MYCROFT - AN AI APPROACH TO DETECT SEMANTIC DATA TYPES USING NLP AND DEEP LEARNING

Shashank Reddy Boosi

A thesis in fulfilment of the requirements for the degree of Master of Information Technology

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

August 25, 2020

Supervisor: Dr. Wei Wang

ORIGINALITY STATEMENT

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the projects design and conception or in style, presentation and linguistic expression is acknowledged.

ACKNOWLEDGEMENT

I would like to express my profound gratitude to my advisor, **Dr.Wei Wang, Professor** for his support and encouragement of my research project, for sharing his immense knowledge. His guidance helped me to face new challenges that were involved in tackling the project.

Besides the advisor, I would like to thank **Yifang Sun, Research Associate** for his guidance which helped me through difficult situations. I am grateful for all the insights he had which helped me navigate and understand the project better.

My sincere thanks to **Yukai Miao** who helped me to get the access and also set up the environment in a CSE lab machine. He was pretty prompt which helped me to set up very quickly.

I deeply thank my family and friends for supporting me in all the tasks that I have carried for the successful completion of this project.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AI	Artificial Intelligence
CRF	Conditional Random Field
DL	Deep Learning
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
NN	Neural Network
STD	Standard Deviation
WDC	Web Data Commons

ABSTRACT

Detecting data types has been achieved using traditional methods of computer science but semantic data type detection is a challenge because it involves a lot of parameters to consider, especially with the number of semantic types being different for every environment and semantic data types are essential for data mining tasks like data discovery, schema detection, schema matching, and data cleaning, etc. Present day methods to detect semantic types are not robust as their accuracy of detecting is very low. The dataset that we use is taken from WDC 2015 [2] where we train on a different range of web tables ranging from 40000 to 100000 and extract the data columns which range from 10529 to 26143 by matching 78 semantic types from DBpedia to column headers [3]. We map each data column that is being produced from the Web tables into 1588 features using different NLP feature characteristics and then train the features on the model that is being tuned for the dataset. We achieved a accuracy, weighted F1-score, precision and recall of 49.464, 49.870, 54.025 and 49.464 respectively for the dataset produced using 100000 web tables.

KEYWORDS: Semantic Type Detection, NLP, Web Tables, Deep Learning, LSTM.

CHAPTER 1

INTRODUCTION

Many Data Mining tasks like data preparation, analysis, discovery rely on the detecting types to infer rules and constraints on the functionality of the process. Automatic detection of these types is very important which facilitates validation, transformation rules, data statistics, similarities across columns all which are dependant on these types. While many systems still depend on the pre-existing types like string, boolean etc., Semantic types will give us the leverage to perform more cases. Knowing about a value type is very essential in a lot of applications and types like location, birth date, name have quite a different structure in different places and there is a lot of ambiguity in the dataset because of that. Many commercial tool analytics like Microsoft Power BI [4] and Tableau [5] are currently relying on the standard methods like ontology based methods, regular expression patterns, dictionary lookups of the column headers to detect the semantic types but the number of types are limited and are pertained to the non ambiguous ones. Our goal is to improve the performance of these semantic types irrespective of their value structure.

In this project, we considered WDC 2015 Web table corpus [2] as our dataset where we extracted the data from the tables with only 78 semantic types which are described in the T2Dv2 Gold Standard [6], which matches properties from DBpedia [3, 7] with column headers from the corpus. Then we do the exact match between the semantic types that we considered and column headers in the web tables which are extracted in relation with the table orientation. Our transformation after this will consists of a mapping between the column headers with their respective values. We then extract 1,588 features from each column by combining the likes of character level distribution, baseline statistics, word embeddings and paragraph vectors. We then train these features on a novel sequence supported neural network model which achieves a weighted F1-score of

49.870, competing with that of the other baseline models in the same dataset proportion.

