**Samtla-Char-NER Report**

**Implementation of Character-based Named Entity Recognition into the Samtla System**

*Student: Matthew Ralph,*

*MSc Computer Science project report, Department of Computer Science and Information, Birkbeck College University of London*

[*mralph02@dcs.bbk.ac.uk*](mailto:mralph02@dcs.bbk.ac.uk)

*Supervisor: Dr Dell Zhang*

*With thanks to Dr Martyn Harris*

*This report is substantially the result of my own work, expressed in my own words, except where explicitly indicated in the text. I give my permission for it to be submitted to the JISC Plagiarism Detection Service. The report may be freely copied and distributed provided the source is explicitly acknowledged.*

# Abstract

Recent approaches to Named Entity Recognition, such as that of (Kuru, Arkan Can and Deniz, 2016), demonstrate that a character-level representation of textual data can yield good results when training a deep learning. In this project, a set of Hansard debates is aggregated, processed and labelled for use in a Bidirectional Long Short-Term Memory (BLSTM) neural network. The trained model, and the original dataset, is integrated with Birkbeck’s Samtla digital humanities text archiving system, such that the Hansard texts can be browsed in the interface, and previously unseen Named Entities are highlighted.

# Acknowledgements

I would like to express my gratitude to the people who taught me to program in Python by working on real problems: to Ali Lotia and Ogonna Iwunze, whose expertise is matched only by their patience and compassion. I would also like to thank Sergio Gutierrez-Santos, whose instruction in the Java programming language was well-structured and helped to open up a world of structured code for me, as well as demystifying unit testing.

I am grateful to Dr Martyn Harris for his help and encouragement when exploring this project and its potential integration with Samtla, and to Dr Dell Zhang for his ideas, advice on the academic landscape surrounding Named Entity Recognition, and quick responses to my queries.

Finally, I would like to thank my wife for all her help throughout this Master’s programme, while she worked on her own master’s and continued to support so many people.

*Ad maiorem Dei gloriam*, from whom all language flows. Ps 19v14.

# List of figures and tables

Compile at end

Table of Contents

[0](#_Toc521058037)

[1. Abstract 1](#_Toc521058038)

[2. Acknowledgements 1](#_Toc521058039)

[3. List of figures and tables 1](#_Toc521058040)

[5. Introduction (including background) 4](#_Toc521058041)

[6. Overall Results (trailer) 4](#_Toc521058042)

[7. Software Architecture 4](#_Toc521058043)

[7.1. The Pipeline of tasks and Invoke 4](#_Toc521058044)

[7.2. Named Entity Downloading 6](#_Toc521058045)

[7.3. Raw Hansard downloading 6](#_Toc521058046)

[7.4. Hansard processing 6](#_Toc521058047)

[7.5. Hansard interpolation 6](#_Toc521058048)

[7.6. Formation of Tensors 6](#_Toc521058049)

[7.7. Overview of files in project and what they do 6](#_Toc521058050)

[8. Implementation issues 6](#_Toc521058051)

[8.1. Wikipedia data cleanliness 6](#_Toc521058052)

[8.2. TWFY API suspect return values 6](#_Toc521058053)

[8.3. NLTK span\_tokenize bugs 6](#_Toc521058054)

[8.4. Toy dataset model – tensor sparsity 6](#_Toc521058055)

[8.5. Hansard Presentation issues 6](#_Toc521058056)

[9. Testing 6](#_Toc521058057)

[9.1. Unit testing 6](#_Toc521058058)

[9.2. Manual evaluation 6](#_Toc521058059)

[9.3. Model cross-validation 6](#_Toc521058060)

[9.4. Overall evaluation 6](#_Toc521058061)

[10. Summary and Conclusions 6](#_Toc521058062)

[10.1. Pre-processing is hard 6](#_Toc521058063)

[10.2. Labelling is hard 6](#_Toc521058064)

[10.3. Sentence tokenization is hard 6](#_Toc521058065)

[11. References 6](#_Toc521058066)

[12. User Manual 6](#_Toc521058067)

[13. Appendix: Code 6](#_Toc521058068)

[14. What’s My Work 6](#_Toc521058069)

