EXTRAÇÃO DE DADOS DA WEB POR ROTULAMENTO DE SEQUÊNCIAS

JOÃO MATEUS DE FREITAS VENEROSO

EXTRAÇÃO DE DADOS DA WEB POR ROTULAMENTO DE SEQUÊNCIAS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Berthier Ribeiro-Neto de Araújo

Belo Horizonte

Julho de 2019

JOÃO MATEUS DE FREITAS VENEROSO

WEB DATA EXTRACTION THROUGH NAMED ENTITY LABELING

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Berthier Ribeiro-Neto de Araújo

Belo Horizonte

July 2019

de Freitas Veneroso, João Mateus

D1234p Web data extraction through named entity labeling / João Mateus de Freitas Veneroso. — Belo Horizonte, 2019

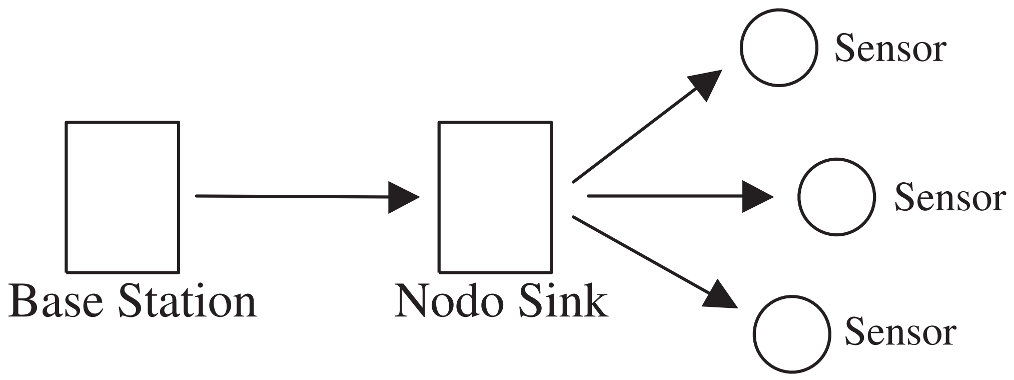
xxiv, 0 f. : il. ; 29cm

Dissertação (mestrado) — Federal University of Minas Gerais

Orientador: Berthier Ribeiro-Neto de Araújo

1. Visão Computacional. 2. Redes. 3. Sabotagens. I. Título.

CDU 519.6*82.10


Base Station
Nodo Sink
Sensor
Sensor
Sensor

Dedicuum cest laborae a quelquis personatum que ajudorat a facirelo.

Acknowledgments

Agradeço aos prótons por serem tão positivos, aos nêutrons pela sua neutralidade e aos elétrons pela sua carga.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum hic tenetur a sapiente delectus, ut aut

reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores repellat.

"Truth and lie are opposite things."

(Unknown)

Resumo

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Palavras-chave: Visão Computacional, Redes, Sabotagens.

Abstract

Web data extraction methods often rely on hand-coded rules to identify and extract data from webpages. These methods are usually suited for extracting information from pages within the same website, however they perform poorly on extraction tasks across different websites. Alternatively, statistical and machine-learning-based sequence labeling methods provide a more flexible approach to Web data extraction. Many times, HTML pages are very different from plain text, because sentences are too short to provide adequate context for conventional Named Entity Recognition methods to work properly. Also, the HTML structure may encode information that is not replicated in the text. Nonetheless, these limitations can be overcome by adequate feature engineering, the use of pretrained word embeddings and neural character representations. In this article, we evaluate the performance of different methods of named entity recognition on the task of Web data extraction. In particular, we introduce a novel dataset[1] consisting of faculty listings from university webpages across the world in multiple languages and test the NER models on the task of extracting researcher names from these listings. We found that a neural network architecture that combines a bidirectional LSTM with a Conditional Random Fields output layer and LSTM-based character representations outperforms other methods on the researcher name extraction task, achieving an F1-score of 0.8867 with no feature engineering. With the addition of hand crafted features, the F1-score can be slightly improved to 0.8995.

Palavras-chave: Named entity recognition, information extraction, web data extraction.

[1]The dataset and all models discussed in this article are available in: https://github.com/jmfveneroso/ner-on-html.

List of Figures

2.1 Example of a faculty directory 10

3.1 Word Frequencies on ConLL-2003 and NER-on-HTML 16

4.1 Graph describing the transitions in a Markov Chain example. 24
4.2 HMM VS MEMM 32
4.3 Graph describing the label bias problem. 33
4.4 RNN for NER 36
4.5 LSTM Cell 37
4.6 Bidirectional LSTM-CRF 38
4.7 CNN based character representations 39
4.8 LSTM based character representations 40

List of Tables

3.1 Description of the data files in the NER-on-HTML dataset 14
3.2 Description of the CoNLL-2003 English dataset 15
3.3 Thirty most frequent words for the ConLL-2003 dataset and the NER-on-HTML dataset 17
3.4 Gazetteer coverage in each data file 17
3.5 Features used in the NER-on-HTML dataset 19

4.1 Model descriptions 41

6.1 Model descriptions 47
6.2 Precision, recall and F1 in the NER on HTML dataset for models that incorporate no features 48
6.3 Precision, recall and F1 in the NER on HTML dataset for models that incorporate all features 49

Contents

Acknowledgments **xi**

Resumo **xv**

Abstract **xvii**

List of Figures **xix**

List of Tables **xxi**

1 Introduction **1**

2 Problem Definition **3**

2.1 Information Extraction in the Web 4

2.2 Named Entity Recognition 7

2.3 Researcher Name Extraction 9

3 Dataset **13**

3.1 Data Description 14

3.2 Differences between NER-on-HTML and ConLL-2003 15

3.3 Dictionary 17

3.4 Features 18

4 Sequence Labeling Methods for NER **21**

4.1 Classical Methods 22

4.1.1 Hidden Markov Models 22

4.1.2 Maximum Entropy Models and Linear Chain Conditional Random Fields 28

4.2 Neural Network Architectures 34

4.2.1 BI-LSTM-CRF 38

4.2.2 CNN character representations 39
4.2.3 LSTM character representations 40
4.2.4 Word Embeddings 40
4.2.5 Technical Concerns 42

5 Sequence Labeling on HTML **43**
5.1 Self training for Hidden Markov Models 43
5.2 Attentions Models 45
5.2.1 Hard Attention 46
5.2.2 Soft Attention 46
5.2.3 Dataset split 46
5.3 F1 Optimization 46

6 Experiments **47**
6.1 Technical specifications 47
6.2 Experiment 1: No features 48
6.3 Experiment 2: All features 49

7 Related Work **51**

8 Conclusion **55**

Bibliography **57**

A HTML sentence segmenter **63**

Chapter 1

Introduction

HTML documents most often lie in between the structured/unstructured data paradigm. DOM hierarchy, element disposition, CSS classes, and other features related to the document structure and indirectly associated with the data itself can be valuable information on the task of identifying entities. Yet, we cannot expect these features to be completely constrained by an underlying pattern. Organization patterns tend to follow some guidelines but they are in no way subject to strict rules. That is why classical web data extraction systems such as automatic wrapper generators **???** do not translate very well across different websites.

Most existing Web data extraction methods are tailored to extract data from a single webpage, producing different compromises between efficacy and degree of human supervision. Some unsupervised approaches proposed to tackle the problem of data extraction for whole application domains **?????**. Usually, unsupervised WDE methods work in two stages. In the record segmentation stage, WDE systems seek to cluster visually and structurally similar webpage regions and identify repeating data records with heuristics and hand-coded rules. In the attribute labeling stage, WDE systems seek to identify the correct attributes on data records, many times resorting to regular expressions or gazetteer matching strategies. The outcome of each of these stages can aid one another. The inner patterns of data records can help in identifying attributes of other data records. Also, by properly identifying data record attributes, it becomes easier to determine boundaries and perform record segmentation.

While unsupervised approaches can be sometimes adequate to extract information from webpages with similar templates, they usually fail on cross-website data extraction tasks. Also, more sophisticated approaches may be needed when we are not dealing with easily distinguishable attributes such as prices and dates. In this regard, machine-learning-based sequence labeling methods can provide a more flexible approach that works regardless of website structure.

In recent years, we saw an amazing progress in the field of Natural Language Processing, particularly with the introduction of deep recurrent neural network architectures for sequence labeling. However, despite Web data extraction being a closely related field, there is a lack of extraction tools that make use of these recent advancements. The attribute labeling stage in Web data extraction systems is essentially a Named Entity Recognition (NER) problem, the problem of detecting named entities in the text and classifying them into predetermined categories such as person names, locations, dates or organizations.

In many cases, such as when we are extracting researcher names from faculty listings, detecting named entities is sufficient to solve the data extraction task. However, when we are dealing with multi-attribute data records or complex relationships, we may need to perform additional steps. Nevertheless, even in the latter cases, flexible NER methods are still desirable because, while many WDE algorithms can effectively exploit the semi-structured nature of Web documents, too much reliance on structural webpage features often produce poor generalizations on cross website extraction. Also, data records may enclose plain text with relevant named entities.

In this article, we investigate methods of named entity recognition for Web data extraction. Recently proposed neural architectures have achieved exciting results on the NER task in plain text while requiring almost no feature engineering or access to gazetteers **???**. NER on HTML poses a slightly different type of challenge. Named entities may occur inside tables, lists, or other types of visual elements that provide little to no textual information that could give hints about the semantic category of a word. Still, even with this limitation, we show that it is possible to obtain very good results with a relatively small training set.

By reliably detecting named entities on HTML, we can improve the performance of existing WDE approaches or even construct an end-to-end neural network architecture to solve domain wide data extraction with considerable flexibility. To test different NER approaches to Web data extraction, we explored the task of researcher name extraction from university faculty listings across the world, introducing a novel NER dataset.

Reliable researcher affiliation information is often missing from public researcher databases (especially in departments other than Computer Science). Also, the display of faculty information varies significantly between different university websites, so this task can provide a good measure of the expected performance and data need of NER methods on other Web data extraction tasks.

Chapter 2

Problem Definition

Web data extraction is a task that is constituted of many overlapping problems that demand integrated and sophisticated solutions. Most often, information extraction tools that work exceptionally well on a given task end up performing poorly when given even a slightly different task. Traditional methods of web data extraction were mainly concerned with the extraction of entities described by a simple ontology from web pages generated by the same source (e.g. collecting house prices and addresses from real estate portals). The extraction of complex entities from a plethora of sources presents a different class of problems. The former case could be solved by rigid tools with hard coded rules that many times yielded close to perfect results. The latter case, however, demands far more flexible approaches since the extraction tools are required to deal with a greater variety of page arrangements and also with entities that are ambiguously defined. In many aspects, the flexible extraction of entities from web pages is more similar to the tasks of named entity recognition and relationship extraction from plain text than to the tasks that concerned the early information extraction methods.

Recent advancements in sequence models for natural language led to important breakthroughs in applications such as speech recognition, machine translation, sentence segmentation, and sequence labeling (which concerns us more directly). In fact, if we treat all sentences in a web page as sequences extracted from an underlying presentation graph (i.e. the DOM tree), the problem of information extraction can be solved through the consecutive application of three NLP tasks: sentence segmentation, named entity recognition, and relationship extraction. First, we need to segment the relevant grouping structures (e.g. sentences, rows in table, items in a list). Then, we must identify relevant named entities (e.g. person names, companies, locations). And finally, we have to discover the relationships between those named entities (e.g. person

X works in company Y). The work flow is the same for plain text and web pages, but there are important differences chiefly related to the structure of the data or the lack of it.

In this dissertation, we investigate the best methods for sequence labeling on HTML. To evaluate these methods, we assess their performance on the task of researcher name extraction from faculty directories of universities across the world. We will be mainly concerned with the task of named entity recognition but a brief discussion of the challenges involved in the task of sentence segmentation for HTML will be provided in Chapter **??**. The task of relationship extraction, however, is not relevant to the problem of researcher name extraction and will not be discussed.

The remainder of this chapter will describe in more depth the subject of this dissertation. Section 2.1 discusses the task of Information Extraction in the Web. Section 2.2 discusses the importance of Named Entity Recognition for Web Data Extraction. Finally, Section 2.3 describes the specific problem of Researcher Name Extraction.

2.1 Information Extraction in the Web

Information Extraction consists of mapping unstructured or poorly structured data to a semantically well defined structure. "(It) is the process of filling the fields and records of a database from unstructured or loosely formatted text" **?**. Usually, the input consists of a corpus containing useful entities that are scattered in the text and the information extraction system is responsible for finding these entities and organizing them according to a rigid hierarchy such as the one defined by the schema of a relational database. It must be stated that it is somewhat misleading to refer to plain text as unstructured data, since prose has a loosely defined structure that ultimately renders it comprehensible. However, in the context of *Information Extraction* we refer to unstructured data in contraposition to tabular data, which are in many cases easier to work with than plain text.

