Samtla-Char-NER

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

Project Proposal

Student: Matthew Ralph, Master's Computer Science, Birkbeck College University of London
mralph02@dcs.bbk.ac.uk
Supervisor: Dr Dell Zhang
With thanks to Dr Martyn Harris

Table of Contents

Table of Contents

1 Introduction....2
2 Project Background....3
2.1 The Samtla System....3
2.2 Character-Based NER....4
2.3 Viterbi post-processing....6
2.4 Hansard corpus....7
3 Related Work....8
3.1 Expectation Maximisation using Tries....8
3.2 Markov Models....9
3.3 Bidirectional Long Short Term Memory Neural Networks....10
4 Proposed Project description....10
4.1 Summary of Features....10
4.2 Data sources and aggregation methods....11
4.3 Machine Learning Algorithm Approach....11
4.4 Evaluation....12
4.5 GUI integration with Samtla....12
5 High-level architecture....13
6 Tools and programming languages....15
6.1 Python....15
6.2 Persisting State....15
7 Methodology / work plan....16
8 Bibliography....17

1 Introduction

This proposal outlines a system for providing new Named Entity Recognition (NER) capability to Birkbeck's Search and Mining Tool for Linguistic Analysis(Samtla), a tool for exploring and analysing digital archives of free text.[1] It aims to learn Named Entities (NEs) at the character-level from a training set in three different categories; people, places and companies.

1 http://samtla.dcs.bbk.ac.uk/

A machine-learning approach is chosen, which will enable the system to classify never-before-seen Named Entities into the correct category, based on internal data from similar words, or words that appear in similar surrounding contexts. This will complement the Samtla system's existing gazetteer NER capability, based on static lists of known NEs.

This Named Entity system will use Hansard as its testbed,[2] the written record of debates in the Houses of Parliament. As such, one of the deliverables of this project will be to load the Hansard text into the Samtla system.

2 Project Background

2.1 The Samtla System

Birkbeck's Samtla system was designed to hold large digital humanities text archives in any language. It provides tools for finding similar passages to the passage currently being examined, free text search, recommendation of similar documents and a Named Entity tool for describing the class of an NE and rendering a map of its location, if relevant.

The Samtla system's design is character-based rather than word-based. It uses a Statistical Language Model (SLM), where documents are represented by n-gram models which store the probability of n-grams occurring in the document (Harris *et al.*, 2014, p. 90). The familiar tf-idf scoring method used in retrieval can then be applied to n-grams, based on the counts of those n-grams and their frequency across the whole collection.

The data structure used for the SLM model is a character-level k-truncated n-gram suffix tree, which can be entirely stored in memory. Character n-grams are taken as sliding windows over the original text, and inserted into an optimised trie data-structure which allows multiple n-grams to be expressed in the same trie. N-grams that share character sequences can use the same nodes in the trie, whereas n-grams with different characters diverge from the existing trie to create new nodes. When the last character of the current n-gram is encountered, represented by a dollar sign ($), a special leaf-node is created, containing metadata describing the total occurrences of the n-gram in a document, and a pointer to that document. By setting the length of the sliding window to k, the trie has a max depth of k nodes.[3]

The character-based approach in Samtla has the following advantages: very little pre-processing of documents is required; the system remains language-agnostic, and has no preference for languages with a clear word-separator like English; sub-word level features can be used which reduces data sparseness; algorithms for search and recommendation use concepts from probability theory which are well-motivated and widely employed. The

2 https://hansard.parliament.uk/
3 K is set to 15 in Samtla, after an analysis of the different languages of the archives it contains, and their average word-lengths.

principal issue with this approach are that it uses a lot of storage space. This issue is mitigated by use of an integer index for the tries, instead of actual character strings; by limiting the trie to a depth of k; and by path compression, which allows special ‘super-nodes’ to hold strings of characters instead of individual characters. This last approach is described in detail at (Harris *et al.*, 2014, p. 96).

The Samtla system currently makes use of a gazetteer, which is a static list of known entities and their category (e.g. person, company, place etc). While this is simple, fast and can provide a very high degree of accuracy with a known dataset, it cannot generalise to recognise or classify previously unseen words as Named Entities, using either internal context (the morphological clues given by character-sequences inside the word), or external context (characters from the surrounding words giving semantic hints about the NE class).

2.2 Character-Based NER

A Named Entity is anything that can be referred to with a proper noun (usually capitalized) (Jurafsky and Martin, 2009, p. 349). Named Entity Recognition (henceforth, NER) is the discipline of extracting entities from free-text data and assigning them to an entity-class (Jurafsky and Martin, 2009, p. 348,350).