# Introduction (including background)

The brief for this project was to demonstrate Named Entity Recognition, using the approach cited in (Kuru, Arkan Can and Deniz, 2016), and the Keras implementation of this provided by GitHub user 0xnurl.[[1]](#footnote-1) The target dataset was the Hansard, the record of debates in both of the houses of Parliament in the United Kingdom.[[2]](#footnote-2) This dataset is now available via the Parliament UK Data API,[[3]](#footnote-3) however this API is largely undocumented and was not available at the start of this project. Instead, I used the They Work For You API,[[4]](#footnote-4) which has all debates from 1919 onwards available for download in a parsed XML format annotated with metadata about the speaker. I did not have time to use this high-quality metadata during the project, and found a few issues with the API (detailed in section 8.2). However, I am grateful for free use of this API which certainly made data preparation easier for me.

In order to implement the model, I had to produce labelled Hansard data. To manually label the few thousand debate documents required to train even a very basic model would have been too time-consuming, for a project of a few months. So, I used a form of labelling I refer to henceforth as ‘interpolation’, the algorithm for which is explained in section 7.5. Interpolation relied on me having a very large set of Named Entities in my chosen categories or locations, organizations and people. I used the DBPedia SPARQL endpoint[[5]](#footnote-5) and Python’s excellent SPARQLWrapper library[[6]](#footnote-6) to download all the Named Entities on Wikipedia in these categories. There were some data cleanliness issues that I never overcame, which are detailed in section 8.1.

I used the interpolated (labelled) Hansards to generate a Y tensor. The processed Hansard debates themselves were chunked into sentences, and then each character was converted to a number, to create the X tensor. I then used the 0xnurl implementation to train the BLSTM model. An overview of results is given in section 6.

I chose this project because of my interest in linguistics and in humanities texts. In my first degree, Classics, I was fortunate to study linguistic change from Classical Greek to *koine*, the language of the New Testament. I also studied some phenomena of Latin that are markers of a particular gender or class. I think that the Hansard is a rich resource to mine, to determine in a principled way, the class- and gender-related semantics of those who govern us. This project is a tiny step, greatly helped by the labours of TheyWorkForYou, to mine the value of the Hansard records.

This project is also in part politically motivated. It is vitally important that democratic citizens re-engage with the task of using factual analysis and solid statistics to make important decisions, rather than being emotionally stirred by the language of tyrants. Learning which companies, places and people we spend most energy talking about as a democracy seems to me, in its own small way, a part of that larger whole.

# Overall Results (trailer)

Do at end

# Software Architecture

## The Pipeline of tasks and Invoke

This project was, in essence, a data pipeline. Data was sourced from Hansard debates and form Named Entities, combined using a variety of algorithms, and then stored in a format that could then be used to predict unseen named entities. As such, it is best visualised using a pipeline flow (see Figure 1). Each element of the pipeline is introduced in more detail in the sections below, along with details of the algorithms and data storage mechanisms used. Implementation difficulties are discussed in section 8.

As so much of this project’s effort was in collecting data for pre-processing,



Figure 1 pipeline data processing model

## Named Entity Downloading

## Raw Hansard downloading

## Hansard processing

## Hansard interpolation

## Formation of Tensors

## Overview of files in project and what they do

# Implementation issues

## Wikipedia data cleanliness

## TWFY API suspect return values

## NLTK span\_tokenize bugs

## Toy dataset model – tensor sparsity

## Hansard Presentation issues

E.g. No speaker information due to XML processing

# Testing

## Unit testing

## Manual evaluation

## Model cross-validation

## Overall evaluation

# Summary and Conclusions

## Pre-processing is hard

## Labelling is hard

## Sentence tokenization is hard

Taught specific abbreviations to the tokenizer. Still bugs outstanding.

# References

# User Manual

# Appendix: Code

# What’s My Work

1. <https://github.com/0xnurl/keras_character_based_ner> [↑](#footnote-ref-1)
2. <https://hansard.parliament.uk/> [↑](#footnote-ref-2)
3. <http://www.data.parliament.uk/dataset/12> and <http://api.data.parliament.uk/> [↑](#footnote-ref-3)
4. <https://www.theyworkforyou.com/api/> [↑](#footnote-ref-4)
5. <https://dbpedia.org/sparql> [↑](#footnote-ref-5)
6. <https://rdflib.github.io/sparqlwrapper/> [↑](#footnote-ref-6)