Information Extraction is a multifaceted problem that spans communities of researchers in the fields of *Text Mining*, Information Retrieval, and Natural Language Processing. *Text mining* is the search of patterns in unstructured text that may involve document clustering, document summarization, *Information Extraction*, and other subtasks. *Information Retrieval* is tipically concerned with the parsing, indexing and retrieval of documents and *Information Extraction* can help giving a more precise answer to the user's information needs. Finally, *Natural Language Processing* is a field of Computer Science concerned with how computers process and understand natural language of which two subtasks, namely Named Entity recognition and Relation Extraction, are of special importance for *Information Extraction.*

A popular application of *Information Extraction* is the identification of entities such as people, organizations, or events from news sources and the determination of their relations. For example, one could be interested in determining who is the CEO of a company that was mentioned in the news or which politicians support a bill that is being considered by the Congress. Another interesting news-related application of *Information Extraction* is tracking disease outbreaks through the extraction of disease names and locations from news sources and determining their relation to outline the geographical area affected by an epidemy (cite example). Other *Information Extraction* systems proposed to track disease outbreaks through the analysis of tweets (cite example), which presents a remarkably different challenge since the context of tweets is very limited and the presence of abbreviations and typos is more widespread than in the case of highly curated content such as news.

The field of bio-informatics has also found important applications for *Information Extraction.* The first BioCreAtIvE challenge dealt with extraction of gene or protein names from text, and their mapping into standardized gene identifiers for three model organism databases (fly, mouse, yeast). **?**. In the BioCreative/OHNLP Challenge 2018 **?**, researchers were required to investigate methods of family history information extraction. Family history is a critical piece of information in the decision process for diagnosis and treatment of diseases, however the main sources of data are unstructured electronic health records. The task is divided in two subtasks: 1) Entity recognition (family members and disease names); and 2) Relation Extraction (family members and corresponding observations).

Another scientific application for *Information Extraction* methods is the the extraction of information from research papers to populate citation databases and bibliography search engines such as Citeseer **?**, DBLP [1], Semantic Scholar [2], and Google Scholar [3]. The vast amount of scientific knowledge produced daily demands automatic methods for extracting authors, titles, affiliations, references, venues and the year of publication from research papers and online resources such as university websites, individual researcher home pages and conference websites. This application is especially important to the evaluation and bibliometrics research communities, that are concerned with the measure of academic impact and researcher productivity through the usage of quantitative indices of academic impact such as the H-index **?** and P-score **?**.

[1]http://dblp.uni-trier.de/

[2]https://www.semanticscholar.org

[3]https://scholar.google.com/

Web Data Extraction works a little different from *Information Extraction* in plain text because HTML documents frequently lie in between the structured/unstructured data paradigm. Relevant entities may occur inside tables, lists, or other types of visual elements that provide little to no contextual information that could give hints about their category. Web pages have a two dimensional tabular structure that is usually more similar to a spreadsheet than to text found in news corpora. For this reason, features extracted from the DOM hierarchy such as element disposition, CSS classes, and nesting structure can provide valuable information in identifying entities and extracting their attributes. Yet, we cannot expect these features to be completely constrained by an underlying pattern. Organization patterns tend to follow some guidelines but they are in no way subject to strict rules.

Most existing Web data extraction methods are tailored to extract data from a single webpage, producing different compromises between efficacy and degree of human supervision. Usually, these methods work in two steps. In the record segmentation step, we seek to cluster visually and structurally similar webpage regions and identify repeating data records. In the attribute labeling stage, we seek to identify the correct attributes for each data record, maybe resorting to regular expressions or simple dictionary matching strategies depending on the task at hand. The outcome of each step can aid one another. The inner patterns of data records can help identifying attributes of other data records. Also, by properly identifying attributes, it becomes easier to determine boundaries and perform record segmentation correctly. Figure **??** shows how a hypothetical **??** system that aims to collect book titles and prices from Amazon could perform record segmentation and attribute labeling.

The task of extracting product names and prices from online catalogs can likely be solved by a rather inflexible system that operates mainly with hard coded rules and simple regular expressions, especially if we only consider pages with a very similar template (e.g. Amazon product listings). Classical wrapper generators **???** are well suited for this type of task. However, when we need to identify more complex and ambiguous entities such as researcher names in many different contexts such as faculty websites, we might be better off with a more flexible approach. The task of Named Entity Recognition aims to identify named entities (e.g. people, organizations, etc.) in plain text, but we will see in the next chapters that the sequence models that work well in plain text can also be employed succesfully in web extraction tasks, sometimes with a few alterations.

This section gives a brief introduction to *Information Extraction*, but there are many applications that were not discussed here. A more detailed view of the field is given in Sarawagi **?**.

2.2 Named Entity Recognition

Named Entity Recognition is a task in Natural Language Processing that aims to identify *Named Entities* in a text. The Named Entity Recognition task first appeared as a subtask of *Information Extraction* in the context of the Message Understanding Conference (MUC) promoted by the american Naval Ocean Systems Center. In MUC-3, a precursor task involved extracting information on terrorist incidents (incident type, date, location, perpetrator, target, instrument, outcome, etc.) from messages disseminated by the Foreign Broadcast Information Service of the U.S. Government **?**. In MUC-6 **?** the "named entity" task was created with the goal of identifying the names of people, organizations, and geographic locations from articles of the Wall Street Journal. In MUC-7, the task was expanded to handle multilingual evaluation and "Named Entities (NE) were defined as proper names and quantities of interest. Person, organization, and location names were marked as well as dates, times, percentages, and monetary amounts" **?**. The shared-task at the Conference on Computational Natural Language Learning in 2003 **?** concerned language-independent named entity recognition and it is especially important because it established an enduring data format for NER and it introduced a dataset of articles extracted from news sources that is to this day still employed to evaluate the quality of NER systems.

The task of *Named Entity Recognition* is essentialy a sequence labeling task. That is, given a sequence of tokens we want to attribute labels to each token classifying them in one of a number of predefined classes. Figure **??** describes the process of atributing Named Entity labels to a sentence according to the CONLL format. There are cases where a token may belong to more than one class because of nested named entities (e.g. in "CoNLL 2003", 2003 is both part of a conference name and a date). This complexity is of importance to some applications **?** but it will not be discussed here because nested named entities are absent from our concrete problem of researcher name extraction and most sequence models can be adapted to handle this special case with a few modifications.

A simple approach to *Named Entity Recognition* can be devised through the use of regular expressions, that are search patterns that describe a regular language and which can be easily implemented in most programming languages. These programs can be perfectly suited to extract regular entities such as dates and prices with almost perfect accuracy. The extraction of other types of entities that belong to a limited set such as the names of states in a country can be accomplished through dictionary matching. Other types of entities such as the ones investigated in the CoNLL-2003 challenge require more sophisticated methods of sequence labeling.

Some classical statistical methods of *Sequence Labeling* that are able to handle the labeling of complex entities with decent accuracy are Hidden Markov Models, Linear Chain Conditional Random Fields and Maximum Entropy Markov Models. The first two will be explored in further detail in Chapter **??**, and the latter is essentialy a more restrictive discriminative model in comparison to Condition Random Fields. These statistical approaches have had remarkable resiliency in *Sequence Labeling* tasks and still provide a good first approximation to a solution because of their simplicity, speed and accuracy.

The deep learning revolution has brought huge advancements to *Named Entity Recognition*, as was the case with most NLP tasks. Deep learning differs from classical machine learning in regard to the levels of abstraction learned by the classifiers. Deep learning techniques combine feature extraction and classification in a single system. While a conventional feed-forward neural network may perform classification by learning the weights of a single hidden layer through backpropagation, a deep learning model is usually composed of multiple hidden layers that handle different levels of abstractions. In text related tasks, the first level of abstraction could consist of a word embedding layer, where words are mapped to a continuous vectorial space with reduced dimensionsality, and the next level of abstraction could be composed of a multi layered recurrent neural network.

Essentialy all the best scoring models at the Conll-2003 task employ some form of deep neural network, and most often Long Short Term Memory recurrent neural networks with a range of differing characteristics. When combined with pretrained word embeddings these models provide a powerful method of sequence labeling that requires practically no feature engineering or dictionary matching. However, these models have become quite complex, constrasting with earlier approaches such as Hidden Markov Models, that only require estimation of a comparatevely small number of weights in closed form. The training time required by such deep models is many orders of magnitude larger than that of classical approaches, and if we take into consideration the training of word embeddings such as Word2Vec's skip gram model in a billion token corpus, it is no exaggeration to say that deep learning models increase the training time a thousandfold when compared to earlier approaches. Additionaly, most deep learning models require expensive hardware in the form of GPUs or TPUs [4] to become practical options.

Despite some noteworthy efforts on data visualization references here, DNNs still lack interpretability. Also, they have limited applicability when we care about causal

[4]Those are Google's Tensor Processing Units "https://cloud.google.com/tpu/"

inference references here, a goal that is especially meaningful in applied statistics and the social sciences. Regardless, it must be acknowledged that deep learning is a remarkably useful tool to build classifiers with unparallaled accuracy in *Named Entity Recognition* as well as in other tasks. We must only be aware of its shortcomings.

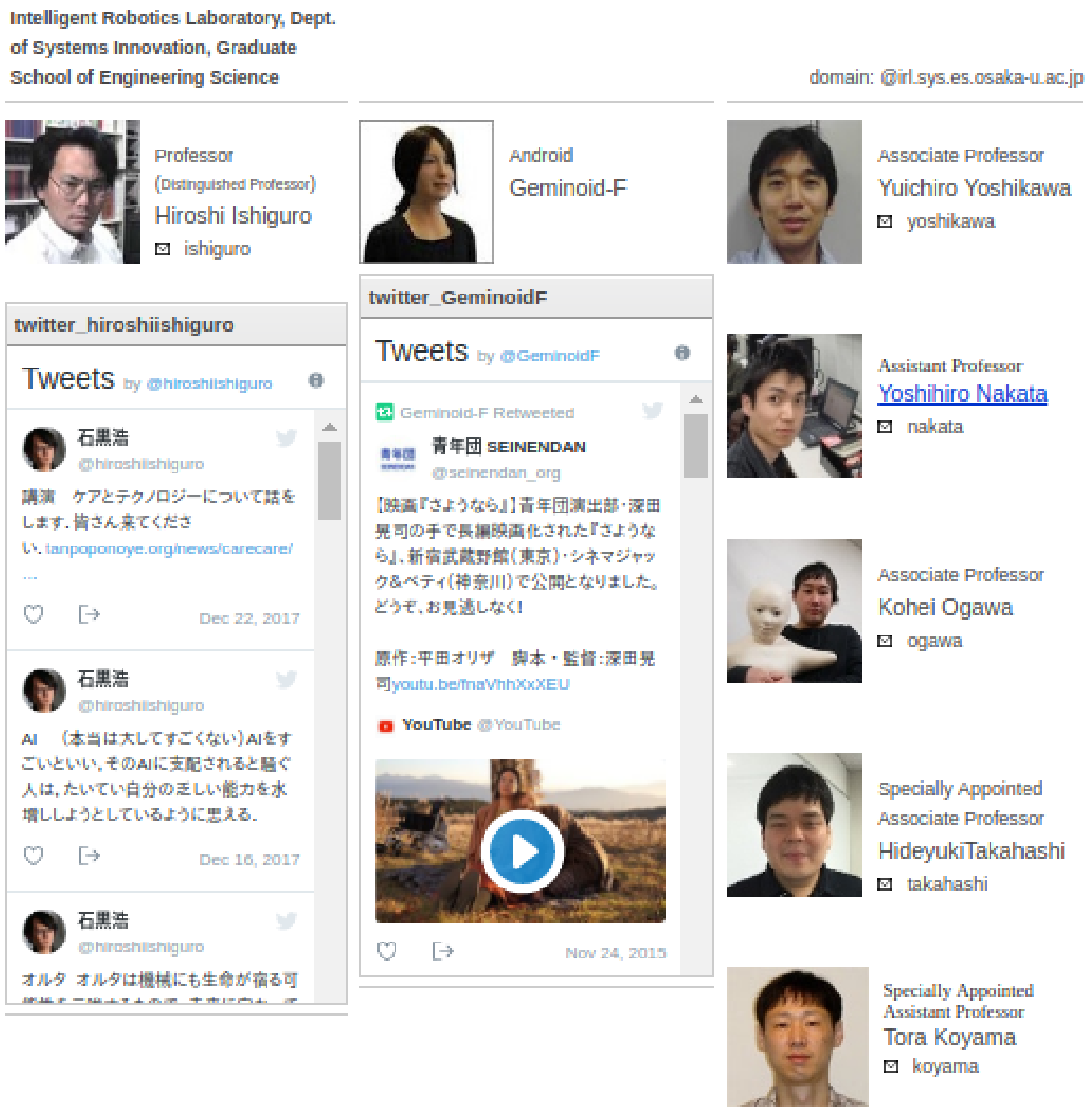
2.3 Researcher Name Extraction

This dissertation has the goal of finding the best approaches to sequence modelling for Web Data Extraction. To achieve this goal, we compare multiple systems at the researcher name extraction task. The task consists of extracting researchers names from university faculty listings across the world to discover their affiliations and link their profiles to public databases such as DBLP or Google Scholar [5]. Public researcher databases only have sparse information about author affiliation and even in fields for which the information is more easily available, such as Computer Science, only a small fraction of the records have reliable affiliation information.