For some terms, classification is non-trivial; the term ‘Birkbeck’ could refer to the 19th Century British academic and philanthropist, or the University in London. Deciding which referent is correct requires examining the word context. Hence, hand-crafted part-of-speech-aware approaches to NER have been popular. However, such systems require a lot of manual work and tend to only have domain-specific effectiveness. More recently, recurrent neural networks, which use the ‘memory’ of their context to learn Named Entity classification decisions, have been successful. But using data only at the word-level still leads to the sparse data problem: assuming an attempt to classify a previously unseen NE, the chances of its inhabiting a specific, previously-seen word context that gives high confidence in its class, is unlikely.

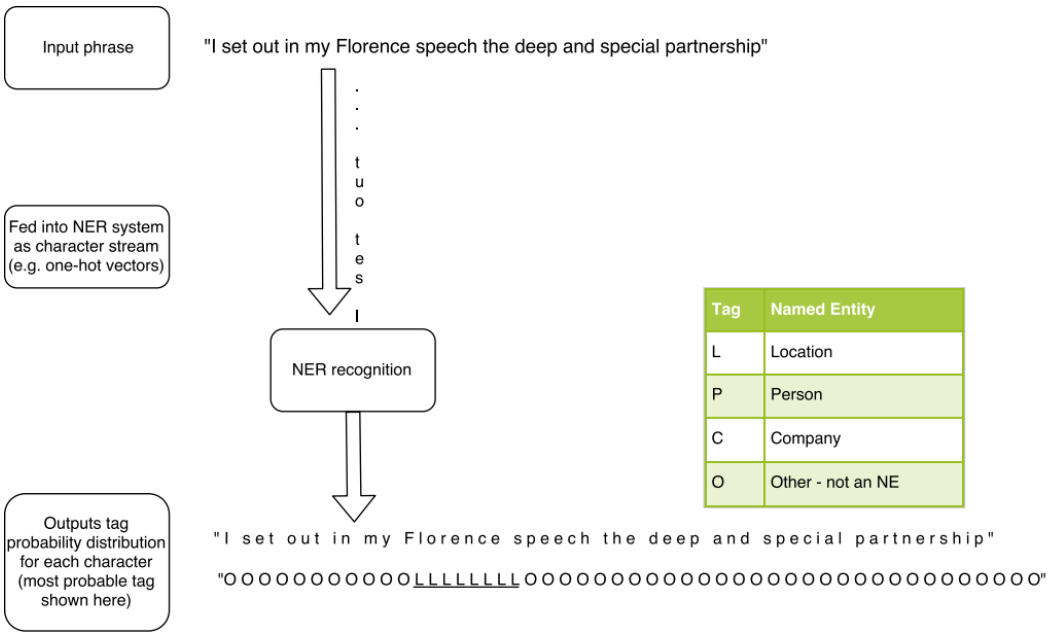

Figure 1: Character-level Named Entity Recognition tags

More recent approaches to the NER task have replaced word-tokenization with character-level processing, as shown in Figure 1; text is fed into a character-based NER system as a stream of characters, and a tag probability distribution per character is emitted. This has several advantages.

Firstly, it avoids all word-separation issues. These are particularly problematic in languages like Chinese, which use a large library of pictorial characters with various possible segmentations, or Japanese, which uses a syllabary combined with Chinese characters. However, agglutinative languages like Turkish, and languages with highly generative morphology (e.g. Finnish, which has around 15 different cases) also perform poorly at the word level, as the data for a particular word are very sparse if it can have multiple morphological representations (such representations are considered, in a word-based system, as entirely separate terms, unless language-specific morphological knowledge is encoded in the system). Even in the English language, which is an ‘analytic’ language (meaning that it has low word-inflection), (Kuru, Arkan Can and Deniz, 2016) show that “taking characters as the primary representation is superior to considering words as the basic input unit” (Kuru, Arkan Can and Deniz, 2016).

Secondly, considering a text as a stream of input-characters avoids the necessity of manually writing and maintaining a set of word-internal features for recognising and categorising previously unseen Named Entities. It provides a principled, consistent way to include word-internal features, derived from the statistical representation of the character stream.

Generally, statistical machine-learning approaches can identify possible features which even human subject-matter-experts would not think of. As Graham puts it in the context of spam-filtering,

> *I thought I was being very clever, but I found that the Bayesian filter did the same thing for me, and moreover discovered a lot of words I hadn't thought of. (Graham, 2004)*

So too in the NER context, a statistical approach has the ability to discover character-based word-internal or word-external (contextual) features that might not stand out to a human rule-writer. Word-internal features can suggest the entity class of an unseen word due to phonaesthetics, sound-sequence features of a word that give clues as to its meaning. So, for example, it is simple for a human to classify 'Novo-Doxylin' as a drug, even if we have never seen it before (see (Smarr and Manning, 2002)). Contextual features allow words before and after the NE to give clues about its class. E.g. the word 'said', occurring after an NE, strongly marks it as a person, or possibly as a company. A statistical, character-based machine-learning model can use these clues without a list of explicitly engineered features.

