Few-shot Learning for Named Entity Recognition in Medical Text - Tech Review

Saikrishna Sanniboina
University of Illinois Urbana- Champaign
Champaign, Illinois, USA
ss235@illinois.edu

ABSTRACT

The research paper, titled "Few-shot Learning for Named Entity Recognition in Medical Text" introduces an innovative approach to enhance Named Entity Recognition (NER) in medical text by leveraging modern Natural Language Processing (NLP) techniques. This review critically assesses the methodologies employed in constructing the input data, with a particular focus on the significance of incorporating character-level features, word-level features, casing features, and the architectural design.

The evaluation not only scrutinizes the methodologies but also validates their accuracy. Additionally, I have implemented personalized modifications, including the incorporation of different word embeddings and adjustments to hyperparameters. A clear representation of input data is deemed imperative for NER tasks, as it facilitates the learning of semantic nuances necessary for accurate NER identification.

Furthermore, this review assesses the alignment of the paper with existing research in NER, critically evaluating how various parameters and word embeddings contribute to its improved performance. The outcomes of these enhancements hold promise for applications in chatbots and emergency tasks within the medical field by precisely identifying named entities.

KEYWORDS

NER, BILSTM, Glove Embeddings,Word2vec Embeddings, Nadam Optimizer, Medical Text,Named Entities

1 INTRODUCTION

The paper titled "Few-shot Learning for Named Entity Recognition in Medical Text" introduces an innovative approach aimed at enhancing Named Entity Recognition (NER) through modern techniques. The exploration of this groundbreaking methodology delves into the complexities of natural language processing (NLP) tasks, highlighting the superiority of machine learning (ML) methodologies over traditional rule-based approaches. Notably, the success of neural networks, particularly architectures such as long short-term memory (LSTM) recurrent neural networks (RNNs) combined with convolutional neural networks (CNNs), is exemplified in achieving state-of-the-art performance in NLP tasks, surpassing conventional methods and setting new benchmarks in datasets like CoNLL-2003 and OntoNotes 5.0.

However, despite their successes, neural networks encounter a significant limitation due to their reliance on substantial amounts of annotated text, presenting challenges, particularly in applications to Electronic Health Records (EHRs). The scarcity of publicly available datasets annotated specifically for medical NLP tasks further compounds this challenge. Recognizing the importance of this issue, the focus of the paper is on enhancing the performance of neural networks when faced with a scarcity of annotated examples.

The comprehensive exploration presented in the paper explores five sequential enhancements meticulously designed to augment the learning capabilities of neural networks in the presence of limited annotated examples. The overarching objective is to optimize NER task performance using a minimal dataset, specifically 10 randomly selected annotated discharge summaries from the i2b2 2009 dataset. By leveraging a state-of-the-art NER architecture as the baseline, the paper systematically showcases the iterative improvements and their cumulative impact on addressing the challenges posed by limited annotated data.

While the enhancements presented demonstrate notable strides in performance, the review also illuminates potential avenues for further refinement and improvement in addressing the critical issue of neural network performance with limited annotated examples.

2 METHODOLOGY OVERVIEW

The NER identification methodologies encompass the utilization of the BiLSTM architecture and various representations of input data. Additionally, this section elucidates bespoke modifications introduced to both the architecture and input representation.

2.1 Original Methodology

The model incorporates three distinct inputs: character-level, word-level, and casing inputs, each encoding different aspects of the text (refer to Figure 1). The architecture initiates independent processing of these three inputs and subsequently merges them for further analysis. The key components and operations of this architecture are described as follows:

The **Character Embedding Layer (char_input)** maps a vocabulary of 97 possible characters to a 30-dimensional embedding. This embedding is initialized randomly from $U(-0.5,0.5)$. The number of input samples per batch ('b') and the number of words per sample ('w') vary from batch to batch. The maximum number of characters per word ('c') is set at 52. Dropout layers (char_dropout1 and char_dropout2) with a drop rate of 0.5 are applied to the character-level input to mitigate the risk of overfitting.

The **1D Convolutional Layer (char_conv)** processes the 1-dimensional character input with 30 kernels of width 3. This layer is followed by a 1D maxpool operation (char_maxpool) with a window size of 52 and a stride of 52, effectively reducing the character dimension to size 1. The kernel is initialized by drawing from a Glorot uniform distribution (Glorot & Bengio, 2010), with bias terms initialized to zero.