CHAPTER 2

RELATED WORKS

Semantic Type Detection has a lot of important applications in many tasks like commercial data preparation, discovery and this kind of detection will help us define, match schemas and will make the life of many data science and AI tasks easier and our work is mainly inspired from Sherlock [1] where they strive to solve this problem using Neural networks like the way we do and we compare our model with Sherlock model as we consider Sherlock to be the standard for Semantic Type detection using DL.

Many commercial tool these days are relying on ontology based methods, regular expression patterns, dictionary lookups of the column headers to detect the semantic types but the types are limited. Open source libraries like messytables [8], datalib [9] use these kinds of heuristics and detect the limited types but remain infeasible due to the amount of semantic types that are available in the data mining tasks.

Due to the high uncertainty of the values of the semantic types, many methods like ontology based, feature based and probabilistic based algorithms couldn't succeed in producing a standard algorithm that will get us close to solving the detection. In ontology based, many researchers tried to use the data from Web Tables [10], DBpedia [3, 7], Wikitology [11] to convert these tables into key-value mappings and use the maximum likelihood estimators where Wikitology [11] helps with the idea of using the column headers and their values to form the dataset.

In feature based, where the data is converted into features which is ontology agnostic.

Ramnandan et al. [12] use Term Frequency and Inverse Document Frequency (TF-IDF) by first separating the numerical and textual types whereas Pham et al. [13] uses slightly more features and their approach is mainly based on extracting similarities of the textual data to train the ML models. Sherlock leverages this kinds of approaches and uses a significantly large varieties of feature characteristics which includes character distributions, word embeddings and paragraph vectors to train the Neural network models.

Finally, in probabilistic based approaches, they use inference procedure by considering a few inferences to detect the semantic types. Goel et al. [14] uses CRF to predict the semantic types based on the values in a column and also assuming that every column is related to the other column in terms of frequency occurrence with in a web table. Limaye et al. [15] uses probabilistic models to consider value as entities, columns as entity types and column pairs with relationships for the graphical model and our approach is quite similar to the probabilistic based models.

CHAPTER 3

METHODOLOGY

The methodology explains the process flow as shown in the figure 3.1.

