Voice Prescription using Name Entity Recognition

Nobel Dang[1], Saksham Thukral[2], Ashish Khanna[3]
Maharaja Agrasen Institute of Technology, Delhi, India

nobeldang@gmail.com , sakshamthukral98@gmail.com, ashishkhanna@mait.ac.in

Abstract —Doctor's around the world write prescriptions and hand it to the patients and maintain the record with paper file based systems. This results in very inefficient methods as well as whipping of time. In this paper, software which will automate this process is introduced. The software is deployed on the doctor's computer or mobile in the form of a web application. Whilst the conversation between the doctor and patient is going-on, the software is listening and capturing the entire context. Then the web application will invoke the micro service deployed on the server which runs in the python environment or flask. With the deep learning model deployed on the server three tasks are done: Classification, Summarization and Named Entity Recognition (NER). With the virtue of Recurrent Neural Network (RNN) and Long-Short Term Memory (LSTM), the classification between four classes: Name, Diagnosis, Prescription, and Advice is done on the preprocessed text having the dimensionality of k*20, where k=> number of sentences. The embedding layer is custom trained with the vocabulary size of 5463. After extracting the features, summarization and Name Entity Recognition (NER) tasks are done using spacy and the final PDF containing the required summaries and tagged keywords is generated.

Index Terms: Long Short Term Memory, Machine Learning, Named Entity Recognition, Parts-of-speech, Recurrent Neural Networks.

I. INTRODUCTION

Natural Language Processing (NLP) is a deep learning branch dealing with the language tasks. The applications have been extended from language modeling to much complicated ones' like parts of speech tagging. Parts-of-Speech tagging is a complex task involving each word to be identified with a grammatical group. Parts-of-Speech tags include nouns, verbs, adjectives, pronouns, conjunctions etc. NLP widens the horizon and makes the POS tagging Unrestricted to encompass custom defined keywords like Geo Location, Time Stamp etc. Many models for custom defined keyword tagging have been introduced: Stanford NLP, Spacy Pipeline etc. For this Named Entity Recognition task comes into play which defines whether or not a word is named entity. Named entities can also group and identify the chunk of words. Thus, NER is a much more sophisticated task than POS tagging. NLP does NER tagging by using various methods like IOB or BOI, chunking etc. IOB, or inside outside beginning, a tagging format that is used for tagging tokens in a chunking task such as named-entity recognition. These tags are similar to part-of-speech tags but give us information about the location of the word in the chunk.

Over the years, NER has been combined with web applications or android ones. Likewise, the authors of the paper built a web application to aid the doctors and patients by performing NER on the conversed sentences which helps doctors to send reports to the patients directly digitally without any paper or manual intervention through email. The report includes the chunks of keywords extracted from the conversation like Name, Prescription, Diagnosis, Advice and Symptoms. With the help of Spacy's NLP pipeline and deep learning classification, Dang N, Thukral S. and Khanna A. are able to correctly perform NER and extractive summarization.

II. DATASET

The dataset of diagnosis, prescription and drugs is gathered from the Food and Drug Administration (FDA). It contains 1500 prescribed drug names from which and 401 common disease names.

Accepted 20th April 2020

However, the dataset for Name Entity Recognition (NER) task for Indian names is custom built and augmented, and the entities are self-tagged for training purposes. A total of four entities are tagged as Name, Advice, Diagnosis and Prescription per training sentence atop of the normal Name Entity Recognition (NER) spacy pipeline which can take BERT embedding's and the best model is chosen according to the survey in [3] and [4].

Lastly, we had a total training batch of 7,000 sentences to be classified and tagged against four keywords.

III. PROPOSED WORK

A. *Pipeline*

Firstly, the speech-to-text is done using IBM Watson as implied in [11]. Then the spoken sentences are reshaped and padded to sentences with word-length of 20. Then the functional model is used to add a side branch with the variables initialized according to [10]. Following which is a classifier, as implied in [2] for text classification with loss function as described in [9], with shape and summary as in figure 1.

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 20, 32)	84576
lstm_1 (LSTM)	(None, 20, 64)	24832
dropout_1 (Dropout)	(None, 20, 64)	0
lstm_2 (LSTM)	(None, 32)	12416
dense_1 (Dense)	(None, 4)	132
activation_1 (Activation)	(None, 4)	0

Total params: 121,956
Trainable params: 121,956
Non-trainable params: 0

Fig 1: Classifier Architecture Summary

On the other branch is the Name Entity Recognition (NER) pipeline, as mentioned in [4], to extract the custom made parts-of-speech (POS) tags and the process is improved by the methods of [7]. At last, both the results are used to finally generate a PDF document using PDF generator functionality. The NLP model pipeline is shown in figure 2.