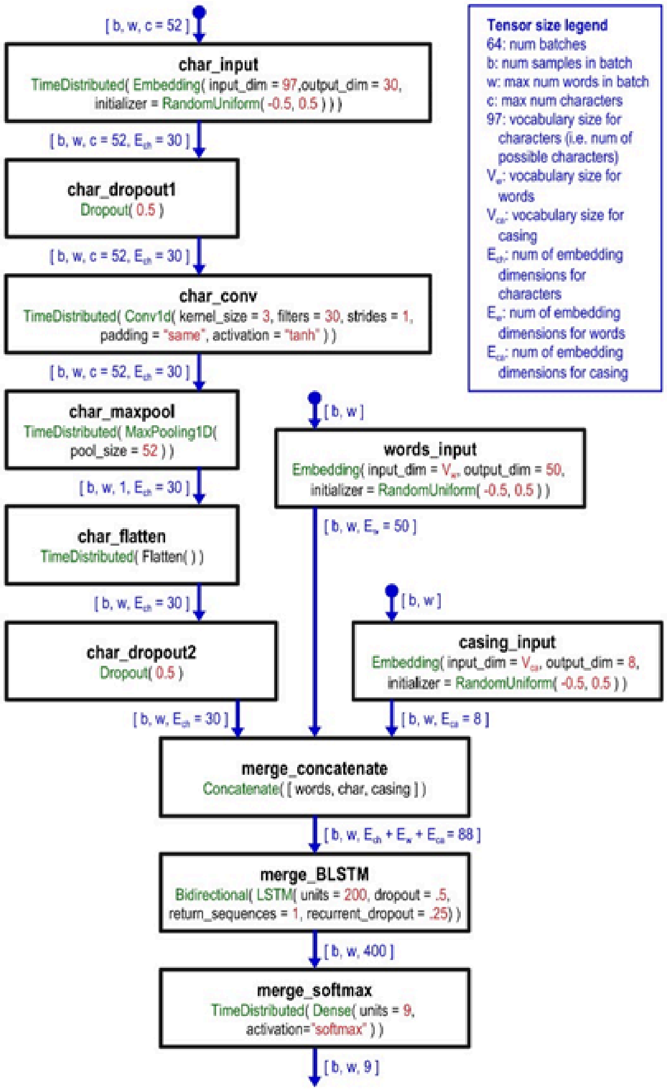
[b, w, c = 52]
char_input
TimeDistributed(Embedding(input_dim = 97,output_dim = 30, initializer = RandomUniform(-0.5, 0.5)))
[b, w, c = 52, E_ch = 30]
char_dropout1
Dropout(0.5)
[b, w, c = 52, E_ch = 30]
char_conv
TimeDistributed(Conv1d(kernel_size = 3, filters = 30, strides = 1, padding = "same", activation = "tanh"))
[b, w, c = 52, E_ch = 30]
char_maxpool
TimeDistributed(MaxPooling1D(pool_size = 52))
[b, w, 1, E_ch = 30]
char_flatten
TimeDistributed(Flatten( ))
[b, w, E_ch = 30]
char_dropout2
Dropout(0.5)
[b, w, E_ch = 30]
[b, w]
words_input
Embedding(input_dim = V_w, output_dim = 50, initializer = RandomUniform(-0.5, 0.5))
[b, w, E_w = 50]
[b, w]
casing_input
Embedding(input_dim = V_ca, output_dim = 8, initializer = RandomUniform(-0.5, 0.5))
[b, w, E_ca = 8]
merge_concatenate
Concatenate([words, char, casing])
[b, w, E_ch + E_w + E_ca = 88]
merge_BLSTM
Bidirectional(LSTM(units = 200, dropout = .5, return_sequences = 1, recurrent_dropout = .25))
[b, w, 400]
merge_softmax
TimeDistributed(Dense(units = 9, activation="softmax"))
[b, w, 9]
Tensor size legend
64: num batches
b: num samples in batch
w: max num words in batch
c: max num characters
97: vocabulary size for characters (i.e. num of possible characters)
V_w: vocabulary size for words
V_ca: vocabulary size for casing
E_ch: num of embedding dimensions for characters
E_w: num of embedding dimensions for words
E_ca: num of embedding dimensions for casing

Figure 1: An overview of model architecture

The **Word Embedding Layer (words_input)** maps a vocabulary of 'Vw' words into 50-dimensional embeddings. Unless stated otherwise, the GloVE Wikipedia 2014 and Gigaword 5 embeddings with 6B tokens (Pennington, Socher, & Manning, 2014) are utilized.

The **Casing Embedding Layer (casing_input)** maps a vocabulary of 'Vca' casing types into Vca-dimensional embeddings. By default, eight casing types are considered: numeric, allLower, allUpper, mainly_numeric (more than 50% of characters of a word are numeric), initialUpper, contains_digit, padding, and other (if no category was applicable).

The **Concatenation Layer (merge_concatenate)** combines processed character-level (a vector of 30 dimensions per sample input), word-level (50 dimensions), and casing (Vca dimensions) data into a vector of 80 + Vca dimensions.

The **Bidirectional LSTM (BLSTM) Layer (merge_BLSTM)** transforms the previously concatenated data into two vectors of 200 units each, with one applying forwards and another applying backward recursion on the input. The kernels are initialized by drawing from a Glorot uniform distribution (Glorot & Bengio, 2010), and bias terms are initialized to zero.