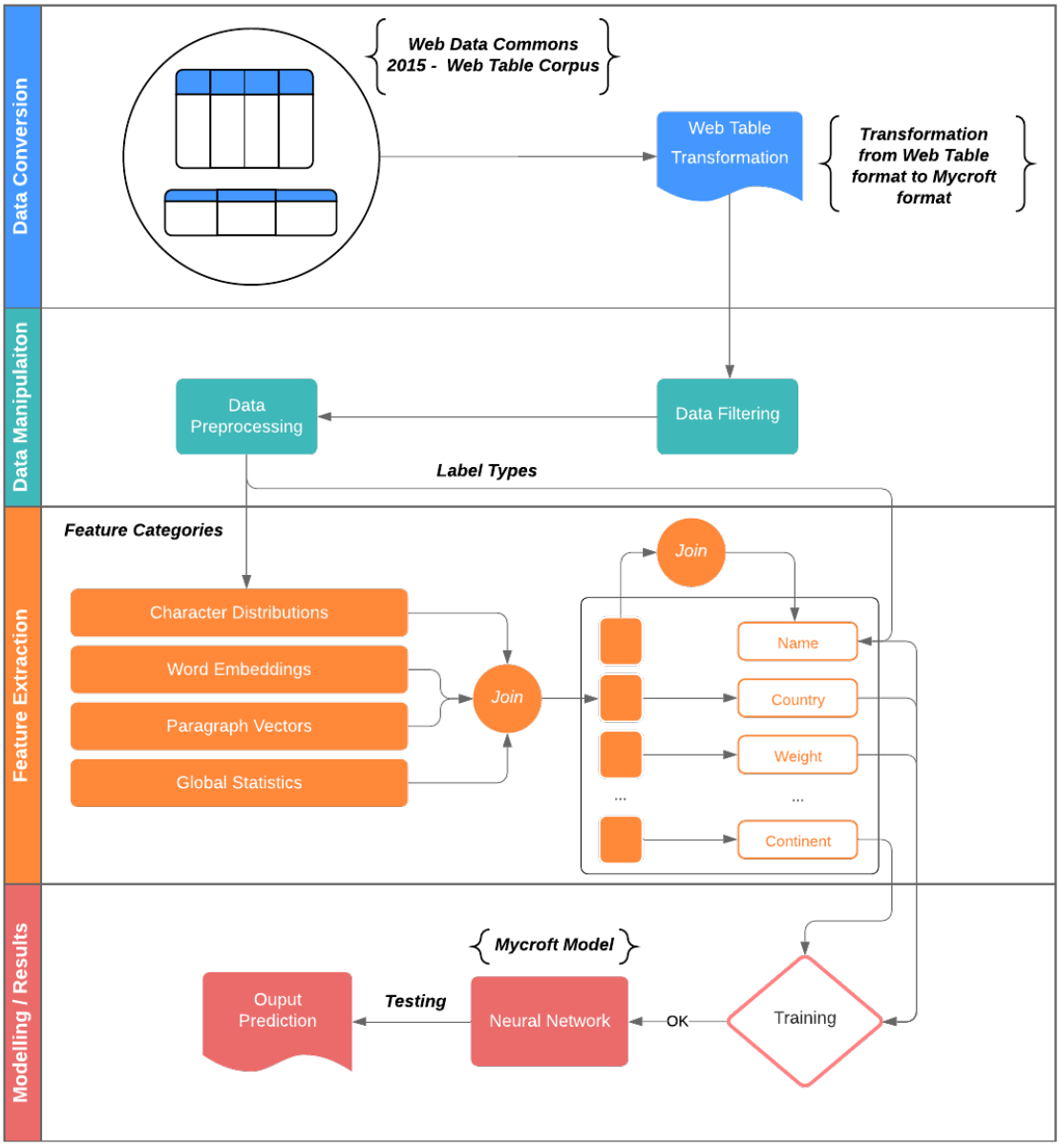

Figure 3.1: Mycroft Architectural Diagram

3.1 DATA

In this section, we will describe how we extracted data columns from WDC Web Corpus 2015 [2] of real-world datasets and also how we did the preprocessing and filtering of the data before extracting features.

3.1.1 DATA COLLECTION

The dataset that we choose for this project is the WDC Web Table Corpus 2015, which parsed 1.78 billion HTML pages and after filtering and preprocessing all the innermost tables which constituted to 10.24 billion, 233 million tables remain. These tables are classified into 4 categories Relational, Entity, Matrix and Layout [2]. We only chose the Relational tables which consisted of 51 million tables.

3.1.2 DATA TRANSFORMATION

Every table from 3.1.1 is represented in the form of a JSON, and every json file consisted of the attributes **relation, pageTitle, title, url, hasHeader, headerPosition, tableType, tableNum, s3Link, recordEndOffset, recordOffset, tableOrientation, TableContext-TimeStampBeforeTable, TableContextTimeStampAfterTable, lastModified, hasKey-Column, keyColumnIndex, headerRowIndex, textBeforeTable** of which most of the information is redundant for our project. So for the transformation, we mainly used the **relation** attribute which consisted of all the information we need for the project and the **tableOrientation**. We only considered the horizontal and vertical orientations and shuffled through the 78 semantic types that is being extracted from T2Dv2 Gold Standard [6] and matching the types from DBpedia [3] to column headers. Finally we do all this operation on 40000, 50000, 100000 web tables and extract the semantic types and their values dataset from the Web Table corpus. After transformation, the number of tuples in the dataset at this stage are 14652, 18257, 36391 for 40000, 50000 and 100000 web tables respectively.

3.1.3 Data Filtering and Processing

Once the data is transformed, we will now move forward and apply the following filtering and preprocessing techniques, so that the model can weigh in all the semantic types rather than only a few of them. They are:

- Remove special characters from the data.
- Remove the data which contains more than 5 percent of the labels in the transformed data 3.1.2.
- Remove the data which contains less than 10 percent of the labels in the refined data which was extracted in the option before.