2.3 Viterbi post-processing

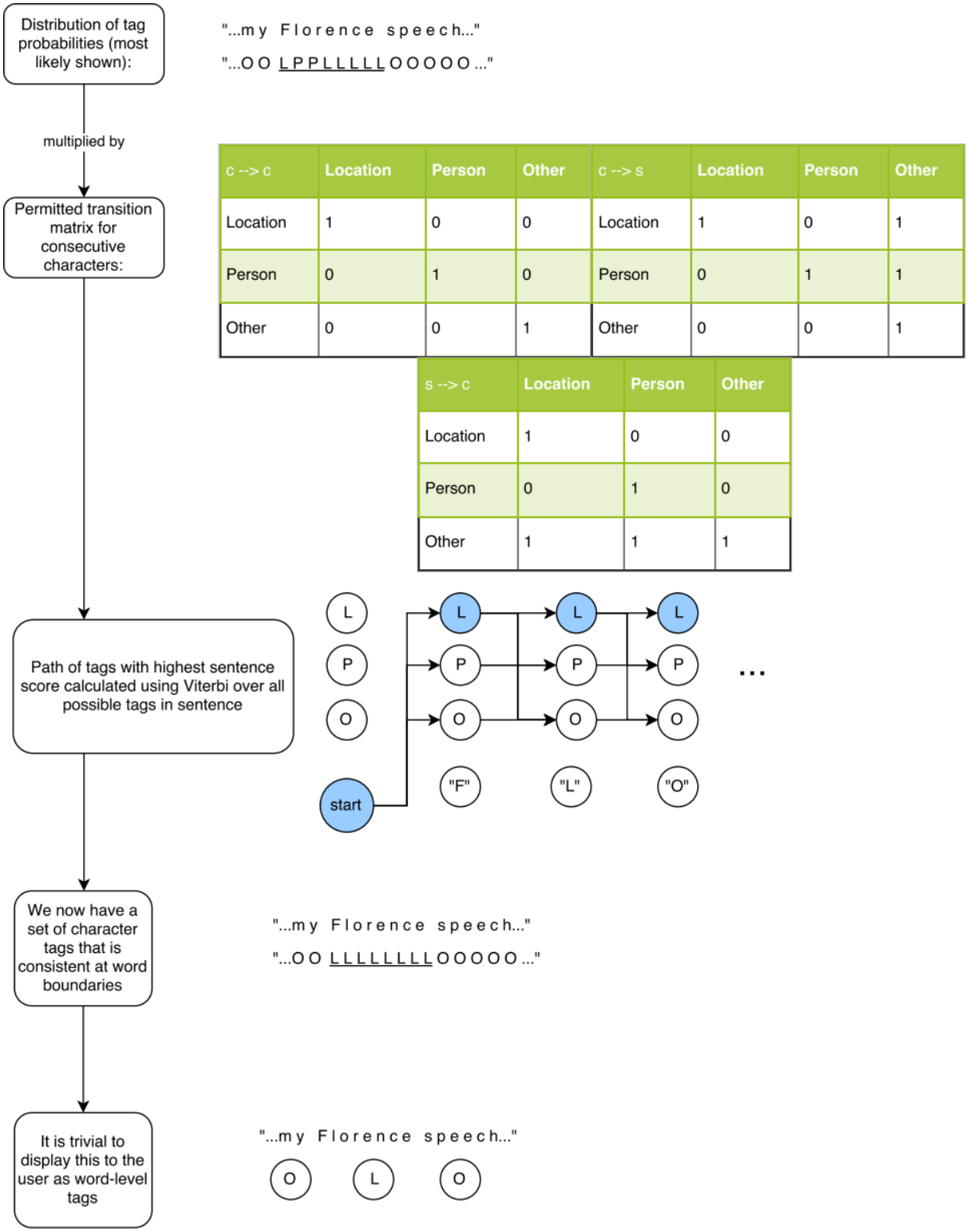

Figure 2: Viterbi post-processing for character-level data

The outputs from character-level analysis often need some form of decoding. Figure 2 demonstrates why this is the case, using an example phrase from Hansard, ‘I set out in my

Florence speech the deep and special partnership...', the permitted NE state transition matrix from (Kuru, Arkan Can and Deniz, 2016), and a representation of the Viterbi decoding process influenced by (Jurafsky and Martin, 2009, p. 219).

It shows that, as character-level processes can by definition have no word-level awareness, the tags produced by such systems will not always respect word boundaries. In the fictitious example in Figure 4, the word 'Florence' has some characters marked as a Person (P), possibly due to instances of the man's name 'Lorence', a variant spelling of Lawrence, in the data. The solution shown in the figure is that of (Kuru, Arkan Can and Deniz, 2016): first the output probabilities from the character-based NER system are multiplied by numbers in the transition matrix.[4] Illegal transitions, e.g. from Person to Location mid-word, are multiplied by 0 and so have no probability mass.