To acknowledge the complexity of this task, take for example a snippet extracted from the staff page for the intelligent robotics laboratory from Osaka University shown in Figure 2.1. There is some structure to the way member profiles are arranged, but the organization is rather flexible. Other pages, even from the same website, can show very different patterns, ranging from tables and lists to free form. Researcher names can appear inside plain text, similar to the case of *Named Entity Recognition* from news sources or in a more tabular structure. Names may also be part of larger sentences such as in "Michael Johnson Chair" and "John Doe Avenue" yielding false positives. Or they can be composed of common words (e.g. Summer Hall) yielding false negatives. There is no rule that fits all cases.

State-of-the-art named entity recognition models trained on news datasets (cite here) do not perform so well at this task, because in many webpages, text alone is insufficient to provide proper contextual information about the semantic category of a word. The absence of context demands extraction systems to rely on information from sources outside the dataset, being them features extracted from unlabeled corpora obtained through unsupervised pretraining, or dictionaries containing many instances of the relevant entities. Although, entity instances will rather frequently be unique occurences, rendering dictionary matching approaches insufficient. Therefore, a key problem in the task of entity name extraction is accounting for all possible name combinations, even those that seem unlikely. Since there is no database with all possible

[5] This is useful, for instance, if one needs to compare the research output of deparments in Germany, or study international publication patterns.



Intelligent Robotics Laboratory, Dept. of Systems Innovation, Graduate School of Engineering Science

domain: @irl.sys.es.osaka-u.ac.jp

Professor (Distinguished Professor)
Hiroshi Ishiguro
ishiguro

Android
Geminoid-F

Associate Professor
Yuichiro Yoshikawa
yoshikawa

twitter_hiroshiishiguro

Tweets by @hiroshiishiguro

石黒浩
@hiroshiishiguro
講演　ケアとテクノロジーについて話をします. 皆さん来てください. tanpoponoye.org/news/carecare/ …
Dec 22, 2017

石黒浩
@hiroshiishiguro
AI　（本当は大してすごくない）AIをすごいといい, そのAIに支配されると騒ぐ人は, たいてい自分の乏しい能力を水増ししようとしているように思える.
Dec 16, 2017

石黒浩
@hiroshiishiguro
オルタ　オルタは機械にも生命が宿る可

twitter_GeminoidF

Tweets by @GeminoidF

Geminoid-F Retweeted
青年団 SEINENDAN
@seinendan_org
【映画『さようなら』】青年団演出部・深田晃司の手で長編映画化された『さようなら』, 新宿武蔵野館（東京）・シネマジャック&ベティ（神奈川）で公開となりました。どうぞ, お見逃しなく!

原作：平田オリザ　脚本・監督：深田晃司youtu.be/fnaVhhXxXEU

YouTube @YouTube
Nov 24, 2015

Assistant Professor
Yoshihiro Nakata
nakata

Associate Professor
Kohei Ogawa
ogawa

Specially Appointed Associate Professor
HideyukiTakahashi
takahashi

Specially Appointed Assistant Professor
Tora Koyama
koyama

Figure 2.1. Example of a faculty directory

named entity combinations, we need holistic statistical methods that can handle unknown tokens with relative efficacy.

Considering that NER datasets built from news corpora such as CoNLL-2003 do not provide an adequate measure for the performance of sequence models on HTML, we introduce the NER-on-HTML dataset to evaluate the performance of different sequence models at the researcher name extraction task. This dataset is described in Chapter 3.

Another important related task in the context of researcher name extraction is performing named entity disambiguation, which consists of linking the extracted named entities to a unique profile in a unified database. This task is called *Named Entity Linking*. Roughly half the records found in faculty webpages can be linked to their records in public databases without great difficulty. For the other half we may need to employ more complex systems or perform manual classification. This task however is not the object of our study.

Chapter 3

Dataset

This chapter describes the dataset used to evaluate the performance of sequence models on the Web data extraction task. We call it the NER-on-HTML dataset. The task consists of finding researcher names in faculty listings from university webpages across the world, mainly from Computer Science departments. This would be a necessary step when linking researcher profiles from university websites to their entries in public databases such as DBLP [1]. Unlike many information extraction datasets, each webpage in the dataset comes from a different website, and therefore has a different format, what makes many information extraction approaches impractical. The idea is to explore systems that are general enough to allow efficient entity extraction from different sources while requiring no supervision between different websites.

This task is similar to labeling authors in comments or articles collected from different publishing platforms. Another similar task is NER on tweets. Because of the character limitation, tweets rarely provide sufficient context for NER systems to perform token labeling adequately.

We collected 145 computer science faculty pages from 42 different countries in multiple languages, although the English version was preferred when it was available. We gathered faculty webpages randomly in proportion to the number of universities in each country[2]. Each HTML page was preprocessed and converted to the CoNLL 2003 data format. That is, one word per line with empty lines representing sentence boundaries. Sentence boundaries were determined by line break HTML tags (div, p, table, li, br, etc.) in contrast to inline tags (span, em, a, td, etc.). Sentences that were more than fifty tokens long were also split according to the punctuation. The algorithm used for sentence segmentation is described in detail in Appendix **??**.

[1]http://dblp.uni-trier.de/

[2]A detailed list of universities can be found in https://univ.cc/world.php

A proper HTML segmenter poses many challenges by itself. We wanted to evaluate models without relying on any sophisticated data record segmentation system. In many cases, entity annotation may precede the segmentation phase on Web Data Extraction methods. Also, depending on the task (as is the case for researcher name extraction), a good annotator that is able to work with raw HTML allows for a more flexible system and can provide a sufficient solution.

Finally, all tokens were tagged using the IOB scheme put forward by Ramshaw and Marcus **?** that is described in Figure **??**. The performance of sequence models in the NER-on-HTML dataset was evaluated according to the precision, recall and F-measures **?** in the test set. Precision accounts for the percentage of named entities found by the model that are correct, this is the proportion of true positives relative to all extracted entities. Recall is the percentage of named entities that are present in the corpus and were found by the model, this is the proportion of true positives relative to all named entities that should have been extracted. The relation between both of these measures is expressed in Figure **??**. The F-measure score is a composite measure that combines precision and recall with the formula:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{3.1}$$

The choice for β depends on the importance attributed to the precision and recall measures. This formula "measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision" **?**. A common choice for the value of β is 1, this measure is called the F_1-score. That is, we attribute as much importance to recall as to precision.

3.1 Data Description

Data file	Documents	Sentences	Tokens	Names
Training	85	24728	110269	5822
Validation	30	8743	36757	1788
Test	30	10399	44795	2708
Total	145	43870	151821	10318

Table 3.1. Description of the data files in the NER-on-HTML dataset

The NER-on-HTML dataset is comparable in size to other popular NER datasets. It was divided in training, validation and test sets, which are described in Table 3.1. The validation set was used in the early stopping validation strategy to avoid overfitting classifiers, and model performance was evaluated by comparing the performance in the test set.

Most web pages in this dataset are faculty directories with informative text from small passages, eventhough prose is not absent from more than a few webpages. Size and structure varies wildly, therefore some documents may contain up to a few hundred names whereas other documents may contain only twenty or thirty names. This difference in document size may be problematic when comparing different extraction systems, because a system that performs well on pages with large content may perform poorly in a set of smaller pages, but this characteristic would not affect so much the averaged results. A different approach might be to consider the averaged precision, recall and F-measures per page, privileging systems that have a more regular performance between websites. However, this approach might cover other deficiencies, since a system that produced 100 false positives in a page with 500 names would account for the same level of drop in average recall as a system that missed 10 false positives in a page with 50 names. For that reason, we chose to keep the same criteria of evaluation used on other *Named Entity Recognition* tasks, that is, calculating global precision, recall, and F-measures relative to the total amount of named entties in the corpus.

3.2 Differences between NER-on-HTML and ConLL-2003

Data file	Documents	Sentences	Tokens	LOC	MISC	ORG	PER
Training	946	14987	203621	7140	3438	6321	6600
Validation	216	3466	51362	1837	922	1341	1842
Test	231	3684	46435	1668	702	1661	1617
Total	1393	22137	301418	10645	5062	9323	10059

Table 3.2. Description of the CoNLL-2003 English dataset

The CoNLL-2003 dataset was introduced in the *Named Entity Recognition* shared-task at the Conference on Computational Natural Language Learning and is still one of the most popular NER datasets, being used to attest the performance of state-of-the-art sequence labeling systems. The English data is composed of news stories extracted from the Reuters Corpus, and provides annotations for four types of entities: people (PER), organizations (ORG), locations (LOC), and miscellaneous (MISC), which includes entities that cannot be classified in one of the former groups. In this section, we present a brief comparison between the ConLL-2003 English dataset and the NER-on-HTML dataset to understand how documents from both sources differ, and why this difference may be responsible for the observed drop of performance when we apply NER systems trained in news corpora to HTML extraction tasks.

The characteristics of the ConLL-2003 dataset are described in Table 3.3. The number of documents in the NER-on-HTML dataset is much smaller: only 145 against 1393. However, the number of sentences in the NER-on-HTML dataset is higher: 43870 against 22137. Also, the number of tokens in the NER-on-HTML is roughly half the number present in the ConLL-2003 dataset. These numbers show that the HTML documents relevant to the researcher name extraction task are longer than news stories and, more importantly, they are composed of much shorter sentences when compared to the text from news corpora (3.46 words per sentence in NER-on-HTML versus 13.62 words per sentence in ConLL-2003). This attests to the fact that HTML provides far less context to be made use by sequence models than plain text. This means that the sequence models must make use of information other than textual context to perform well.

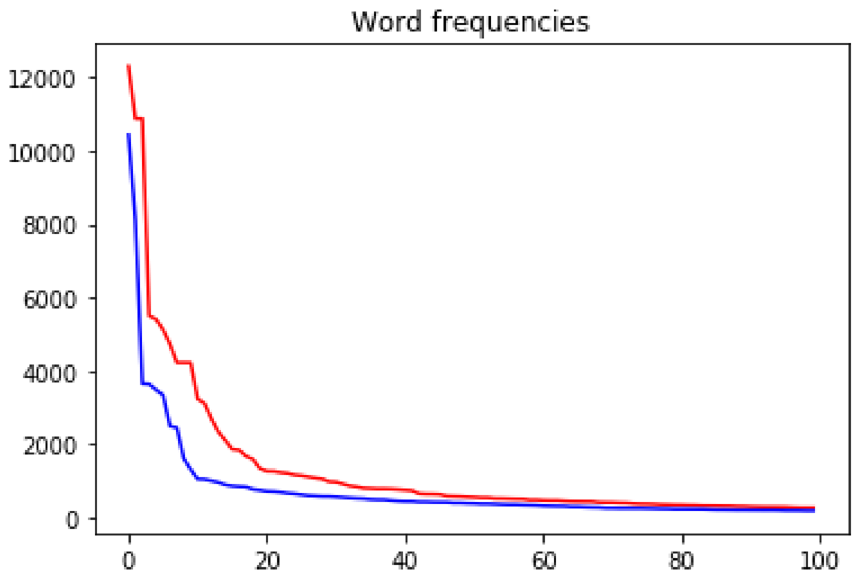


Figure 3.1. Word Frequencies on ConLL-2003 and NER-on-HTML

The word frequency distributions for the hundred most frequent words in both

Word	Frequency	Word	Frequency
the	12310	,	10439
,	10876	-	8140
.	10874	)	3655
of	5502	(	3641
in	5405	:	3484
to	5129	of	3345
a	4731	and	2499
(	4226	professor	2456
)	4225	university	1611
and	4223	research	1315

Table 3.3. Thirty most frequent words for the ConLL-2003 dataset and the NER-on-HTML dataset

datasets are described in Figure 3.1. Their shape is rather similar, presenting a small number of very frequent words and a long tail of words that happen only once or twice in the dataset. This long tail is where most named entities are located. To allow further comparison between both datasets, we present the ten most frequent words (including punctuation signs) for each dataset in Table **??**. As expected, the ConLL-2003 contains more generic terms whereas the subject of the NER-on-HTML becomes evident with words such as "professor" and "university" happening with a high frequency in the corpus. Also, punctuation signs are relatively more frequent in the NER-on-HTML owing to the factual tabular structure of the HTML texts present in the collection.