Figure 3.2 shows the number of columns extracted and their count after the data stage. After all the filtering and preprocessing the data is now ready to move on to the feature extraction stage and the number of tuples in the dataset are 10529, 13034, 26143 for 40000, 50000 and 100000 web tables respectively.

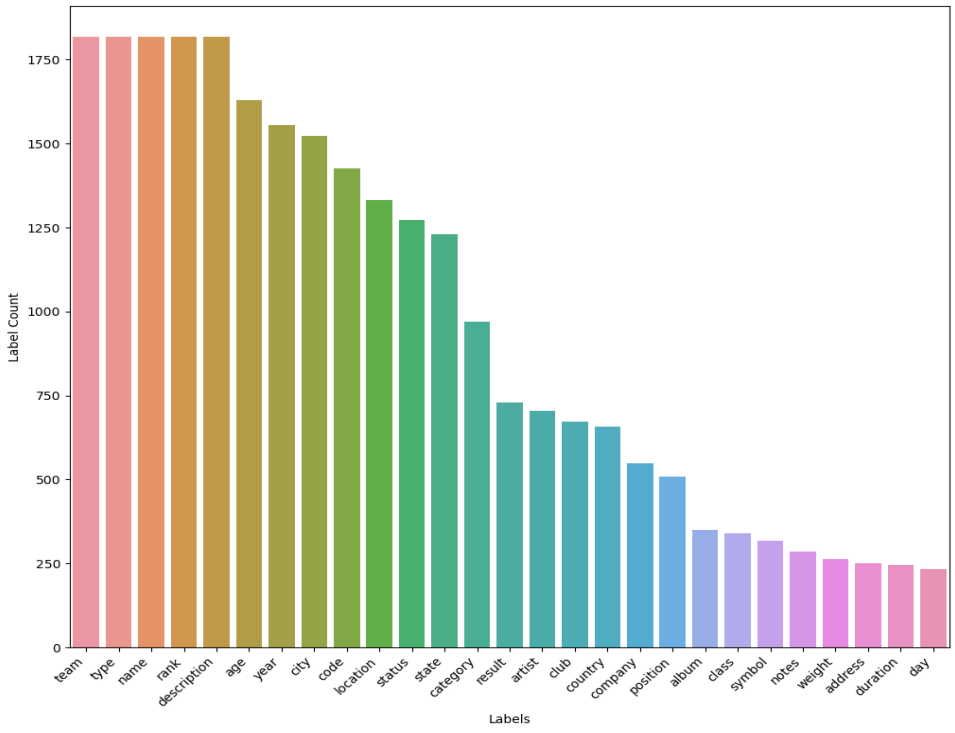

Figure 3.2: Number of columns extracted from WDC and its count

3.2 Feature Extraction

Due to the high variable change in the length of the columns in the dataset, we decided to convert into a fixed-length representation, so that the model training and result extraction would be smooth. The different kinds of feature characteristics [1] that are used in order to make it possible are : global statistics (27), character distributions (960), glove word embeddings (200), and self-trained paragraph vectors (400).

3.2.1 Global Statistics

In this category of features, we have extracted all the high level statistical features that many researchers use to extract meaningful information out of a dataset. Information which describe how uniform the data is distributed, all the numerical features are captured by mean, std etc., All the 27 statistical features are described in the table 3.1

Feature Description
Number of values.
Column entropy.
Fraction of values with unique content.
Fraction of values with numerical characters.
Fraction of values with alphabetical characters.
Mean and std. of the number of numerical characters in values.
Mean and std. of the number of alphabetical characters in values.
Mean and std. of the number special characters in values.
Mean and std. of the number of words in values.
Percentage, count, only/has-Boolean of the None values.
Stats, sum, min, max, median, mode, kurtosis, skewness,any/all-Boolean of length of values.