The resulting probability distribution is fed into a Viterbi decoder, which uses a dynamic programming algorithm to build up a trellis,[5] computing the highest-probability path through the whole sentence. Once this path is determined, it *must* be consistent at the word-level, thanks to the transition matrix minimising the score for any path that contains an illegal transition. Then, it is trivial to render NE tags to the user at the word-level. This process results in more accurate word-level NE tags than simply taking a 'majority vote' of the most-likely tags from all characters in the word.[6]

2.4 Hansard corpus

As a 'test-bed' for a machine-learning model of NER, this project will use the Hansard, the written record of the speeches of all formal sittings in the Houses of Parliament (the Commons and Lords, including Westminster Hall debates). As the 'primary text' of enacted democracy in the UK, there is a public interest in analysing what our elected representatives are discussing. Such discussions can ultimately result in primary legislation or in government policy, and are a good measure of the nation's topical focuses.

A new Hansard is produced after each day of debate, which occur Monday-Thursday most weeks in both Houses, with recesses in holidays. As such, it is a living document, with new content and new previously-unseen words occurring continuously (as per Heaps' law, $V_R(n) = Kn^B$, where V_R is the total vocabulary size, n is the size of the text in question, and K and B are set empirically). It is of particular interest to track which Named Entities are being discussed, and with what frequency. For this project, companies, people and places have been selected.

---

4 The matrix is comprised of three tables for different types of transition: c→c is for the transition from non-space character to non-space character, c→s from non-space character to space character, and s→c from space character to non-space character.
5 See (Jurafsky and Martin, 2009, p. 219) for description of trellises to represent dynamic programming algorithms.
6 The voting approach ignores both cases of close ties between two different tags in several characters, and the probability distribution of the surrounding context characters.

The Hansard will be the first such ‘continuously updated’ document to be integrated with Samtla, and so it is appropriate that a Named Entity Recognition method is chosen which can classify unseen words, based on training.[7]

3 Related Work

What follows is a brief appraisal of models used in previous approaches to NER at the character-level.

3.1 Expectation Maximisation using Tries

(Cucerzan and Yarowsky, 1997) use an Expectation Maximization (EM)-style bootstrapping algorithm, starting with only a seed-list (100 items or so) of examples in each Named Entity category, and unannotated texts for training and testing. Also provided are parameters about whether capitalisation is significant in the language in question, what the word-separators are, and a small number of frequent exceptions (like the English word “I”, capitalised but not a Named Entity). The algorithm then builds character-level prefix tries, where

> [t]he distribution[s] stores at each node contain the probability of each name class given the history ending at that node (Cucerzan and Yarowsky, 1997, p. 91, 2nd column).

Considering both morphology and context, the algorithm bootstraps using the seed list, adding newly discovered Named Entities to each class. It only adds words for which the certainty of their class is over a confidence threshold. The probability of a stream of characters belonging to an NE class is given by

$$p(class_j|l_1,l_2\ldots l_n)=\sum_{(i=1)}^{n}\lambda_i p(class_j|l_1,l_2\ldots l_i) \quad (1)$$

where

$$\lambda_1\leq\lambda_2\leq\ldots\leq\lambda_n \quad (2)$$

The following conditions also hold:

- $\lambda_i \epsilon[0,1]$ (3)
- $\sum_{(i=1)}^{n}\lambda_i=1$ (4)

In other words, characters are weighted, with the probability of a character indicating a class growing as the number of characters grows longer. They also have a hyper-parameter for setting an initial probability for the first letter. Their evaluation and analysis shows strong results for English and Romanian, which have many NEs starting with a capital letter, but weaker results for Hindi and German (Cucerzan and Yarowsky, 1997, p. 97).

---

7 Any ‘automatic-loading’ mechanism to update Samtla with new Hansard documents is out of scope of this project, but in principle, the fact that Hansard continuously introduces new vocabulary, is a motivation for integrating a non-static form of NER.

They show that using a list of roughly 100 seed words provides variable results; a given word on the seed-list might often in the same context, which does not add much new information for the bootstrapper to use to identify new NEs. However, as the size of either the seed-lists or the training data (unannotated text) increases, the F-measure of performance increases near-logarithmically. This is good news, as it suggests that using more web-scale datasets may yield better performance and counteract the requirement for language-specific parameters.

Finally, they use a concept of unassigned probability mass (assigned to its own pseudo-class 'questionable'), instead of using maximum entropy smoothing. This creates a distinction between a MaxEnt 'smoothed' data model, and a model in which there happens to be high confidence in an even probability distribution.

(Cucerzan and Yarowsky, 1997) show that character-level NER is feasible, but more recent approaches do away with the need for setup parameters, and are able to obtain stronger results in languages which do not mark NEs with an initial capital letter.

