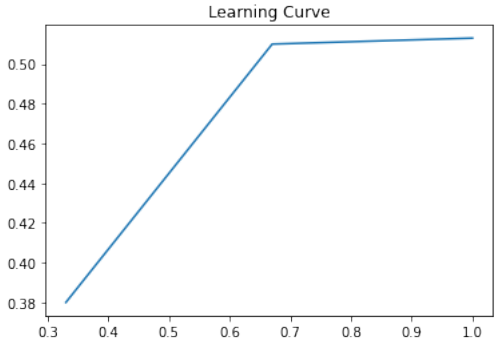
As shown in the table, the CRF structure gives the performance quite a boost.

Sentence Length	F1 Score
40	0.513
52	0.515

Table 3: Exp. 2

There is really no significant difference between two choices on the development set since there is only one sentence that has length above 41. Since I am randomly initializing the transition weights, the fluctuation might be result from there rather than difference in sentence length.

Learning Curves (4pt) For the learning curve, I used 33%, 67% and all of the training data. From the curve I find that there is no significant increase in performance after I switched from 67% of the data to full training data.



Speed Analysis (4pt) When training on full data one epoch takes about 5 minute and there are 2394 tokens (sentences), so 0.13 s/token.

9 Analysis

Error Analysis (7pt) One source of failure is the combination of the non-entity words becomes an entity. For example, "cedar point" in the first example is an entity but they are falsely classified as non-entity.

10 Conclusion (3pt)

One major finding is that contextualized embeddings like ELMo does perform well in working with noisy and colloquial text data. One thing I can improve from now is working on hyperparameter tuning and weights initialization to achieve better results.