Table 3.1: Description of the 27 statistical features

3.2.2 Character Level Distribution

Character level based matching has been quite prevalent in many researched which will give a predictive power to the output and as shown in Sherlock [1] it contributes a lot towards the end results. In the character level distributions, we will convert every character in the column and distribute across the 96 ASCII-printable characters which include digits, letters, punctuations ,special characters except the white space. Once the mapping is done, we aggregate all these character counts with 10 statistical features namely any, all, mean, variance, min, max, median, sum, kurtosis, skewness resulting in 960 features.

3.2.3 Word Embeddings

Word embeddings are one of the most popular text based feature extraction technique used in many data science and AI tasks where we use high dimensional fixed-length vectors to represent words. In this project, we used a pre-trained Glove dictionary [16] containing 50 dimensional representation which consists of 400K English words. For each value in a column, we lookup the value in the Glove dictionary, and omit if the term does not exist and multiple words are looked up separately and we take the mean of the vectors as a whole. Finally, once the lookup and mapping is done we calculate the mean, mode, median and variance of the word vectors across all the values in a column adding up 201 dimensions into out feature vector.

3.2.4 Paragraph Vectors

Finally the last technique that we used to convert text into fixed-length numerical vectors is by using Distributed Bag of Words version of Paragraph vector [17]. We will assume every column to be a **paragraph** and each and every value in a column to be **words** and the whole structure is represented by one-hot encoded vectors.

After we pool all columns together across the classes, we will randomly select a window of value vectors and train the model to predict the former from latter by using gensim library [18] over every 20 iterations. Once training is done we will map the columns to a 400 dimensional paragraph vector which will provide us with better predictive power over both training and test sets.

3.3 Neural Network: Mycroft Model

The models used to train and test Semantic type detection are very simple models like traditional statistical models, Machine Learning models such as Logistic Regression, Regular Expression etc., but their functionality is only restricted to a small feature set and these algorithms couldn't account for the high variability of the values of the semantic types. Sherlock was the first of a kind model which uses multi input neural network with feed forward layers which could achieve remarkable results in the space of Semantic Type Detection. Our model is inspired from Sherlock where it could account for high variability of the values for the semantic types and also could predict many semantic types at once. The hyper parameters for the Mycroft model are shown in the table 3.2.

Hyper parameters	**Values**
Metric	Accuracy, F1-score, Precision, Recall
Loss Function	Categorical Cross-Entropy
Optimizer	Adam
Epochs	20
Early Stopping Patience	5
Learning Rate	1e-4
Weight Decay Rate	1e-4

Table 3.2: Hyperparameters used for training

As shown in the figure 3.3, Mycroft NN model consists of multi inputs and each input is assigned a subnetwork and the number of inputs is based on the number of feature categories that are used in the project. We evaluate all the sub-networks individually especially 3 of the 4 subnetworks are assigned the same layers except the sub-network

with the feature category of Statistical features which is just normalized and all the outputs of the sub-networks are concatenated to form the input of the primary network.

Each sub-network consists of 2 Dense hidden layers with varying hidden size ranging from 100 to 400 depending on the size of the input and Relu activation function is used to scale the data and we use dropout and batch normalization to avoid overfitting of the data.

The primary network consists of 3 Dense hidden layers with varying hidden sizes from 78 to 1000 and 1 bi-directional LSTM [19] which will help us remember the sequences of the values in the columns thus boosting the performance of the model. Other than that the primary network consists of Relu activation function, dropout and batch normalization for every hidden layer. The final output results corresponds to the weights assigned to semantic type belonging to each class, the predicted label is then the class with the highest confidence. **Mycroft** model is implemented in Pytorch [20].

Training and Evaluation

The dataset is divided into 80/10/10 training/validation/testing splits and to account for the class imbalances, we evaluate the model performance using weighted F1-score which is the harmonic mean of precision and recall as shown in the equation below.

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall} \tag{3.1}$$

We will also calculate categorical accuracy, recall, precision. The average time taken was in the feature extraction process which took 8 seconds per column.