The **Dense Output Layer (merge_softmax)** applies a layer-wise softmax function to produce predictions for locating and classifying sequences of words in the input text. The number of units in this layer depends on the specific objective task. The kernel is initialized by drawing from a Glorot uniform distribution (Glorot & Bengio, 2010), and bias terms are initialized to zero.

2.2 Modified Architecture

In the modified architecture, several key enhancements were introduced to optimize the performance of the model. These adjustments specifically targeted the word-level features, aiming to improve the model's ability to capture semantic nuances in medical text.

Firstly, the original architecture employed GloVE pre-trained word embeddings with 50 dimensions for word-level features. As a modification, this was replaced with **Word2Vec embeddings trained on the Google News dataset.** This change was motivated by the desire to leverage embeddings that have been pre trained on a diverse dataset, including a substantial coverage of medical terminology. The integration of Word2Vec embeddings with medical domain knowledge is anticipated to enhance the model's understanding of medical text, potentially leading to improved Named Entity Recognition (NER) outcomes in the healthcare domain.

Secondly, the **dimensionality of the word embeddings** was adjusted from the original 50 dimensions to 32. This reduction in dimensionality was a strategic decision made to expedite the training process. While maintaining an adequate level of representational capacity, the decrease in dimensionality from 50 to 32 is expected to contribute to faster convergence during training. This modification aligns with the need for efficient training on large medical datasets, where quicker convergence can be pivotal.

Additionally, a shift in the optimization strategy was implemented by replacing the original Stochastic Gradient Descent (SGD) optimizer with the **Nesterov Adam (Nadam)** optimizer. Nadam combines the advantages of both Adam and Nesterov Accelerated Gradient (NAG) methods. The introduction of Nadam brings the benefits of adaptive learning rates, as seen in Adam, while also incorporating the accelerated convergence property from the Nesterov trick. This strategic adjustment in the optimizer aims to improve the model's training dynamics, potentially resulting in a more efficient and effective learning process.

3 Experimental Setup

In our experimental setup, we conducted a comprehensive evaluation comparing an original architecture with a modified version, focusing on Named Entity Recognition (NER) tasks using the CoNLL-2003 dataset. This annotated dataset, comprising newswire articles with word-level NER labels, part-of-speech (POS) information, and word annotations, served as the foundation for our investigations.

Both architectures integrated pre-trained word embeddings in dictionary format, where words from the dataset were associated with their respective embedding vectors. The modified architecture departed from the original by adopting Word2Vec embeddings trained on the expansive Google News dataset. This strategic shift aimed to augment the model's vocabulary, particularly with a focus on medical terminology, potentially enhancing its performance on medical text.

In addition to word embeddings, character-level and casing features played a crucial role in our models. We defined character-to-index (char2idx) and case-to-index (case2idx) dictionaries to map characters and casing types to numerical indices. These dictionaries were instrumental in imparting character-level and casing-level information to the models.

```
char2idx =
["0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJ
KLMNOPQRSTUVWXYZ.,-_()[]{}!?:;#'\"/\\%$`&=*+@^~|
<>"]
case2idx = {'numeric': 0, 'allLower': 1,
'allUpper': 2, 'initialUpper': 3, 'other': 4,
'mainly_numeric': 5, 'contains_digit': 6,
'PADDING_TOKEN': 7}
```

Throughout our experiments, both architectures were executed under uniform conditions in a Colab notebook, ensuring hardware consistency and fairness in the comparative analysis. This meticulous experimental setup allowed us to systematically assess the impact of architectural modifications on NER performance, providing valuable insights into the efficacy of the introduced changes.

4 RESULTS

4.1 Original Results

The original implementation demonstrates commendable performance, yielding a test F1 score of approximately 86 after 30 epochs. Notably, experimentation indicates the potential for further improvement, as extending the training duration to 80 epochs propels the F1 score beyond the 90 mark. This underscores the model's capacity for enhanced performance with prolonged training.

4.2 Modified Implementation Results

In contrast, the modified implementation achieves a test F1 score of approximately 79 within the initial 30 epochs. Although the achieved accuracy falls short of the original paper, a noteworthy advantage emerges in terms of training efficiency. Despite the lower F1 score, the modified model exhibits faster training dynamics, completing each epoch a full minute quicker than its predecessor. Over the course of 30 epochs, this translates to a substantial time-saving of around 30 minutes, highlighting the trade-off between accuracy and training speed introduced by the modifications.

5 CONCLUSION

In summary, the original approach demonstrates robust NER performance, while the modified implementation, albeit achieving a slightly lower accuracy, introduces a valuable trade-off—faster training times, saving approximately one minute per epoch. This highlights the nuanced balance between model accuracy and training efficiency introduced by the modifications.

Considering future advancements, the exploration of transfer learning architectures emerges as a promising avenue to leverage pre-existing knowledge and further boost NER performance. Additionally, addressing the challenge of out-of-vocabulary (OOV) words becomes imperative for more comprehensive and efficient NER applications. These areas present exciting opportunities for refining and extending the capabilities of NLP models in the medical domain.