3.3 Dictionary

Data file	Precision	Recall	F1	Correct names
Training	0.7316	0.2303	0.3504	1341 of 5822
Validation	0.8474	0.2858	0.4274	511 of 1788
Test	0.8717	0.3287	0.4773	890 of 2708

Table 3.4. Gazetteer coverage in each data file

A dictionary or gazetteer can be a powerful aid on generic sequence labeling systems, eventhough deep learning architectures can be much less reliant on these types of features, especially when we make use of pretrained word embeddings. We extracted a list of 1,595,771 researcher names from the DBLP database and annotated tokens in the NER-on-HTML dataset with exact and partial matches tags. That is, if a sequence of tokens corresponded exactly to a name from the DBLP list, the entire sequence was annotated as an exact match. Otherwise, if some tokens in the sequence matched a name from the DBLP list partially, the matching tokens were annotated with a partial match tag. Table 3.4 shows the precision, recall and F1 scores obtained with an exact gazetteer matching strategy in each NER-on-HTML data file as a baseline. This is the performance obtained by only extracting sequences of tokens that correspond to an exact match with the DBLP list. As expected, recall is very low and precision falls short of top performance extraction methods. Having established this baseline, any useful extraction method should easily beat the dictionary exact matching approach.

3.4 Features

Finally, thirteen categorical features were associated with each token in the dataset. These were selected through feature engineering from a larger pool of features, considering their aid to the performance of the extraction systems. Deep learning architectures can work incredibly well without any of these features, however these features are of critical importance to classical approaches such as HMMs and CRFs. The selected features are presented in Table 3.5. Additionaly, all models also make use of the unmodified tokens as features directly or as embedding lookups.

Feature 1 represents the token converted to unaccented lowercase format. Feature 2 represents an exact match of a sequence of tokens to any of the 1,595,771 names from the DBLP list, and feature 3 represents a partial match. Feature 4 is the rounded logarithm of the frequency of a token in the gazetteer, and feature 5 is the rounded logarithm of the frequency of a token in a word corpus obtained through a random crawl on university websites. Features 6 to 11 represent a simple regular expression match to an email, number, honorific, URL, capitalization or punctuation sign. Feature 12 represents the HTML enclosing tag and its parent concatenated. Finally, feature 13 represents all CSS classes concatenated. Features 12 and 13 are of importance to the self-training strategy of HMMs described in Section ?? and the attention architecture for DNNs described in Section ??.

Feature	Description
1	Unaccented lowercase token
2	Exact gazetteer match
3	Partial gazetteer match
4	Log name gazetteer count
5	Log word gazetteer count
6	Email
7	Number
8	Honorific (Mr., Mrs., Dr., etc.)
9	URL
10	Is capitalized
11	Is a punctuation sign
12	HTML tag + parent
13	CSS class

Table 3.5. Features used in the NER-on-HTML dataset

Chapter 4

Sequence Labeling Methods for NER

Many important tasks in Natural Language Processing ranging from Part-of-speech tagging to Named Entity Recognition involve attributing labels to sequences of words. In the *Named Entity Recognition* task, we want to attribute labels to tokens, classifying them into one of many predefined classes according to the available evidence. In Part-of-speech tagging we do essentialy the same task with different classes. Both of these tasks, as any other sequence labeling task can be modeled generically. If $X = \{x_1, x_2, ..., x_n\}$ is a sequence of words and $Y = \{y_1, y_2, ..., y_n\}$ is a sequence of labels attributed to the word sequence X. Our goal is to find the optimal sequence of labels $Y*$ that maximizes the conditional probability of Y given X. That is:

THIS DEMANDS BETTER EXPLANATION. THIS IS ACTUALLY THE MAXIMUM LIKELIHOOD ESTIMATE.

$$Y^* = \arg\max_Y P(Y|X) \tag{4.1}$$

HOW EXACTLY DO WE ESTIMATE THESE FOR NEURAL NETWORKS?

MAXIMUM ENTROPY

Calculating these probabilities precisely is usually impractical due to the exponential increase in the number of label combinations as we increase the sequence size n. As n becomes large, label combinations will become increasingly uncommon in the dataset, what will make probability estimation less reliable. Taking this into consideration, ultimately, the difference between sequence labelling models lies in the assumptions behind the estimation of the conditional probabilities $P(Y|X)$. In this chapter we will describe several machine learning approches to sequence labeling, starting with the classical methods in Section **??**, which are: Hidden Markov Models, Conditional Random Fields and Maximum Entropy Markov Models. In Section **??** we will describe some neural architectures for sequence labelling, which are: bidirectional recurrent neural networks, LSTM-CRF, CNN character representations, LSTM character representations. Finally, in Section **??** we will describe how word embeddings can help improve the performance of some of the previously introduced sequence labeling methods on the Named Entity Recognition for the web task.

4.1 Classical Methods

4.1.1 Hidden Markov Models

The *Hidden Markov Model* is an expanded *Markov Chain* where the sequence of observed states depends on an underlying sequence of hidden states. A *Markov Chain* is a stochastic model for describing sequences, when the described sequences have the *Markov Property.* That is, consider a sequence $X = \{x_1, x_2, \dots, x_n\}$ in which each observed state $x_i, \forall i \in [1, n]$ takes its value from a set of m possible states $\{S_1, S_2, \dots, S_m\}$. Sequence X only satisfies the *Markov Property* if:

$$P(x_i|x_{i-1}, x_{i-2}, \dots, x_1) = P(x_i|x_{i-1}, x_{i-2}, \dots, x_{i-k}) \tag{4.2}$$

The probability of observing state x_i at time i depends only on the k previous observed states. When $k = 1, k = 2, \dots$ the *Markov Chain* is said to be of first order, second order, and so on. What the *Markov Assumption* means is that, at any time, the entire history of observed states in the sequence is encoded in a limited number of previous states. So, by knowing a limited number of previous states, we can make as accurate predictions about the future as we would make by knowing all the sequence history. Conditioned on this limited set of previous states, the future and past states are independent.

Because of our present interest in describing sequence labeling models, we only consider *Discrete-Time Markov Chains*, when, in fact, there are also *Continuous-Time Markov Chains* [1]. These models are hardly useful on text modelling tasks because words and characters in a text are always countable and therefore discrete. We also consider *Markov Chains* to be of first order (i.e. $k = 1$) in the following derivations, without loss of generality, to simplify the notation.

[1]http://www.columbia.edu/ ks20/stochastic-I/stochastic-I-CTMC.pdf

A *Markov Chain* can represent a wide range of phenomena such as daily temperatures in a region, closing prices of a stock in the financial market, yearly demographic growth, or words in a text. Nevertheless, it represents a simplification of reality that makes sequences mathematically treatable. The extent to which this simplification will hurt our predictions depends on the nature of the studied phenomena.

To see why the *Markov Assumption* may introduce a problem on text related tasks, consider the following sentence: *"The wife had discovered that the husband was carrying on an intrigue with a French girl, who had been a governess in their family"*. To understand which entities are related to the passages *"governess"* and *"in their family"*, we need to keep track of distant relations in the text that would not be captured by a *Markov Model* of second or third order. We could consider employing a *Markov Model* of higher order to overcome this limitation, however, considering more than three or four states at any time makes parameter estimation unreliable because each combination of states becomes increasingly more uncommon in the training set. Also, when the considered number of states increases, the cost for predicting the optimal sequence of states quickly becomes prohibitively high. There are some *Markov Chain* variations that try to address this problem such as (CITE time-homogeneous MCs, MC with memory), as well as the other models presented later in this chapter. Nevertheless, this limitation is of lesser significance in the context of *Named Entity Recognition*, because we can complement the model with features that make up for the lack of knowledge that would be provided by a longer context.

We still have made no restrictions on the form taken by the transition probabilities $P(x_i = S_b|x_{i-1} = S_a)$ besides the *Markov Assumption.* It is usual to consider these probabilities to be time invariant.

$$P(x_{i+1} = S_b|x_i = S_a) = P(x_i = S_b|x_{i-1} = S_a) \tag{4.3}$$

That is, given states S_a and S_b, the probability of going from S_a to S_b on timestep i is the same as in any other timestep. The graph in Figure 4.3 describes the transitions for a *Time Invariant Markov Chain* example with only two states (Name, Word), that models a sequence of words that can be either names or common words. The edges in this graph represent the transition probabilities between states.

With the *Time Invariance Assumption*, all parameters in our model can be described by a $m \times m$ transition matrix θ where each element $\theta_{a,b}$ with row a and column b holds the probability of going from state S_a to S_b. Naturally, the probabilities of row $\theta_{a,*}$ must sum up to one since they represent the entire scope of transition possibilities starting from state S_a. Equation 4.4 describes a 2×2 transition matrix Z for the same

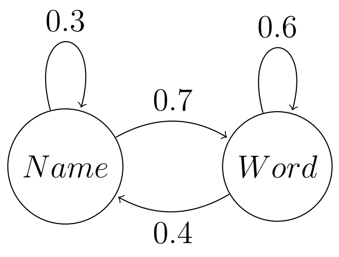


Figure 4.1. Graph describing the transitions in a Markov Chain example.

Markov Chain example described in Figure 4.3.

$$Z = \begin{bmatrix} 0.3 & 0.7 \\ 0.4 & 0.6 \end{bmatrix} \tag{4.4}$$

If we know the transition matrix θ for a *Markov Chain*, we can easily calculate the probabilities of being in each state at time t. Consider that ρ_t is a vector of size m (i.e. the number of possible states) with the probabilities for each state at time t. Then:

$$\rho_{t+1} = \rho_t \cdot \theta \tag{4.5}$$

And, for an arbitrary number of timesteps $t \geq 0$:

$$\rho_t = \rho_0 \cdot \theta^t \tag{4.6}$$

Where ρ_0 is the vector of starting probabilities for each state. If we know nothing about the initial conditions of our *Markov Chain* we can simply assign the same probability to each state.

The probability of observing a sequence of states $X = \{x_1, x_2, ..., x_n\}$ with an initial state x_0 is:

$$P(X) = P(x_0) \prod_{t=1}^{n} P(x_t | x_{t-1}) \tag{4.7}$$

Now, to obtain the likelihood $\mathscr{L}(\mathscr{X}; \theta)$ we need only replace the probabilities with the respective parameters in our model:

$$\mathscr{L}(X; \theta) = \rho_0 \prod_{t=1}^{n} \theta_{x_t, x_{t-1}} \tag{4.8}$$

By defining $\eta_{a,b}$ to be the count of transitions from a to b in X. We can write the likelihood as:

$$\mathscr{L}(X;\theta) = \rho_0 \prod_{a,b} \theta_{a,b}^{\eta_{a,b}} \tag{4.9}$$

Next we want to obtain the maximum likelihood estimator for the transition matrix $\hat{\theta}$. The likelihood function is always positive, so finding the set of parameters that maximizes the log likelihood $\ell(X;\theta)$ is equivalent to finding the parameters that maximize the likelihood itself. With the log likelihood, we transform the product over transition probabilities into a sum of logarithms, which is easier to work with.

$$\ell(X;\theta) = log(\rho_0) \sum_{a,b} \eta_{a,b} \cdot log(\theta_{a,b}) \tag{4.10}$$

This way, we want to find $\hat{\theta}$ such that:

$$\hat{\theta} = \arg\max_{\theta} \ell(X;\theta) \tag{4.11}$$

However, if we try to optimize this function as it is, we will find that $\hat{\theta}_{a,b} = \infty, \forall a, b$. This happens because every row in the transition matrix must sum up to one so the degrees of freedom for the model are actually $m(m-1)$ and not m^2. This means that:

$$\sum_{b} \theta_{a,b} = 1, \forall a \in [1, m] \tag{4.12}$$

A way to incorporate this constraint in our optimization problem is with the aid of *Lagrange Multipliers*. So we define the *Lagrangian*:

$$Lag(\theta, \lambda) = \ell(X;\theta) - \sum_{a} \lambda_a \cdot \left(\sum_{b} \theta_{a,b} - 1 \right) \tag{4.13}$$

Taking the derivatives of $Lag(\theta, \lambda)$ relative to $\theta_{a,b}$ and making them equal to zero we get:

$$\frac{\partial Lag(\theta, \lambda)}{\partial \theta_{a,b}} = \frac{\eta_{a,b}}{\theta_{a,b}} - \lambda_a = 0 \tag{4.14}$$

$$\theta_{a,b} = \frac{\eta_{a,b}}{\lambda_a} \tag{4.15}$$

$$\tag{4.16}$$

And finally, by using the constrain equations $\sum_b \theta_{a,b} = 1$, we find that:

$$\sum_b \frac{\eta_{a,b}}{\lambda_a} = 1 \tag{4.17}$$

$$\lambda_a = \sum_b \eta_{a,b} \tag{4.18}$$

$$\theta_{a,b} \quad = \quad \frac{\eta_{a,b}}{\sum_{b'} \eta_{a,b'}} \tag{4.19}$$

$$\tag{4.20}$$

Yielding a closed form expression for the maximum likelihood estimator $\hat{\theta}_{a,b}$. What is really convenient, since this is the average number of transitions from state a to state b in sequence X. A value that can be easily obtained in $O(n)$ time complexity.