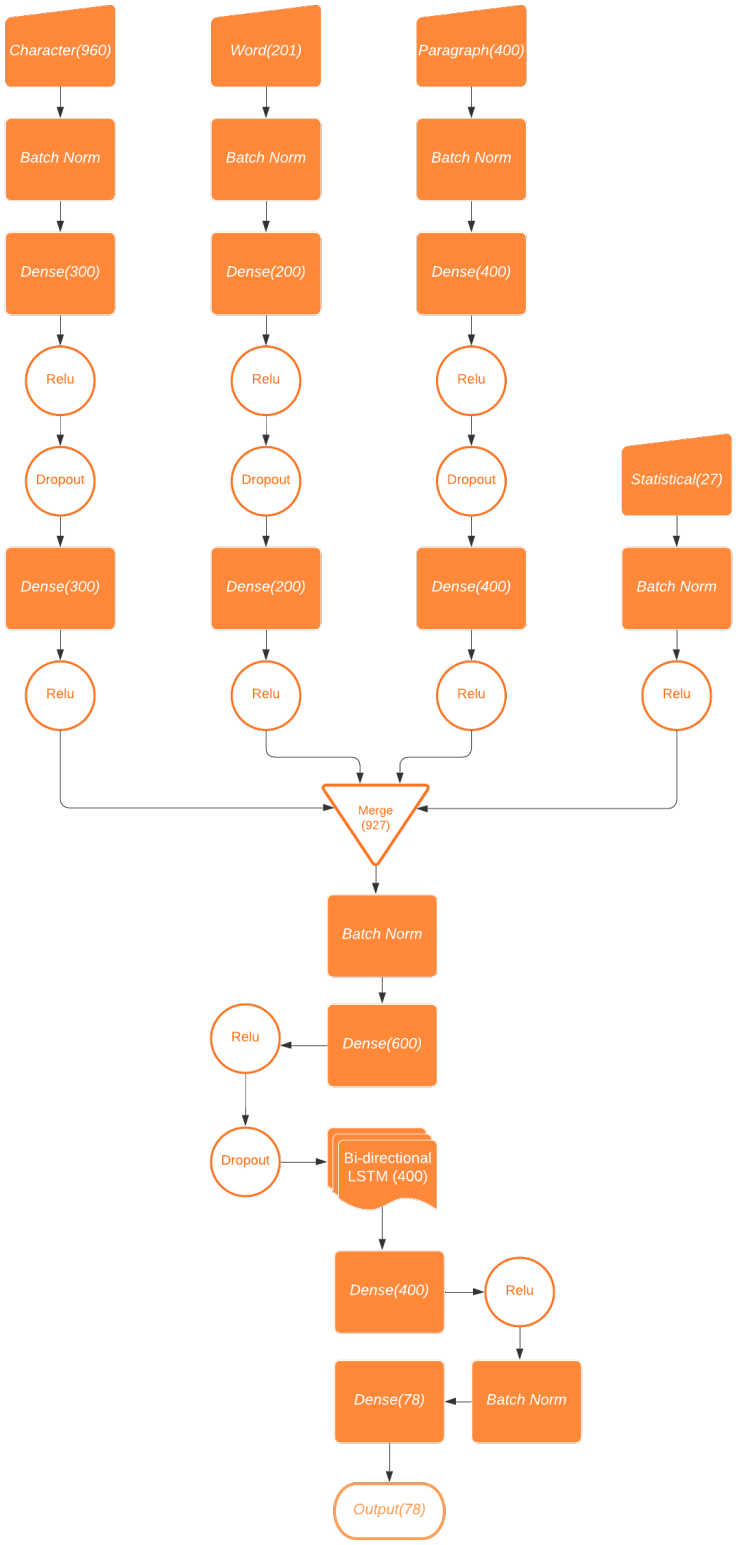

Figure 3.3: Mycroft Neural Network Model

CHAPTER 4

RESULTS AND ANALYSIS

EXPERIMENTS

Many different kinds of experiments are performed to achieve the optimal hyper parameters that are suitable for our dataset. Table 4.1 shows the the comparison between Sherlock [1] and Mycroft with web tables of size 100000. For the dataset of size 100000, we extracted a total of 26143 columns and we compared the Sherlock and Mycroft model and after running both the models against the same data, we outperformed Sherlock both in terms of accuracy and F1-score and the margin of success is too high because of the use of sequence based algorithm LSTM which gives us the extra predictive power by handling the high variability of the values in the dataset.