3.2 Markov Models

(Klein *et al.*, 2003) write a character-level Hidden Markov Model (HMM) with minimal context, and a Conditional Markov Model (CMM) with more contextual features, using characters and introducing character n-grams to recognise NEs. They point out that, for previous models which use entire words as input features, data sparsity has meant that most of the engineering effort is generally spent on crafting word-internal features.[8]

The HMM has hidden states depending on the previous 1 state, and emitted characters depending on the current (hidden) state, and the previous *n* - 1 characters (thus using n-grams, and borrowed from (Smarr and Manning, 2002)). They find that 6 is the empirically optimal value for *n*. To prevent different characters of the same word from being allocated different NE classes, they use a 'transition topology':

> each state is a pair (*t,k*) where *t* is an entity type … and *k* indicates the length of time the system has been in state *t (Klein et al., 2003)*.

E.g. (PERSON,2). The space following a NE has special state F e.g. (PERSON,F). The output from the HMM is then Viterbi-decoded to uncover the best-fit sequence of NEs.

(Klein *et al.*, 2003) find that the HMM performed 30% better than an equivalent word-level model, and that adding gazetteer entries into the training counts actually reduces performance.[9]

Finally, they improve upon this performance by building a character-based CMM. This allows current classification decisions to interact with previous ones - for instance, 'Grace Road', which is probably a street location and not a person's name. Part-of-speech tags and other language-specific features further improve performance.

---

8 '[I]n these tasks, providing correct behavior on unknown words is typically the key challenge'.
9 The gazetteer was found to 'provide only a flat distribution of name phrases whose empirical distributions are very spiked'.

(Klein *et al.*, 2003) show that the task of NER can be achieved without requiring word-internal feature engineering. They also demonstrate the effectiveness of using a Viterbi-decoder for post-processing of the results to obtain the best-fit set of NE tags, which is picked up by (Kuru, Arkan Can and Deniz, 2016) (see below).

3.3 Bidirectional Long Short Term Memory Neural Networks

(Kuru, Arkan Can and Deniz, 2016) use stacked Bidirectional Long Short Term Memory (BLSTM) neural networks to process characters in a forward and backward direction (allowing context from both sides of the current character to influence the tag decision), and output tag probabilities which are again fed into a Viterbi decoder. This demonstrates impressive performance in seven languages with 'no hand-engineered features or other external resources like syntactic taggers or Gazetteers'. They explicitly note that their work is 'motivated by the desire to eliminate the tedious work of feature engineering'.

Their model takes a sequence of characters (represented by one-hot vectors) and outputs a tag probability distribution for each character, i.e. the probability that that character is a member of each NE class. The BLSTMs avoid the vanishing and exploding gradient problems by using a memory cell to control exactly which long-term dependencies the model 'remembers' from the character stream. So for a given character sequence $c_i...c_n$, the output probability (o) of character c_i having a particular tag y_i is given by

$$o(c_i)_{yi} = p(y_i | [c]_1^i, [c]_i^n) \quad (5)$$

where

$$[c]_j^k = c_j, c_{(j+1)}, \ldots, c_k \quad (6)$$

However, they still need to use dropout (turning off random bits in the sequence at the input layer), to avoid overfitting the data.

A transition matrix is used to ensure that the NE type of an entity cannot change within characters of the same word, and a softmax function and Viterbi decoding are used after the BLSTM processing, to choose the best sequence of NE tags.

Kuru, Can and Yuret's code is open-source,[10] so integrating their approach with Samtla is a feasible aim for this project.

10 https://github.com/ozanarkancan/char-ner or a tensor-flow based implementation at https://github.com/0xnurl/keras_character_based_ner which uses standard Python libraries so will be easier to integrate.

4 Proposed Project description

4.1 Summary of Features

The proposed system will load Hansard debates using the They Work For You (TWFY) Application Programming Interface (API),[11] which provides the Hansard debates in Extensible Markup Language (XML) markup together with metadata about the speaker of each paragraph of the text. These XML documents will need pre-processing (removing markup tags, n-gram sliding windows), before submitting the documents to the Samtla suffix-tree API.

The proposed system will train a model based on data-sources listed below, to recognise Named Entities in the classes of Company, Person and Place.

The proposed system will fetch data from the trained model when a text is requested by a Samtla user, and will render the classified NEs onto the Samtla text-exploration screen, so that a user can view these.

4.2 Data sources and aggregation methods

The following data sources will be used to build training lists. Not that while a character-based approach is language-agnostic, for simplicity I will focus solely on data in English:

- DBPedia public SPARQL API [12] - a publicly available API for SPARQL[13] queries which contains structured data derived from Wikipedia. This datasource may need some post-processing or truncating, as it is not always reliable – for instance, 'The 14th Century' is listed as of type Person.[14]
- NASDAQ dataset of listed companies[15]
- London Stock Exchange dataset of listed companies[16]
- Biography-center.com human names dataset[17]
- CONLL 2003 English NER List dataset[18]

---

11 https://www.theyworkforyou.com/api/

12 http://wiki.dbpedia.org/ and also http://dbpedia.org/sparql. The following ontology resource classes can be used: http://dbpedia.org/ontology/person, http://dbpedia.org/ontology/PopulatedPlace, http://dbpedia.org/ontology/Company

13 https://www.w3.org/TR/rdf-sparql-query/

14 http://dbpedia.org/page/14th_century

15 https://www.nasdaq.com/screening/company-list.aspx. I concatenate the NYSE, AMEX and NASDAQ datasets into one list

16 http://www.londonstockexchange.com/statistics/companies-and-issuers/companies-defined-by-mifir-identifiers-list-on-lse.xlsx

17 http://www.biography-center.com/ taken from (Smarr and Manning, 2002).