So far, we have assumed that all states X are observable, but in *NER*, only the words are observed, while the *Named Entity* labels associated with these words are not. That is why we need another layer of complexity. The *Hidden Markov Model (HMM)* differs from the *Markov Chain* in that it does not observe the states X directly, but rather a probabilistic function of these states.

With the *Hidden Markov Model* we want to predict a sequence of labels $Y = \{y_1, y_2, ..., y_n\}$ (i.e. the *Markov Chain*) from a sequence of observed states $X = \{x_1, x_2, ..., x_n\}$ (i.e. the words in a text). So we make an additional assumption:

$$P(x_i|y_{i-1}x_{i-1}, ..., y_1x_1) = P(x_i|y_i) \tag{4.21}$$

That is, the probability of observing word x_i depends only on the current label y_i, the hidden state (e.g. PER, LOC, ORG, etc.). By making this assumption, we are stating that if we know a token's assigned label, then we can reliably predict what are the probabilities that this token takes any value in a vocabulary. This is a very simplistic assumption and many models (such as Conditional Random Fields) try to overcome this issue, but this step is necessary if we want to get a closed form estimator for our model. To model the emission probability distributions $P(x_i|y_i)$, we assume again that the probabilities are time invariant and that x_i takes its value from a fixed vocabulary with size V. Similar to the *Transition Matrix*, we introduce a $V \times K$ *Emission Matrix* μ where each cell $\mu_{a,b}$ represents the probability that, given label a, we will observe the word b.

Now, the probability of observing a vector of labels Y given a sequence of words

X can be calculated with Bayes' theorem:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{4.22}$$

Since $P(X)$ is invariant for every sequence of labels Y, we can simply optimize in terms of the joint probability $P(X, Y)$:

$$P(Y|X) \propto P(X|Y)P(Y) = P(X, Y) \tag{4.23}$$

And we get:

$$\mathscr{L}(Y|X;\theta) \propto \rho_0 \prod_{a\in L, b\in L} \theta_{a,b}^{\eta_{a,b}} \prod_{c\in L, d\in V} \mu_{c,d}^{\eta_{c,d}} \tag{4.24}$$

The parameters θ and μ are independent, so the procedure to find $\hat{\theta}$ remains the same. Also, we can employ the same procedure we used to find $\hat{\theta}$ on *Markov Chains* to find $\hat{\mu}$ for *Hidden Markov Models*. This way we will find that:

$$\mu_{c,d} = \frac{\eta_{c,d}}{\sum_{d'} \eta_{d,c'}} \tag{4.25}$$

This is the expected number of times we observed word d when we were at the state c. A number that is also easy to obtain in $O(n)$ time. With that, we conclude the maximum likelihood parameter estimation for the *Hidden Markov Model*. This procedure presupposes that we have labeled data, since we need to observe the correct labels Y to calculate $\hat{\theta}$ and $\hat{\mu}$. If we only have unlabeled data, we can do the parameter estimation with the Baum-Welch algorithm, though the results are much less reliable on the sequence labelling task.

The last task we need to do is to calculate the most likely sequence of labels given a sequence of observations. To obtain this sequence exactly we can employ the Viterbi algorithm, which is explained in Appendix (REF). For higher order Hidden Markov Models the best sequence of labels can be computed with a variable state Viterbi approach **?**. However, as we increase k, this computation becomes exponentially more expensive. The beam-search strategy may be employed for a faster search, but we found that for $k \leq 4$, the Viterbi algorithm is still viable.

(DISCUSS HOW TO SMOOTH WITH LAPLACE SMOOTHING)

(CITE MCCALLUM HERE) HMM based taggers have been successfully applied in many NLP and WDE tasks **???**. They are incredibly fast to train and also they are very interpretable, making them a good choice for a first approximation. However, these models are highly dependent on the right selection of features, what may outweigh the benefit of a small training cost.

4.1.2 Maximum Entropy Models and Linear Chain Conditional Random Fields

(DISCUSS HOW UP TO THIS POINT WE HAVE BEEN WASTING MODELLING EFFORT ON MODELLING THE JOINT DISTRIBUTION AND WE COULD GO DIRECTLY TO THE CONDITIONAL)

An example of Maxent classifier for sequence tagging: https://www.aclweb.org/anthology/W96-0213.

A *Linear Chain Conditional Random Field* is the particular case of a *Conditional Random Field* where the labels are conditionally independent given the previous observations. (THIS SENTENCE HAS TO BE CHECKED). It is the generalized case of a Maximum Entropy Markov Model, which is itself a combination of Maximum Entropy Classifier for sequences.

The task of labeling sequences in *Natural Language Processing* can be seen as a classification task where we want to predict a label y, given a word and its context, which we call jointly as x. This context can be composed of any relevant features extracted from the text, such as surrounding words, previous labels, and sentence length. Because of its mathematical properties, the *Maximum Entropy* framework provides a compelling way for estimating the probability of a label given its linguistic context, that is $P(Y|X)$.

The concept of Information Entropy established by Shannon (CITE HERE, A mathematical theory of communication) aims to quantify the amount of information expressed in a statistical distribution. Namely, the entropy H of a probability distribution X with a probability mass function p is given by:

(THE ENTROPY HERE SHOULD BE THE JOINT ENTROPY BETWEEN WORDS AND TAGS)

$$H(p) = \sum_{x \in X} p(x) \cdot log_2 \left(\frac{1}{p(x)} \right) \tag{4.26}$$

In fact, the logarithm in the entropy function can be taken in any base. The change of base will only provide a linear scaling of the same entropy function. In this case, $H(p)$ is essentialy the expected number of bits necessary to encode the value of a single random outcome drawn from X, that is $E[log_2 \left(\frac{1}{p(X)} \right)]$. The more bits we need to encode the possible outcomes from a probability mass function, the more uncertain we are about the underlying distribution. For example, if an event has only one possible outcome, we always know the value of a random sample, therefore $H(p) = 0$ and the entropy is minimum. On the other end, if the probability distribution is uniform, all outcomes have the same probability, and therefore we are as uncertain as we can be and the entropy is maximum.

The principle of maximum entropy (CITE HERE) [Jaynes, 1957, Good, 1963] states that, when we do not know exactly what probability distribution generated a sample, the best estimate for the parameters of this probability distribution is the one that makes the least assumptions, or the one with maximum entropy. This is the distribution that is closest to the uniform distribution.

A similar argument can be made to limit the number of free parameters in the model. That is, when comparing two models with similar predictive power, the one with the least degrees of freedom should be prefered. However in practice, this is rarely taken into account.

The principle of maximum entropy is similar to a principle in the philosophy of science called Occam's Razor, that is, when there are two explanations for an outcome, the one that makes the least number of assumptions is the best. Which is a good rule of thumb for science in general, eventhough, ultimately nothing guarantees that the rules that explain reality will be simple.

Naturally, if we optimize the objective function without constraints, we will find that the *Maximum Entropy* estimator will yield an uniform distribution, meaning that we are no better than simply predicting labels at random for our data. So, it becomes necessary to establish some constraints over our optimization problem.

The choice of constraints is more or less arbitrary, but under the *Maximum Entropy* framework, there is a strong motivation for setting these constraints in a way that makes the *Maximum Entropy* estimator equivalent to the *Maximum Likelihood* estimator for our model. By doing it, we can make sure that the model fits the data as tightly as possible (the maximum likelihood estimate), while making as few assumptions as it needs to (the maximum entropy estimate).

If we set these constraints such that, given k binary feature functions f_i, we have k constraints:

$$E_{p(a,b)}[f_i] = \sum_x \tilde{p(a)} p(b|a) f_i(x) \tag{4.27}$$

$$E_{\tilde{p(a,b)}}[f_i] = \sum_x \tilde{p}(x) f_i(x) \tag{4.28}$$

$$E[f_i] = E_{\tilde{p(x)}}[f_i], \forall i \in \{1, 2, \ldots, k\} \tag{4.29}$$

On which $E[f_i]$ is the model expectation of f_i and $E[\hat{f_i}]$ is the observed expectation f_i. And we maximize the conditional entropy subject to these constraints:

$$H(p) = -\sum_{a,b} \tilde{p}(b) p(a|b) log p(a|b) \tag{4.30}$$

Note that we use the simplification $\tilde{p}(b) \approx p(b)$ suggested by Berger et. al., because the probability space of $p(b)$ is too big to allow a simple calculation. (PAGE 12 of http://www.ai.mit.edu/courses/6.891-nlp/READINGS/adwait.pdf)

BERGER is here https://www.aclweb.org/anthology/J96-1002.

We can once again resort to Lagrange Multipliers to transform this problem into an unconstrained optimization problem. The Lagrangian Λ takes the form:

$$\Lambda(p, \theta, \theta_p, \lambda, \beta) = H(p) + \sum_i^n \lambda_i (E_{p(x)}[f_i] - E_{\tilde{p(x)}}[f_i]) + \beta(\sum_{x,y} \tilde{p}(x) p(y|x)) - 1 \tag{4.31}$$

Now we need to make set all derivatives equal to zero to find the point that maximizes the Lagrangian, and by consequence the conditional entropy subject to our constraints. The derivatives relative to λ and β simply yield the initial constraints. Now the the derivative relative to p is:

$$\frac{\partial \Lambda}{\partial p} = -1 - log p(y|x) + \beta + \sum_i^n \lambda_i f_i(x, y) = 0 \tag{4.32}$$

$$p(y|x) = exp(\sum_i^n \lambda_i f_i(x, y) + \beta + 1) 4.39 \tag{4.33}$$

Also, we know by our initial constraints that:

$$\sum_y p(y|x) = 1 \tag{4.34}$$

$$\sum_y exp(\sum_i^n \lambda_i f_i(x,y) + \beta + 1) = 1 \tag{4.35}$$

$$e^\beta \cdot \sum_y exp(\sum_i^n \lambda_i f_i(x,y) + 1) = 1 \tag{4.36}$$

$$e^\beta = \frac{1}{\sum_y exp(\sum_i^n \lambda_i f_i(x,y) + 1)} \tag{4.37}$$

$$\tag{4.38}$$

By replacing β in Equation 4.39 we finally get:

$$p(y|x) = \frac{exp(\sum_i^n \lambda_i f_i(x,y))}{\sum_y exp(\sum_i^n \lambda_i f_i(x,y))} \tag{4.39}$$

At this point, we know that the model that maximizes the conditional entropy subject to the constraints established in (REFERENCE TO CONSTRAINTS) has to belong to the parametric family described in Equation **??**. This equation describes a family of exponential probability models known as logistic classifiers or maximum entropy classifiers, this is the unique solution to our optimization problem (CHECK APPENDIX).

(SKETCH THE PROOF FOR UNICITY HERE)

Furthermore, by maximizing the Lagrangian Λ we can find the optimal set of parameters λ_i (a results granted by the Kuhn-Tucker theorem). This optimization problem is identical to maximizing the log likelihood:

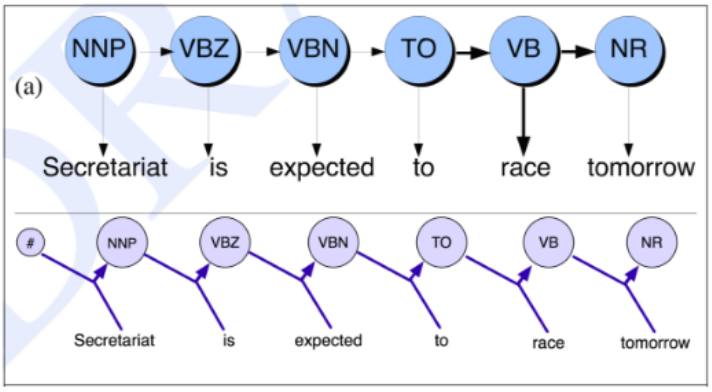
(THIS RESULT SHOULD BE BETTER EXPLORED)

$$\mathscr{L}(Y|X;\theta) = log \prod_{x,y} p(y|x)^{\tilde{p}(x,y)} = \sum_{x,y} \tilde{p}(x,y) logp(y|x) \tag{4.40}$$

When the training samples are independent, maximizing the likelihood

And this is essentialy the same thing as maximizing the cross-entropy (https://peterroelants.github.io/posts/cross-entropy-logistic/). Now, there is no closed form expression to find the optimal set of parameters θ that maximizes the likelihood / entropy, so we must resort to numerical optimization methods such as Stochastic Gradient Descent or L-BFGS.