	Sherlock	**Mycroft**
Accuracy in %	41.41	49.52
F1-score in %	35.19	49.69

Table 4.1: Accuracy and F1-score comparison between Sherlock [1] and Mycroft on the web table size of 100000

Once the comparison with the models is completed we wanted to explore this model by experimenting on the WDC Web Table Corpus 2015 by choosing 3 different dataset sizes of 40000, 50000 and 100000 web tables respectively. Table 4.2 shows the metrics accuracy, F1-score, precision and recall for all the 3 different web table sizes. As you can see in the table, the bigger the size of the web tables gets, the greater the predictive power of the overall model. We experimented this results by only choosing 1 percent of the size of what Sherlock used in order to get their results and yet our results are on par and challenging with Sherlock evaluations. The main focus was calculating the weighted F1-score which takes into account the variability of each classes by weighing them accordingly.

Web Tables	**Accuracy in %**	**F1-score in %**	**Precision in %**	**Recall in %**
40000	44.919	44.225	46.353	44.919
50000	45.168	45.127	47.826	45.168
100000	49.464	49.870	54.025	49.464

Table 4.2: Accuracy, F1-score, precision, recall for different size of web tables [2]

As seen in the figure 4.1, which is a graph between the number of epochs versus the accuracy for different web tables, the accuracy with the 100000 web tables is high right from the beginning when compared to the other web table size. This is because of the dataset being trained with more data which in turn reduces the variability of semantic type values. We can also see that the lines are not in a smooth incremental curve and this is also due to the high uncertainty of the values being exposed while training.