18 https://www.clips.uantwerpen.be/conll2003/ner.tgz

4.3 Machine Learning Algorithm Approach

I will write a set of Python scripts to programatically aggregate data from the sources listed above, and produce labelled Hansard debate documents using this data, where any mention of Named Entities in the text is labelled with the correct classification.

I will then run the BLSTM from (Kuru, Arkan Can and Deniz, 2016), starting with its open-source Keras implementation.[19] I will use the labelled Hansard data for training and will fork and modify the implementation as required to train the model. The trained state will be persisted (see section 6.2 below).

Following from (Klein *et al.*, 2003; Kuru, Arkan Can and Deniz, 2016), the output of a character-based Named Entity classifier should then be Viterbi-decoded (see section 2.3). The Keras-based implementation used as my starting-point has not implemented this feature. This can again be viewed as an extension to the work; the system can start by using a simple majority-voting scheme between characters, adding Viterbi if time allows (see section 7).

Any integration piece with an external system represents a risk. If integration with Birkbeck's Samtla system is not possible for technical or administrative reasons, I will aim to construct a simple Graphical User Interface (GUI) using Bootstrap libraries[20] to display the Hansard text decorated with its categorised NEs.

4.4 Evaluation

Textual snippets of Hansard will be taken unlabelled, and then run through the NE classification model. These will be evaluated against the same snippets, but semi-automatically labelled using the data sources listed above. This will be done using k-fold cross-validation, to ensure the system is not tested on data it has been explicitly trained on.

In NE classification, as in many other areas of Information Retrieval, Precision and Recall are the standard measures. We will take F1 as our combination of these, judging precision and recall to be of equal importance in this task. Human performance on the NE task has an F1 of about 96% according to (Cucerzan and Yarowsky, 1997, p. 96), which represents the best possible score.

4.5 GUI integration with Samtla

Samtla uses Django to manage its Model View Controller (MVC) web-system architecture. For Named Entities, Samtla loads javascript into the browser, which uses a jQuery Asynchronous Javascript And XML (AJAX) call back to the Django controller with the document ID and NER type in the Uniform Resource Indicator (URI). The Django controller in turn uses the Python xmlrpc library[21] to call an internal API (each corpus has its own

19 https://github.com/0xnurl/keras_character_based_ner
20 https://getbootstrap.com/
21 https://docs.python.org/2/library/xmlrpclib.html

Transmission Control Protocol (TCP) port number on which it listens for these queries), and receives back Javascript Object Notation (JSON) data, which it then returns to the AJAX call to decorate the document. The JSON data is simply an object containing key-value pairs, where the key is the NE type, and the value is a list of pairs, indicating the first and last character in the text which is part of that NE. Here is an example, using the 'Occupation' NE data stored for the King James Bible dataset in Samtla at present:

```
"Occupation": {
    "Midwife": [[3598, 7203]],
    "Prostitute": [[1734, 3478], [2561, 5132], [2635, 5280], [2762, 5534], [3005, 6020]]
}
```

Figure 3: NE JSON Data format in Samtla

Integration with this system would require adding a new /Dynamic-NER route to Django, to be used just for the Hansard corpus. This route would then similarly make a backend API call to get the data, which would be provided by the new machine learning model described above. When the user clicks on the NER tab, and is viewing Hansard as their current corpus, the javascript responsible for calling /NER on the Samtla backend would instead call the /Dynamic-NER route, and receive back JSON data as before. In this way, the new NER functionality could be trialled on the new corpus, without affecting the existing NER functionality on current Samtla corpora.

5 High-level architecture

A high-level logical architecture diagram is given in Figure 2. Boxes with green highlighting represent the existing Samtla system. Boxes with purple highlighting represent existing third-party systems, and boxes with blue highlighting represent new code that this project will deliver. Boxes with cream highlighting indicate existing Samtla code that may need to be changed for integration purposes.