In the particular case of sequence labeling, the simplest conceivable classifier inside the Maximum Entropy Framework would be a multinomial logistic regression that decodes each label independently. That is, given a context x the classifier would predict a label y without taking in consideration any of the previous or forward labels. This would be the discriminative analogue to the Naive Bayes model. However, the independence assumption is too restrictive for most language related tasks. Maximum Entropy Markov Models (MEMM) try to overcome this problem by incorporating the insights obtained from Hidden Markov Models into the Maximum Entropy Models while preserving the flexibility of discriminative models.



Contrast in state transition estimation between an HMM and a MEMM.
(taken from "Speech and Language Processing" Daniel Jurafsky & James H. Martin)

Figure 4.2. HMM VS MEMM

In MEMMs, we assume that the label probabilities are connected in a Markov Chain like in a HMM. However, in HMMs we have transition probabilities $P(Y_t|Y_{t-1})$ and emission probabilities $P(X_t|Y_t)$, but in MEMMs we have a single transition function $P(Y_t|Y_{t-1}, X_t)$ that combines evidence from the previous state Y_{t-1} and the observations X_t. Observe that, with this model we do not care about the probability of the observations X_t in contrast to generative models, that model the joint probability $P(x, y)$. Because of this, the feature vector X_t may contain overlapping and non-independent features since there is no independence assumption.

In MEMMs, we have $|Y|$ transition functions $P_{y'}(y|x) = P(y|y', x)$, each constituting a different model that has the exponential form:

$$P_{y_{t-1}}(y_t|x_t) = \frac{1}{Z(y_{t-1}, x_t)} exp(\sum_i (\lambda_i f_i(y_{t-1}, x_t))) \tag{4.41}$$

Where $Z(y', x)$ is a partition function that guarantees that the distribution over

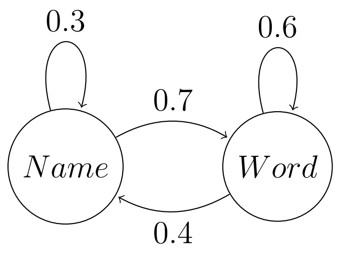


Figure 4.3. Graph describing the label bias problem.

the next states s sums to one, starting from the previous state s'. That is, each exponential function is normalized locally in the context of the defining state s'. Like the exponential models described earlier, this form is a unique distribution that maximizes the entropy and the likelihood subject to the constraints that the expected value of each feature in the learned distribution agrees with the expected value in the training set, starting from state s'.

We can calculate the probability for a sequence of labels by chaining the transition probabilities:

$$P(y * | x*) = \prod_{i=1} P_{y_i}(y_i | y_{i-1}, x_i) \tag{4.42}$$

And the most probable sequence can be easily calculated with the Viterbi algorithm, as we did in HMMs. Originally, the maximum entropy/likelihood parameters for the MEMM were calculated with Generalized Iterative Scaling algorithm, but this method has fallen out of favor in face of gradient based numerical optimization methods such as SGD. It is also possible to perform parameter estimation with incomplete labels by employing a variation of the Baum-Welch algorithm.

MEMMs provide a more flexible approach to the problem of sequence labeling in comparison with HMMs, however they suffer from the label bias problem. To understand the label bias problem, consider the case where we have a MEMM named entity classifier that aims to label person names and locations in a sentence.

INSERT HERE EXPLANATION OF LABEL BIAS PROBLEM FOR SEQUENCE LABELING

Conditional Random Fields solve the label bias problem by jointly decoding the entire sequence of labels. That is, instead of normalizing the transition probabilities at each timestep as described in Equation 4.42, we normalize the probabilities for the entire sequence of labels, that is:

$$P(Y|X) = \frac{1}{Z(x)} \prod_{t=1}^{T} exp\left(\sum_{k=1}^{K} \theta_k f_k(y_{t-1}, y_t, X)\right) \tag{4.43}$$

$$Z(x) = \sum_{Y} \prod_{t=1}^{T} exp\left(\sum_{k=1}^{K} \theta_k f_k(y_{t-1}, y_t, x)\right) \tag{4.44}$$

The space of label combinations y' is huge $|Y|^n$, being n the sequence length.

VERY GOOD TUTORIAL ON CRFs: https://homepages.inf.ed.ac.uk/csutton/publications/crftu fnt.pdf

which is a sum over all possible label assignments Y. The partition function can be efficiently and exactly calculated with the sum-product algorithm. Parameter estimation is usually done through negative log likelihood minimization. The function can be optimized with techniques suitable for other maximum entropy models such as L-BFGS **?**. The most likely label sequences can be decoded with the Viterbi algorithm, as was the case for HMMs.

CRFs are more general than HMMs because the transitions from y_{t-1} to y_t can depend on the whole vector of observations X. This flexibility of feature functions allows for a wide range of possibilities.

Recently, Maximum Entropy Markov Models and Conditional Random Fields have been largely replaced by neural network based models. However, Conditional Random Fields are still employed as the output layer of complex neural architectures.

4.2 Neural Network Architectures

The recent upsurge in the popularity of neural networks owes to the increasing computational capacity brought by Graphical Processing Units and discoveries such as Deep Belief Networks (CITE HINTON) in 2006 Autoencoders(?) and LSTMs (CITE HOCHREITER) in 1997 that helped overcome the limitations of earlier neural architectures. Deep neural architectures established new state of the art results for problems ranging from computer vision to speech recognition.

Neural networks are a family of classification algorithms that were vaguely inspired in the functioning of the human brain. They consist of a weighted graph of artificial neurons, which are functions that receive an input from other neurons or from an input vector and produce an output according to their activations. Neural networks are a general framework that can encompass other machine learning approaches. For example, a single layer neural network with a softmax activation function assumes the same form of a multinomial logistic classifier (PUT PICTURE HERE).

An extension of the logistic classifier to a neural network with multiple layers can still be trained the same way as the original model, that is, by minimizing the cross entropy. However, once we add multiple layers on top of the logistic classifier, the optimization problem no longer retains its convexity. This means that numerical optimization methods can get trapped on local minima and miss the best set of parameters for the model. For a long time, this limitation hindered the development of neural architectures. Nonetheless, in practice, the rugged shape of the cost function seems to be of minor importance. Even though the global optimization is not guaranteed, in most classification scenarios finding a local optimum solves the parameter estimation problem sufficiently well. Also, heuristic methods such as adding momentum to gradients can help the optimization algorithm avoid being trapped.

The topic of neural networks is incredibly vast and cannot be properly discussed in this dissertation. For a thorough view of the field consult (CITE BENGIO DEEP ARCHITECTURES FOR AI).

In contrast to earlier probabilistic models that were more statistically oriented, many improvements to neural networks derive from empirical results obtained in specific applications. In the probabilistic modelling of text, we are especially interested in the subclass of recurrent neural networks (RNN). RNNs have been successfully employed on numerous NLP tasks such as language modelling, POS tagging, speech recognition and NER. In RNNs, some of the neuronal layer activations become inputs to the same layer at the next timestep. Additionaly, different from feed-forward neural networks, RNNs can retain information in their internal state. This characteristic can function as a memory cell, preserving long distance relationships across the chain, and making them more suitable for processing sequences, and consequently for solving text related tasks. Figure 4.4 describes an RNN for sequence labeling unrolled through multiple timesteps.

At each timestep, the neural network computes a hidden state h_t using an input vector x_t and the previous hidden state h_{t-1}, that retains information from past iterations. The input vector x_t for our RNN can consist of word features similar to the ones used in the previously discussed statistical models encoded in one-hot vectors, word embeddings or a combination of both. Finally, the RNN produces an output vector y_t representing the label for that timestep. A common definition for an RNN cell is given by the equations:

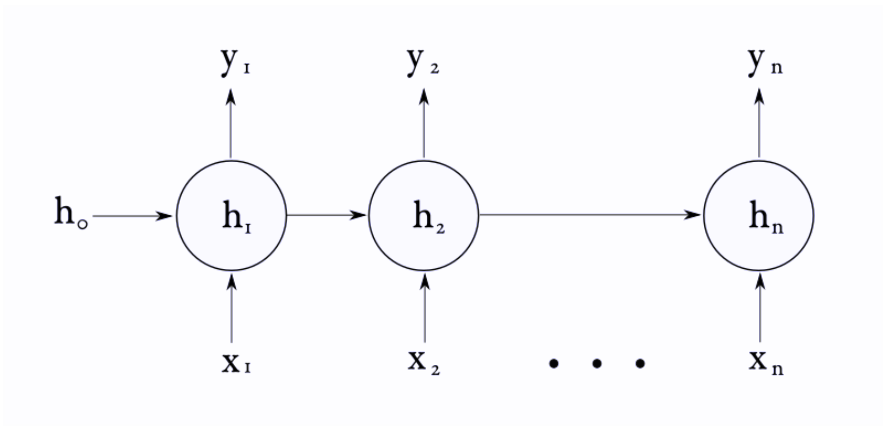


Figure 4.4. RNN for NER

$$h_t = tanh(W_x x_t + W_h h_{t-1})$$
$$y_t = softmax(W_y h_t)$$

Where W_x, W_h and W_y are weight matrices that can be trained with the Back-propagation Through Time (BPTT) algorithm. On a sequence labeling task, W_y is a $|H| \times |Y|$ weight matrix where H is the size of the hidden layer, defined experimentally according to the task, and Y is the number of classification labels for our problem. With the softmax activation, the RNN will generate a probability distribution across the range of possible labels and we can simply select the most probable label at each timestep, assuming independence between labels. Theoretically, RNNs are capable of learning and retaining long term dependencies with their internal state h_t. However, in practice, it becomes difficult due to the vanishing gradient problem.

(INSERT EXPLANATION OF VANISHING GRADIENT PROBLEM HERE)

Long short term memory networks (LSTM) were introduced by Hochreiter and Schmidhuber **?** with this problem in mind and have been popularized since then.

LSTMs incorporate a memory cell c in the RNN definition and three gates to control the flow of information that comes in and out of the memory cell. The input gate Γ_i controls the amount of new information that will flow into the memory cell, the forget gate Γ_f controls the amount of previous information that will be retained in the memory cell, and the output gate Γ_o controls the amount of information stored in the memory cell that will be used to compute the output activation of the LSTM unit. LSTM cell implementations vary slightly in the literature. A visual description of our LSTM cell is provided in Figure 4.5.

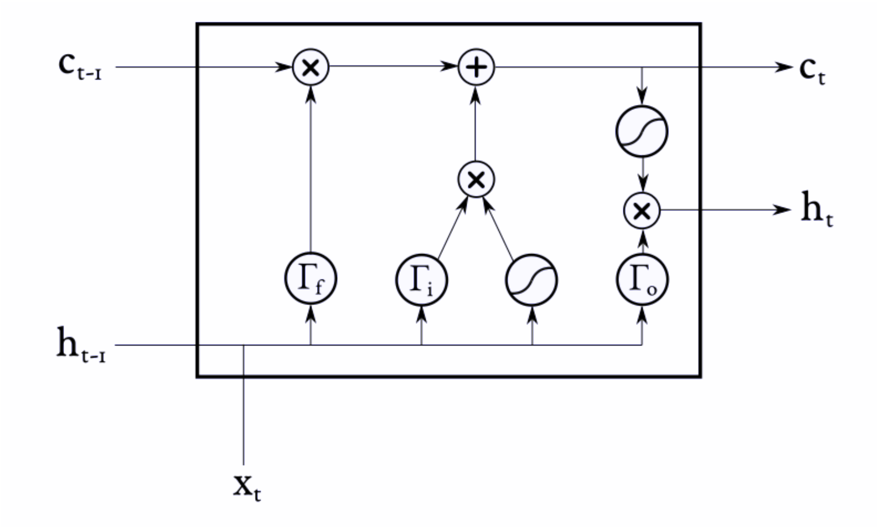


Figure 4.5. LSTM Cell

The equations for the LSTM cell are:

$$\begin{aligned}
\Gamma_i &= \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \\
\Gamma_f &= \sigma(W_f \cdot [x_t, h_{t-1}] + b_f) \\
\Gamma_o &= \sigma(W_o \cdot [x_t, h_{t-1}] + b_o) \\
c_t &= \Gamma_f * c_{t-1} + \Gamma_i * tanh(W_c \cdot [x_t, h_{t-1}] + b_c) \\
h_t &= \Gamma_o * tanh(c_t)
\end{aligned}$$

Where σ is the logistic sigmoid function. Γ_i, Γ_f, and Γ_o are the input, forget and output gates, respectively, and W_i, W_f, W_o are the weight matrices corresponding to each gate. c_t is the cell state at time t and h_t is the hidden state at time t. The vector $[x_t, h_{t-1}]$ is formed by concatenating the current input vector x_t and the hidden vector from a previous timestep h_{t-1}. Finally, $A*B$ represents the element-wise multiplication of matrices A and B and $A \cdot B$ represents the dot product of A and B.