Fig 2: NLP Model

B. *Components*

To build the final model following components are used:

i) *Front End:*

The front-end for the web application is built using Reactjs which uses JavaScript as the backbone. React is a high-level UI building language made and maintained by Facebook. React uses components to build various and render UI. It is a JavaScript library to build dynamic and responsive web pages or web applications. Another approach for summarization is exploited in [16] and has been improvised by Spacy.

The front end contains the interactive webpage with a recording button which when pressed invokes the IBM Watson API to do a speech-to-text conversion with a word-error rate of 0.50 as shown in [11].

ii) *Back End:*

The backend of the web application is written in Flask and python. The flask part is deployed on the local server so as to invoke the calls or micro service. When the speech-to-text task is performed, the handle is given to the model predictor where the Machine Learning (ML) model is deployed in the background server. The Machine Learning (ML) model pipeline is discussed in section 3.1 which are the services offered by the back-end.

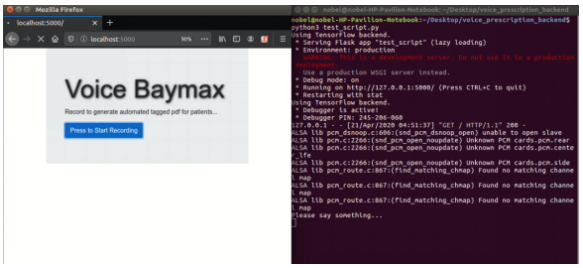

Fig 3: Web Application Front-End and Back-End

iii) Recurrent Neural Network (RNN) with Long-Short Term Memory (LSTM):

Recurrent Neural Network (RNN) or recurrent neural network is a type of neural network generally used for text or NLP related tasks. In this the output of the previous layer is fed as the input to the next layer according to (1) and so on as speculated by the authors of [2].

$$h_t = \begin{cases} 0 & t = 0 \\ f(h_{t-1}, x_t) & otherwise \end{cases} \quad (1)$$

Unlike previous neural networks where the inputs and outputs of each layer are mostly independent, Recurrent Neural Network (RNN) shows dependency to a very high extent. Recurrent Neural Network (RNN) also has a "memory" which stores the context of previous sentence words or layers thus providing a more suitable output or even next word in a sentence.

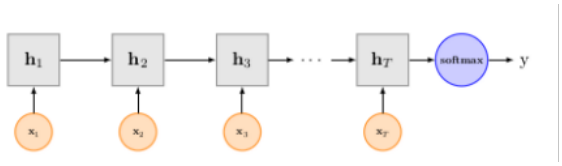

Fig 4: Recurrent Neural Network Architecture

Long Short Term Memory networks – or also known as "Long-Short Term Memory (LSTM)'s" – are another unique form of Recurrent Neural Network (RNN) which is capable of learning long-term dependencies as mentioned by the authors of [1]. They are designed explicitly to avoid long-term dependency problems. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn as shown in [15].

iv) Named Entity Recognition:

Name Entity Recognition (NER) is the process of identifying and categorizing named entities in a given text. Examples of categories are organizations, locations, time, names, money, and rate. Name Entity Recognition (NER) is part of information extraction (IE) or the process of automatically getting structured information from an unstructured document. The Name Entity Recognition (NER) task is performed using spacy which uses deep learning as shown by authors of [4] and [5].

Spacy is a library for advanced Natural Language Processing in Python and Cython. Spacy features and possesses state-of-art speed, parsing and named entity recognition, parts-of-speech (POS) tagging and easy deep learning-NLP integration.

v) Hyper Parameters:

Table 1 show the hyper parameters used and tuned for the NLP pipeline and Machine Learning (ML) model. The NER pipeline is from Spacy, and we are performing the task in English language so only the English model has to be linked.

Table 1: Hyper parameters of NLP model

Name	Value
Input Tensor	1*20
Recurrent Neural Network (RNN) Batch Size	256
Optimizer	Adam
Learning Rate	3e-4
Epochs for Classifier	30
Spacy Pipeline	en_core_web_sm
Dropout	0.2
Xavier Initialization	Seed = 4
Bias Initialization	0.0

vi) Use Case and IDEs used:

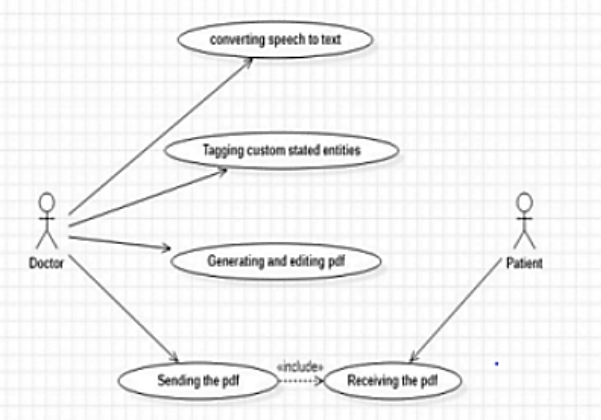

Fig 5: Use Case of Web Application

The IDE used for the development of web applications are Visual Studio Code and Sublime Text. The React components are made in VS or Visual Studio Code and the python files or flask backend in Sublime Text editor.

vii) Conclusion:

The voice prescription requires the doctor to have conversation with the patient and whilst performing extractive summarization by extracting named entity chunks or words as defined by the keywords; Name, Symptoms, Diagnosis, Prescription, Advice. The authors of the paper were able to build the web application with a speech-to-text model to perform NER tasks and generate a PDF. The PDF is then mailed to the patient at his/her respective email address. The front-end is built in Reactjs whereas the back-end in flask or python. To start the recording of conversation and invoke the speech-to-text recorder the recording button has to be clicked. As soon as the recording starts, the flask server logs the starting. After that, the request to the deployed model is made for performing the required text classification and NER tasks.

The NLP model used for this task can further be improvised by having a larger dataset for the IOB tagging. Furthermore, we can use BERT embedding's [14] for extractive summarization [13] which follows text classification. The speech-to-text rate can be improvised by following a survey of the rates by various audio models. The scope can be further extended to mobile web applications or browsers too as done in [17].

REFERENCES:

[1] Sherstinsky A. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network", *2018*.

[2] Liu P., Qiu X. and Huang X. "Recurrent Neural Network for Text Classification with Multi-Task Learning", *International Joint Conference on Artificial Intelligence (IJCAI-16)*.

[3] Li J., Sun A., Han R. and Li C. "A survey on Deep Learning for Named Entity Recognition", *IEEE Knowledge and Data Engineering (2020), PP. 1-1. 10.1109/TKDE.2020.2981314.*

[4] Yadav V. and Bethard S. "A Survey on Recent Advances in Named Entity Recognition from Deep Learning models", *Proceedings of the 27th International Conference on Computational Linguistics, 2018*.

[5] Yao L., Liu H., Anwar W. M., Li X. and Liu Y. "Biomedical Named Entity Recognition based on Deep Neural Network.", *International Journal of Hybrid Information Technology (2015), https://doi.org.10.14257/ijhit.2015.8.8.29.*

[6] K. Fukushima "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biological Cybern. 36(4), 93–202 (1980).*

[7] Jatav V., Teja R., Bharadwaj S. and Srinivasan V. "Improving Part-of-Speech Tagging for NLP Pipelines" *(2017).*

[8] H. Shimodaira "Improving predictive inference under covariate shift by weighting the log-function", *Journal of Statistical Planning and Inference 90, 227–244 (2000)*.

[9] S. Weisler and H. Ney "A convergence analysis of log-linear training", *Advances in Neural Information Processing Systems (NIPS) 24 (2011)*.

[10] X. Glorot and Y. Bengio "Understanding the difficulty of training deep feed-forward neural networks", *Artificial Intelligence and Statics* (*AISTATS) (2010).*

[11] Kim Y. J., Liu C., Calvo A. R., McCabe K., Taylor R. S., Schuller W. B. and Wu K. "A comparison of Online Automatic Speech Recognition Systems and the Nonverbal Responses to Unintelligible Speech", *ArXiv (2019)*.

[12] Kim Y., ''Convolutional neural networks for sentence classification'', *Proceeding Conference on Empirical Methods Natural Language Process. (EMNLP)*, Oct. 2014, pp. 1746–1751.

[13] Moratanch N. and Gopalan C., "A survey on extractive text summarization", *International Conference on Computer, Communication and Signal Processing (ICCCSP),* Jan 2017.

[14] Devlin J., Chang W. M. and Lee K.,Toutanova K., "BERT: Pre-training of Deep Bidirectional Transformers forLanguage Understanding", *NAACL-HLT 2019*.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Comput.*, vol. 9, no. 8, pp. 1735-1780, 1997.

[16] Yih W., Goodman J., Vanderwende L., and Suzuki H., "Multi-Document Summarization by Maximizing Informative Content-Words", *IJCAI,* Vol. 2007. 20th.

[17] Bose J., K P. D, Kasi S and Bhide A., "A framework for text summarization in mobile web browsers", *ICCIC* 2013.