The new code contributed by this project is summarised as follows:

1. Hansard Raw Data ingest – Python code to download Hansard documents, using the TheyWorkForYou API (which takes unstructured PDFs from the gov.uk web pages, and produces structured XML documents with metadata tags about the speakers), and parses the XML into a format usable by Samtla.
2. NE Data ingest – Python code to collect NE examples from the DBPedia SparQL API (which itself is populated with Wikipedia data), and other sources as listed above under 'Data Sources and Aggregation Methods'.
3. Hansard documents will be semi-automatically labelled by interpolating NE data to NEs in the text, from the static lists gathered in 2 above.
4. Machine Learning Model – fork and modify the 0xnurl implementation of the BLSTM NER approach using Keras. Add post-processing Viterbi decoding as a possible extension.

Dotted lines indicate tentative integrations – at this stage, it is not yet clear whether the Samtla Corpus API could be modified to allow it to use the new NE model's state directly, or whether the Django App would need to be changed to bypass this, and call the new model directly. In any case, the Hansard data will need to be loaded into Samtla's data backend – if there is an API prepared for this as planned, then this will be used. Otherwise, the data will need to be loaded manually.

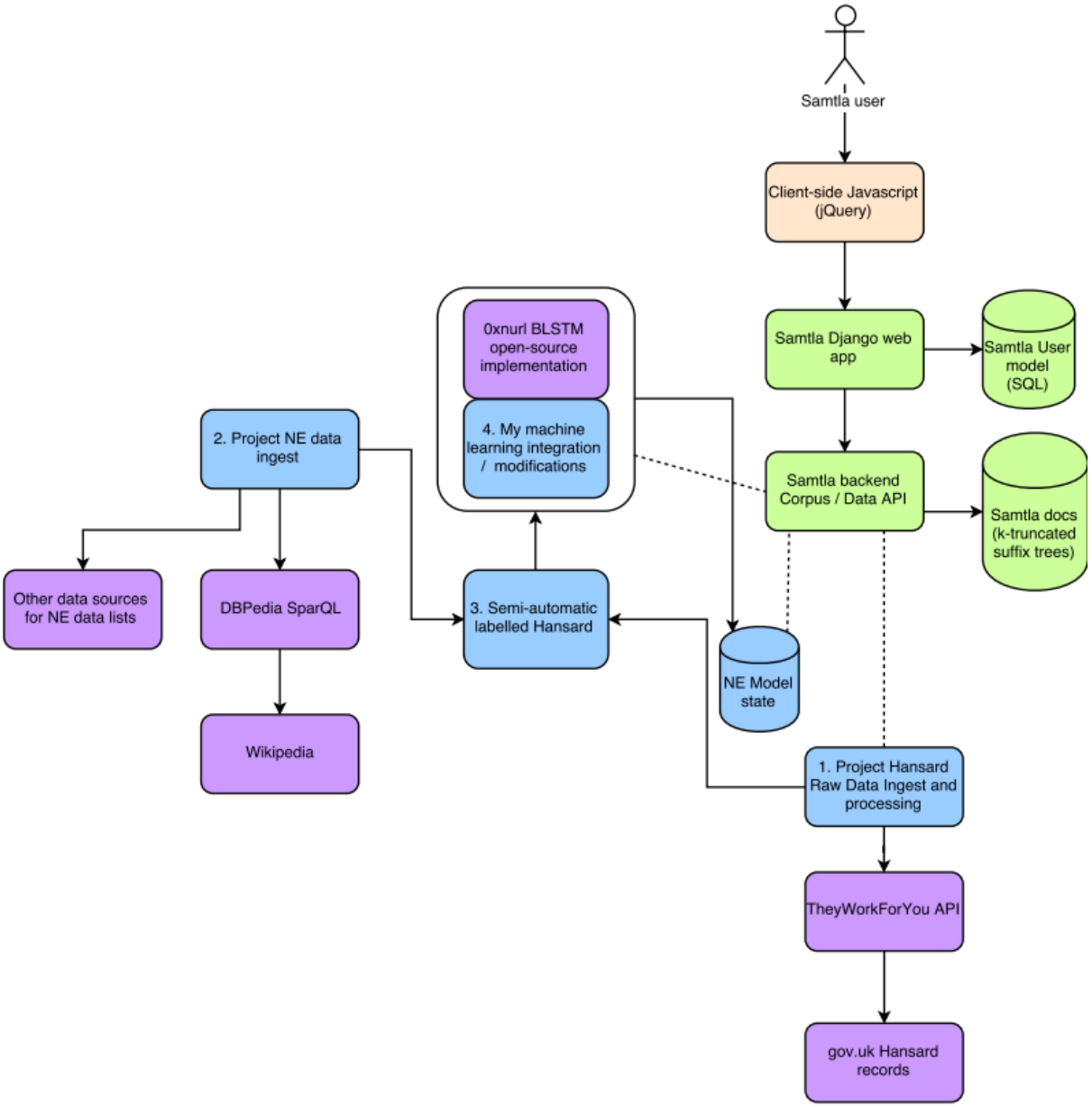

Figure 4: High-Level Logical Architecture

6 Tools and programming languages

6.1 Python

Python will be used for this project, due to the huge amount of libraries and support available in the Natural Language Processing (NLP) and Deep Learning domains. It is also the language used for the backend code of Samtla, which employs the Django framework[22] and an MVC architecture, a widely-followed standard for web applications.