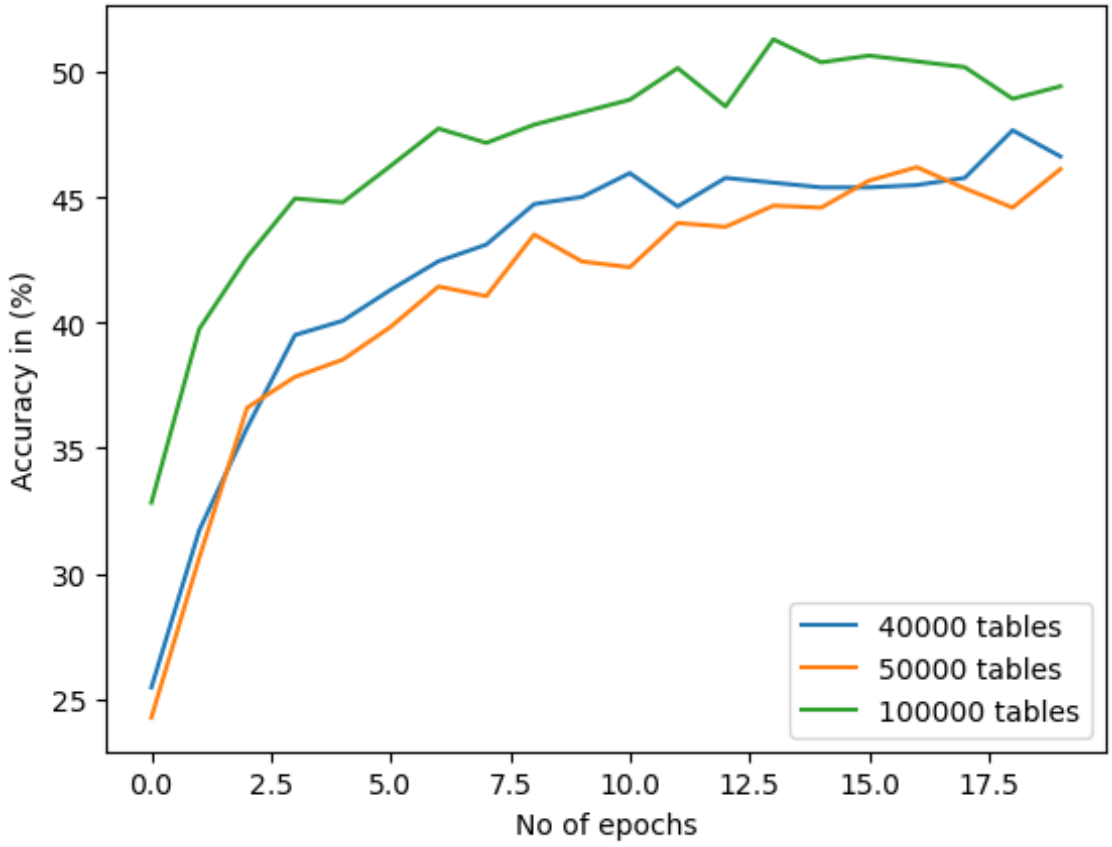

Figure 4.1: Number of epochs vs validation accuracy graph between different web tables

CHAPTER 5

CONCLUSION AND FUTURE WORKS

In conclusion, detecting semantic types is very important in solving many important data science and AI related tasks and we have explored a way to improve the process of detecting the semantic types and the results of this outcome looked more promising as we used sequence based algorithms in our model which gave the boost in performance in terms of all metrics used.

We can further develop the process by applying Conditional Random Fields [21] like linear chain and ordering based etc., to leverage the similarity in terms of the Web Table structure. We can also reduce the uncertainty in the dataset by using more web tables as used in Sherlock [1], which will help us gather different values in the semantic types thus avoiding outlier values whilst testing.

REFERENCES

[1] M. Hulsebos, K. Hu, M. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, c. Demiralp, and C. Hidalgo, "Sherlock: A deep learning approach to semantic data type detection," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2019.

[2] O. Lehmberg, D. Ritze, R. Meusel, and C. Bizer, "A large public corpus of web tables containing time and context metadata," in *Proceedings of the 25th International Conference Companion on World Wide Web*, pp. 75–76, 2016.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*, pp. 722–735, Springer, 2007.

[4] "Microsoft. 2019. power bi | interactive data visualization bi.." `https://powerbi.microsoft.com`.

[5] "Tableau - data analytics.." `https://www.tableau.com/`.

[6] "T2dv2 gold standard." `http://webdatacommons.org/webtables/goldstandardV2.html`.

[7] D. Ritze and C. Bizer, "Matching web tables to dbpedia-a feature utility study," *context*, vol. 42, no. 41, pp. 19–31, 2017.

[8] "Open knowledge foundation. 2019. messytables." `https://pypi.org/project/messytables`.

[9] "Interactive data lab. 2019. datalib: Javascript data utilities." `http://vega.github.io/datalib`.

[10] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: exploring the power of tables on the web," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 538–549, 2008.

[11] Z. Syed, T. Finin, V. Mulwad, A. Joshi, *et al.*, "Exploiting a web of semantic data for interpreting tables," in *Proceedings of the Second Web Science Conference*, 2010.

[12] S. K. Ramnandan, A. Mittal, C. A. Knoblock, and P. Szekely, "Assigning semantic labels to data sources," in *European Semantic Web Conference*, pp. 403–417, Springer, 2015.

[13] M. Pham, S. Alse, C. A. Knoblock, and P. Szekely, "Semantic labeling: a domain-independent approach," in *International Semantic Web Conference*, pp. 446–462, Springer, 2016.

[14] A. Goel, C. A. Knoblock, and K. Lerman, "Exploiting structure within data for accurate labeling using conditional random fields," in *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, p. 1, The Steering Committee of The World Congress in Computer Science, Computer . . . , 2012.

[15] G. Limaye, S. Sarawagi, and S. Chakrabarti, "Annotating and searching web tables using entities, types and relationships," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1338–1347, 2010.

[16] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[17] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, pp. 1188–1196, 2014.

[18] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, Citeseer, 2010.

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

[20] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[21] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," 2015.