This implementation differs from the LSTM cell described in Huang et al. **?** in that the gates Γ_i and Γ_f do not receive inputs from the previous cell state c_{t-1} and the output gate Γ_o does not receive inputs from the current cell state c_t. This variation produces little difference in terms of model accuracy on the performed task, but it reduces model complexity.

4.2.1 BI-LSTM-CRF

On named entity recognition tasks, both past and future words are important to attribute a label at time t, however a regular LSTM network only takes past states into consideration. A bidirectional LSTM solves this problem by stacking two regular LSTMs, and feeding them with observations in opposite directions. The first LSTM receives forward states and the second LSTM receives backward states. The hidden states from both networks can then be concatenated at each timestep to produce output labels. With this architecture, LSTM cells may use information from past and future timesteps to decide the label at time t.

Huang et al. **?** proposed a bidirectional LSTM with a CRF layer (BI-LSTM-CRF) on the output to tackle the sequence tagging problem. The main benefit of adding a CRF layer in the neural sequence model is that the labels are jointly decoded for a whole sentence instead of being predicted individually. Another possibility would be to use a beam search decoder to find an optimal sequence of labels. Predicted tags should be highly correlated in a named entity recognition task, so it is desirable to predict sequences conjointly. The BI-LSTM-CRF is described in Figure 4.6.

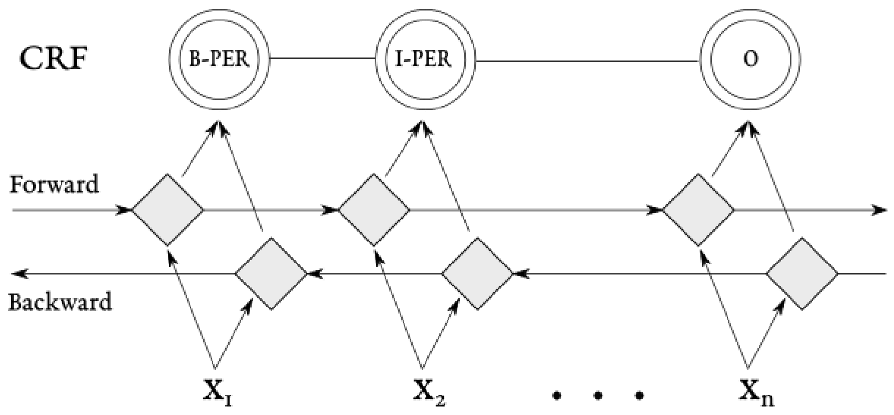


Figure 4.6. Bidirectional LSTM-CRF

This architecture achieved an F1 score of 90.10 on the English data from the CoNLL-2003 NER shared task **?**, in contrast to 85.17 for a bidirectional LSTM without a CRF layer. In our experiments, the LSTM-CRF architecture uses a bidirectional LSTM with 100 hidden states, no peepholes and input and output dropout layers with a dropout rate of 0.5. The dropout layers have proven to be very important to prevent overfitting and allow better generalization.

4.2.2 CNN character representations

Ma and Hovy **?** proposed to add a convolutional neural network (CNN) layer on top of a bidirectional LSTM-CRF to encode character-level information. The CNN layer is described visually in Figure 4.7.

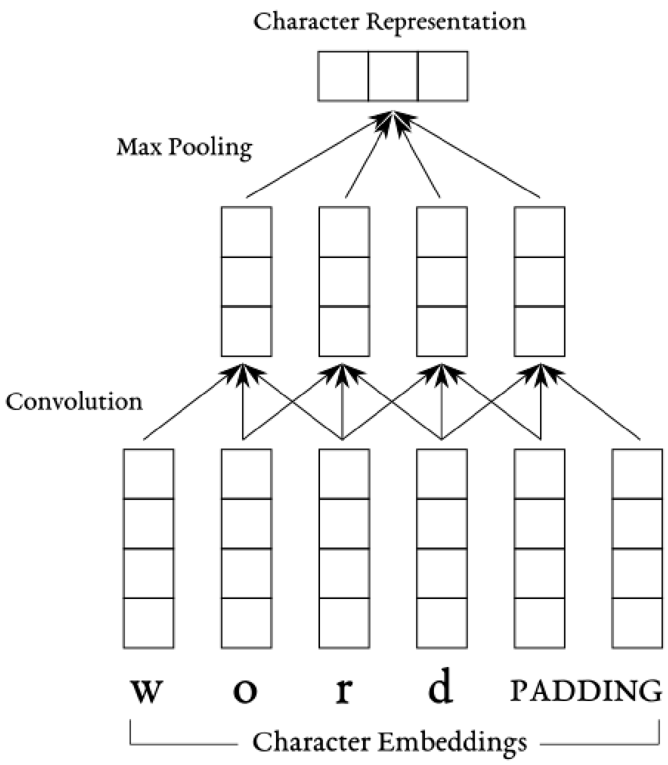


Figure 4.7. CNN based character representations

The convolutional neural network receives character embeddings as inputs. The character representations generated by the CNN are combined with word level representations and fed to the BI-LSTM-CRF described in section 4.2.1. This architecture can learn morphological features that are very useful in the NER task, since similar named entities often present morphological similarities. This architecture obtained an F1 score of 91.21 in the CoNLL2003 dataset. In our experiments, the LSTM-CRF architecture with CNN character representations uses a one dimensional convolutional neural network with 30 filters and a window size of three characters on top of the LSTM-CRF architecture. The character embeddings fed to the CNN have 30 dimensions that are randomly initialized.

4.2.3 LSTM character representations

Lample et al **?** proposed to use a bidirectional LSTM to model character-level representations on top of a BI-LSTM-CRF. Combining the forward and backward LSTM hidden states to form the character representation, as described in Figure 4.8.

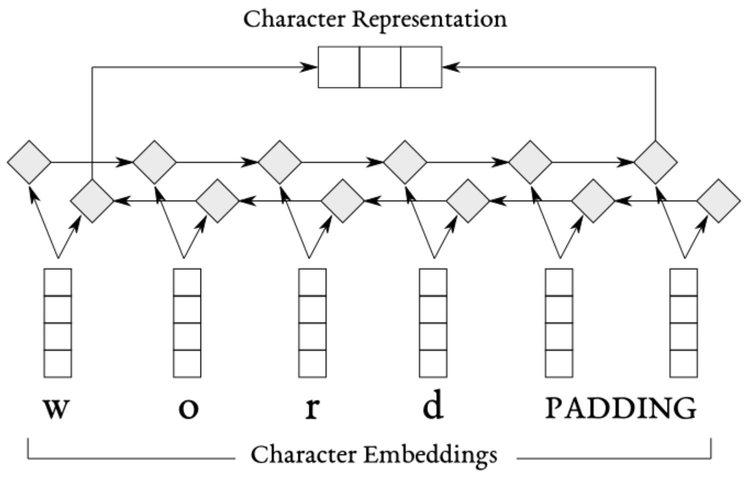


Figure 4.8. LSTM based character representations

This character representation is also combined with a word representation and fed to a BI-LSTM-CRF network. The forward state is expected to be a better representation of the suffix of a token, and the backward state is expected to be a better representation of the prefix of a token. This differentiates the architecture from the CNN based approach described in Section 4.2.2, because CNN filters discover positional invariant features, while LSTMs can better represent suffixes and prefixes. In our experiments, the LSTM-CRF architecture with LSTM character representations was implemented with a bidirectional LSTM with 25 hidden states, producing character representations of size 50. The character embeddings have 30 dimensions that are randomly initialized.

4.2.4 Word Embeddings

An important element of recurrent neural network architectures for text related tasks is the choice of word embeddings. Word embeddings are lower-dimensional representations of words in continuous space. That is, each word is represented by a vector of continuous features (tipically a few hundred dimensions), often obtained with unsupervised clustering methods for words in large corpora. In comparison, one-hot encoded word vectors, that were popular before the introduction of word embeddings, are vectors with one dimension per word in the vocabulary (tipically a few hundred thousand dimensions) with zeros in all positions except for the index relative to that word. Figure **??** describes the difference between these two types of encodings.

There are many advantages to the use of word embeddings relative to other encodings. First, the number of weights learned by the sequence model is smaller because of the dimensionality reduction. Second, the mapping of words into feature vectors makes possible the comparison of similarity between words according to their shared features. Finally, and most importantly, word embeddings can be pretrained without supervision on large corpora and be used on tasks for which there is little labeled data. Pretrained word embeddings are a powerful form of transfer learning, what means bootstrapping the learning of model parameters by first training it on a bigger dataset. This is especially useful to the named entity recognition task, because named entitities that belong to the same class tend to have similar word embeddings. Thus, with good word embeddings, a neural network can predict the correct label for a word that was never seen in training because of its similarity to other words that occurred in the training set.

Pretrained word embeddings can also be further improved by fine tuning the feature values in the specific problem of sequence labeling, by allowing backpropagation to alter the embeddings. However, due to the limited size of most sequence labeling datasets, this technique ends up introducing noise to the word representations instead of improving the word representations.

There are multiple methods that produce word embeddings, and most of the official implementations also provide pretrained word embeddings for the English language. In this work, we explored three sets of pretrained embeddings: Word2Vec, Glove and Elmo (CITE HERE). A brief description of these models is provided in Table 4.1.

Model	Description
Word2Vec	Regular HMM
Glove	HMM with $k = 2$
Elmo	HMM with $k = 3$

Table 4.1. Model descriptions

Most breakthroughs in sequence labeling tasks in the past few years came through the introduction of novel methods for constructing word embeddings. Different from earlier methods such as Word2Vec and Glove that produced static vectorial representations of words, more recent approaches such as Elmo and Bert produce context dependent embeddings for a specific dataset with a neural network pretrained on a language modelling task.

(HOW MUCH ELMO AND BERT HELPED IN THE CONLL-2003 TASK)

As word embedding and language modelling methods become more sophisticated, it comes to mind if the complexity is justifiable.

4.2.5 Technical Concerns

Eventhough word embeddings and character representations can make up for the feature engineering that was required by earlier statistical approaches to sequence labeling, there are still some technical concerns to be faced that require quite a bit of engineering. The loss functions that need to be optimized by our neural architectures is not convex, therefore the choice of the optimization function can impact the resulting set of parameters for our model as well as the parameter choice such as the learning rate for these functions. Neural networks are also prone to overfitting, what can be partially avoided with the addition of dropout layers (CITE HERE) or with the early stopping strategy over a validation dataset **?**.

Another difficulty in training neural networks comes from the training time required for convergence to a good set of parameters. This sometimes requires the employment of expensive hardware, layer normalization https://arxiv.org/pdf/1607.06450.pdf and parallelization strategies. All of this adds into the complexity, hardware cost and training cost of our models, many times providing only slight improvements over simpler approaches. This is not to say that simple models are preferable, but depending on the task, deep neural architectures may be an over expensive approach.

Chapter 5

Sequence Labeling on HTML

Up to this point, we have discussed sequence models that can be generally employed in many natural language tasks and also in other similar scenarios. However, there are specificities related to the HTML structure and the problem of named entity extraction on HTML that can be explored to allow further improvement at the researcher name extraction task.

5.1 Self training for Hidden Markov Models

As described on Section **??**. The simplest conceivable Hidden Markov Model for the sequence labeling on HTML task is a generative classifier that has the form:

$$P(X, Y) \propto \prod_{i=1}^{n} P(X_i|Y_i)P(Y_i, Y_{i-1}) \tag{5.1}$$

Where X is a sequence of words and Y is a sequence of labels, each with size n. This means we need to estimate the probabilities:

- $P(X_i|Y_i)$: the emission probabilities of word X_i given label Y_i
- $P(Y_i|Y_{i-1})$: the transition probabilities of goind from label Y_{i-1} to label Y_i

Surely, we may consider X_i to be a feature vector $X_i = \{f_{i,1}, f_{i,2}, ..., f_{i,k}\}$ as long as all the features are independent. That is:

$$P(X_i|Y_i) = P(f_{i,1}, f_{i,2}, ..., f_{i,k}|Y_i) = \prod_{j=1}^{k} P(f_j|Y_i) \tag{5.2}$$

With this assumption, we can estimate parameters as described by the equations in Section **??** (CITE SPECIFIC EQUATIONS). This approach yields consistent results, but if we try to use HTML features such as the HTML element encompassing a word or the CSS classes related to this element, we will find a problem. There is not much statistical similarity between the HTML structure in different web pages. What this means is that, only because a given named entity label occurs more often inside "<div>" tags in a page, that does not mean it is the case for most web pages. Consider for example a faculty web page that shows researcher names in a table ("<td>" tags) in contrast to a web page that organizes researcher names in a list ("<li>" tags). If the first page is in our training set, the second page is in our test set and we use the HTML tag feature to predict the labels of researcher names we will probably get many wrong predictions.