The Samtla code-base has recently been updated to use Python 3, so this is the obvious choice for the Named Entity Recognition code. Python 3 has the advantage of native type annotations,[23] which greatly simplify the debugging process during development, as well as simpler and more consistent object-oriented semantics than Python 2, and native handling of unicode strings.

The following libraries will also be used:

- Pyinvoke[24] - a build system like CMake, providing a uniform interface to repeatable scripted tasks, like downloading Hansard from the TWFY API.
- Pytest[25] - a simple Unit Testing library in Python with minimal boilerplate requirements.
- Keras[26] – library for creating neural networks, used to implement the BLSTM.

6.2 Persisting State

Samtla currently uses an SQL database, purely to store data in the Django model - user registrations with their usernames and passwords, and a log of their activity, used for Samtla's user-specific recommendations of similar documents. All storage of the texts themselves is done using a custom language-model, a space-optimised k-truncated suffix tree, indexed and hosted on Birkbeck's own servers, equipped with Solid State Disks (SSDs) for performance - see (Harris *et al.*, 2014, p. 92ff).

It makes sense to decouple the Named Entity Recognition data from the state persisted by the rest of the system, and also the model's straightforward persistence needs do not justify the overhead of an SQL database. The Python shelve API[27] will be used to create a backing file containing the results of the machine-learning process. A 'shelve' file contains a dictionary where keys are object-names, and values are object states, serialised using Python Pickle.[28]

---

22 https://www.djangoproject.com/
23 https://docs.python.org/3/library/typing.html
24 http://www.pyinvoke.org/
25 https://docs.pytest.org/en/latest/
26https://keras.io/
27 https://docs.python.org/3/library/shelve.html
28 https://docs.python.org/3/library/pickle.html

7 Methodology / work plan

The challenge of this project is that it has several integration elements (between Samtla and an existing NER model implementation), and as such has several packages of work with different dependencies and risks:

ID	Work package	Target date	Risks / mitigations
1	Sourcing and aggregation of data sources for Named Entities of companies, people and places. Preparation of Hansard documents with NEs interpolated into the document using the NE lists aggregated above.	End April 2018	Data may not be available, or may be in unworkable formats. Use other/less data.
2	Perform manual validation of above Hansard documents to complete semi-automatic labelled data. Construction of NER learning model BLSTM; integration with and modification of https://github.com/0xnurl/keras_character_based_ner implementation.	End May 2018	Model may not perform the task
3	Familiarisation with Samtla SLM. Conversion of Hansard into correct format, and loading of Hansard data into Samtla.	Mid June 2018	Samtla SLM API may not be complete. Mitigation is for Dr Martyn Harris to load the corpus into the SML manually.
4	Familiarisation with Samtla back-end (Python Django) and integration of NER processing with departmental server.	End June 2018	Integration may not be possible due to unforeseen dependencies.
5	Familiarisation with Samtla front-end (Javascript jQuery) and integration of NER visualization.	End July 2018	Integration may not be possible due to unforeseen dependencies.
6	Evaluation of Samtla Char-NER system using k-fold cross-validation techniques.	End August 2018	Fall back to simpler evaluation methods; as a last-resort,

			crowd-source evaluation of Hansards enriched with NE data.
7	Finalise write-up into report	Before 17th Sept 2018	

Items 3-5 above also have a dependency on Birkbeck's Samtla system, which is itself evolving. In particular, the SLM learning model will shortly be updatable by an HTTP-based API which is currently under development. As such, the project has some of the dependency-based time constraints of any IT delivery project, and must accordingly consider contingency plans in case of dependencies not being available, or in case of integrations not working due to unforeseen complexity.

8 Bibliography

Cucerzan, S. and Yarowsky, D. (1997) 'Language Independent Named Entity Recognition Combining Morphological and Contextual Evidence', *Emnlp*, pp. 90–99.

Graham, P. (2004) 'Hackers and painters: Big ideas from the computer age', in *http://www.paulgraham.com/spam.html*, pp. 121–129.

Harris, M. *et al.* (2014) 'The anatomy of a search and mining system for digital humanities', *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, (August), pp. 165–168. doi: 10.1109/JCDL.2014.6970163.

Jurafsky, D. and Martin, J. H. (2009) *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Second. Pearson/Prentice Hall.

Klein, D. *et al.* (2003) 'Named entity recognition with character-level models', *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 -*, 4, pp. 180–183. doi: 10.3115/1119176.1119204.

Kuru, O., Arkan Can, O. and Deniz, Y. (2016) 'CharNER : Character-Level Named Entity Recognition', *Coling*, pp. 911–921.

Smarr, J. and Manning, C. D. (2002) 'Classifying Unknown Proper Noun Phrases Without Context'.