Nonetheless, the HTML features are not useless. Inside a single web page, the HTML tag is a good predictor for the label attributed to a word. A word's HTML context is a good predictor of its label. The question is how to incorporate this intution in the Hidden Markov Model.

Consider a set of textual features unrelated to the HTML structure $F^T = \{f_1^T, f_2^T, ..., f_m^T\}$ a set of HTML features that are related to the HTML structure $F^H = \{f_1^H, f_2^H, ..., f_k^H\}$. Next, consider two HMMs ϕ_1 and ϕ_2, where ϕ_1 incorporates only the textual features F^T, while ϕ_2 incorporates the whole set of features $F^T \cup F^H$. Then, if $\tilde{\phi_1}$ is the Hidden Markov Model with maximum likelihood parameters for a given training set, we can estimate labels $\tilde{Y}$ for the test set and approximate the probabilities $P(f_i^H|Y)$ by:

$$P(f_i^H|y) \approx MLE_{\tilde{y}}P(f_i^H|y) \tag{5.3}$$

Where $MLE_{\tilde{y}}P(f_i^H|y)$ is the maximum likelihood estimator $P(f_i^H|y$ considering that the predicted labels for our test $\tilde{y}$ set are the correct labels. This self training strategy can be implemented like this:

- Train the HMM without any HTML features.
- Compute labels for a website with the trained HMM.
- Use the computed labels as a proxy for the actual labels in the website and estimate HTML feature frequencies for this website alone.
- Recompute the labels now using the HTML feature probabilities.

With some effort, this strategy could be incorporated in the Baum-Welch algorithm, but this heuristic approach already yields a consistent improvement. In theory, this strategy could be used with any sequence tagger, however retraining a classifier with new features can become prohibitively expensive depending on the algorithm being used. This strategy is only possible because the computation of HTML feature frequencies can be performed very quickly. This adds very little overhead to the original HMM and incorporates our intuition that HTML features are useful in sequence labeling tasks for HTML.

5.2 Attentions Models

The self-training strategy for HMMs demonstrates a way to incorporate HTML features in sequence models, however it is not clear how to apply the same intuition to neural networks. Ultimately, we want the model to consider the predictions that it made for words in similar HTML contexts when constructing the neural representation for the current word in a sequence. A natural way to incorporate this intuition into the neural architectures described in Chapter **??** is with the use of self attention mechanisms (https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf). Originally, attention mechanisms were developed with the goal of solving sentence alignement for neural machine translation (https://arxiv.org/pdf/1409.0473.pdf), but since then it found other uses in Natural Language Processing. An attention mechanism is a way to combine inputs from multiple timesteps in a sequence to perform an operation at the current timestep. Figure **??** describes an attention mechanism that incorporates evidence from the whole sequence at time t.

In a LSTM-CRF model, the bidirectional LSTM layer produces output representations at each timestep, producing a $T \times H$ matrix ϕ, where T is the number of timesteps in the sequence and H is the LSTM hidden layer size, which is defined arbitrarily (we omit the batch size in this matrix representation). If the sequence length in the dataset varies, we can simply pad the short sentences with zero vectors. In other words, the matrix ϕ constitutes a neural representation for a sentence with vectors of size H representing words at each timestep. In the original LSTM-CRF model without an attention layer, these neural representations would be passed directly as features to a linear chain conditional random fields decoder, but in the Self-Attended LSTM-CRF we add an attention mechanism between the LSTM output and the CRF input as described in Figure **??**.

Now we need to find a way to combine the vectors at each timestep in a way that transforms the representations according to the similarity between HTML contexts. Essentialy, we want to compute a $T \times T$ attention matrix $alpha$ where $alpha_{i,j}$ is the weight attributed to the word representation h_j at timestep i. In other words, $alpha_t$ is measuring the amount of attention that we pay to all the words in a sentence at timestep t. With the α matrix, we can calculate new representations c_i for each timestep by performing a linear combination of the hidden states according to their attention values:

$$c_i = \sum_j = \alpha_{i,j} h_j \tag{5.4}$$

Also, the attention matrix is calculated by:

$$\alpha_{i,j} = \frac{exp(A(h_i, h_j))}{\sum_k exp(A(h_i, h_k))} \tag{5.5}$$

Where $A(h_i, h_j)$ is an attention function that computes the similarity between hidden states at timesteps i and j and outputs a real number. The exponentials are a softmax normalization function to make sure that $1 \geq \alpha_{i,j} \geq 0$ and $\sum_j \alpha_{i,j} = 1$. Now we propose two ways of defining the attention function A. The hard attention function and the soft attention function.

5.2.1 Hard Attention

The hard attention function computes a binary similarity that is either one when contexts are identical or zero when they are dissimilar. The attention function is then given by:

5.2.2 Soft Attention

5.2.3 Dataset split

Soft attention

Hard attention

5.3 F1 Optimization

F1 optimization

Chapter 6

Experiments

6.1 Technical specifications

All neural models were trained using mini batch Stochastic Gradient Descent over 50 epochs with batch size 10, learning rate 0.01, momentum 0.9 and decay rate 0.05. We used early stopping **?** to select the best parameters, considering the F1 measure in the validation set. All neural models used GloVe 100-dimensional word embeddings **?** that were fine tuned during training. In the case of NER on HTML, word embeddings work similarly to a gazetteer. Named entities with the same type have similar embeddings, so good word embeddings can achieve exceptional performance with little training and without a gazetteer.

We conducted experiments to evaluate sequence labeling methods of named entity recognition on HTML in the context of Web data extraction using the dataset described in Section **??**. The tested models are described in Table **??**.

Model	Description
hmm-1	Regular HMM
hmm-2	HMM with $k = 2$
hmm-3	HMM with $k = 3$
crf	Linear chain conditional random fields
bi-lstm-crf	BI-LSTM-CRF model
bi-lstm-crf-cnn	BI-LSTM-CRF with CNN character representations
bi-lstm-crf-lstm	BI-LSTM-CRF with LSTM character representations

Table 6.1. Model descriptions

The evaluation of model performance was done through the precision, recall and F1 scores **?**. Precision is the percentage of named entities found by the model that are correct. Recall is the percentage of named entities that are present in the corpus and were found by the model. The F1 score is a composite measure that combines precision and recall with the formula:

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{6.1}$$

Named entities were only considered to be correct if they were a complete match of the corresponding entity in the dataset.

6.2 Experiment 1: No features

Experiment 1 aimed to evaluate the performance of sequence model with no features besides GloVe-100 embeddings. In the case of HMMs, only the lowercase unaccented token was used as a feature. Table **??** shows the Precision (P), Recall (R), and F1-scores (F1) for this experiment.

Model	Validation			Test		
	P	R	F1	P	R	F1
hmm-1	0.6965	0.5749	0.6299	0.6263	0.4431	0.5190
hmm-2	0.7047	0.6286	0.6645	0.6480	0.5222	0.5783
hmm-3	0.6127	0.6141	0.6134	0.5471	0.4634	0.5018
crf	0.7173	0.6683	0.6920	0.6671	0.5868	0.6244
bi-lstm-crf	0.8484	0.9044	0.8755	0.8358	0.8497	0.8427
bi-lstm-crf-cnn	0.9058	0.9575	0.9309	0.8779	0.8737	0.8758
bi-lstm-crf-lstm	0.9134	0.9435	0.9282	**0.8920**	**0.8815**	**0.8867**

Table 6.2. Precision, recall and F1 in the NER on HTML dataset for models that incorporate no features

Without carefully designed features or gazetteers, HMMs and CRFs have a very poor performance, achieving an F1-score of only 0.5783 for HMM-2 and 0.6244 for CRF at the test set. This is expected, since these models rely on good feature selections.

The neural models achieved high F1-scores in the test set even with the absence of features. The plain BI-LSTM-CRF architecture improved performance significantly in comparison with the conventional CRF (0.8427 against 0.6244). Also, neural character representations boosted performance by a significant margin reaching an F1-score of 0.8758 for CNN-based representations and 0.8867 for LSTM-based representations. LSTM based representations were superior in modelling morphological features, perhaps because they are able to differentiate suffixes and prefixes, while CNN filters are position invariant.

The results in Experiment 1 also show that pretrained word embeddings can work as a kind of universal gazetteer. Words with similar embeddings are likely to belong to the same class. This knowledge combined with the ability to learn morphological features can make up for the scarcity of textual data on some webpages.

6.3 Experiment 2: All features

Experiment 2 aimed to evaluate the performance of sequence model with all the features described in Table 3.5. In this experiment, we also evaluate the self-training strategy for HMMs described in Section **??**. The self trained HMMs are described with the suffix "+ST". Table **??** shows the results for Experiment 2.

Model	Validation			Test		
	P	R	F1	P	R	F1
hmm-1	0.6061	0.7282	0.6616	0.7106	0.7633	0.7360
hmm-2	0.6279	0.7550	0.6856	0.7521	0.7810	0.7663
hmm-3	0.6573	0.7819	0.7142	0.7523	0.7795	0.7657
hmm-1+ST	0.7032	0.9077	0.7925	0.7522	0.8663	0.8052
hmm-2+ST	0.7321	0.9172	0.8143	0.7737	0.8789	0.8230
hmm-3+ST	0.7551	0.9172	0.8283	0.7961	0.8534	0.8237
crf	0.9024	0.9049	0.9037	0.8751	0.8227	0.8481
bi-lstm-crf	0.9430	0.9530	0.9480	0.8998	0.8527	0.8756
bi-lstm-crf-cnn	0.9244	0.9715	0.9474	0.9017	**0.8973**	**0.8995**
bi-lstm-crf-lstm	0.9465	0.9692	0.9577	**0.9108**	0.8715	0.8907

Table 6.3. Precision, recall and F1 in the NER on HTML dataset for models that incorporate all features

Conventional models like HMMs, and CRFs can become competitive with the right selection of features and a good gazetteer, however they still lose to the best neural model without features, demonstrating their inherent limitations. HMMs that employed trigrams or quadrigrams (hmm-2, hmm-3) performed better than regular HMMs. Also, we can see that the self-training strategy for HMMs improved the quality of the models significantly in all cases, boosting both precision and recall. This hints at the possibility to adapt this strategy to neural networks and boost the performance of neural models on the NER on HTML task.

The neural models also improved a little with the addition of features. The plain BI-LSTM-CRF model gets a closer performance to the models that employed neural character representations. It suggests that the LSTM and CNN character representations were able to learn at least part of the morphological features automatically in the first experiment. So, when these features are added explicitly, the differences in performance between different neural models become less noticeable.

Chapter 7

Related Work

In the last 20 years, the astonishing growth of public information in the Web has led to the development of a number of different approaches to the problem of Web data extraction. Traditionally, the task was solved by designing special purpose programs called wrappers to recognize relevant data and store records in a structured format. These early tools varied wildly relative to their degree of automation.

It was readily perceived that manual wrapper generation was a rather tedious and error prone process, unsuited for large scale operations. Wrappers tend to break frequently because they rely heavily on webpage features that can change often. So, in the late nineties, several authors advocated for wrapper induction, a technique that consists of automatically constructing wrappers from a small set of examples by identifying delimiters or context tokens that single out the desired attributes. Some remarkable wrapper induction methods are WIEN **?**, Soft Mealy **?** and STALKER **?**.

Despite being better than constructing wrappers manually, wrapper induction methods still suffered from a lack of expressive power and flexibility. These methods had trouble handling records with missing attributes or unusual structures because patterns could only be identified if they happened at least once in the examples.

Other approaches such as NoDoSE **?** and Debye **?** brought greater flexibility to wrapper induction methods by requiring a greater level of human interaction through graphical user interfaces. Web data extraction techniques often require some sort of assistance from human experts to boost accuracy. One of the main challenges in the field lies in determining an adequate trade-off between the degree of automation and the precision and recall of the data extraction tool.

To automate the task of Web data extraction completely some approaches, such as Road Runner **?**, removed entirely the need for data examples. Road Runner parses documents belonging to a same class (e.g. books on Amazon) and generates wrappers based on their similarities and differences, yielding comparable results to those obtained by wrapper induction methods. However, like previous approaches, it was unsuited for cross site extraction tasks because the learned rules were not general enough.

NLP based approaches aimed at extracting more general rules that could possibly be employed over multiple websites. RAPIER **?** is a method of rule extraction that uses information such as part-of-speech tags and semantic classes from a lexicon to derive patterns from a set of training examples. This approach is more flexible than the wrapper induction methods, however it achieves much lower rates of recall and precision.

In 2002, a survey by Laender et al. **?** made a thorough classification of the early approaches with a taxonomy based on their main technology, being them: languages for wrapper development, HTML-aware tools, NLP-based tools, Wrapper Induction Tools, Modeling-based tools and Ontology-based tools. Some noteworthy examples from this era are:

- TSIMMIS **?** and WebOQL **?**, which are special purpose languages for building wrappers.
- Road Runner **?**, XWRAP **?** and W4F **?**, which are HTML-aware tools that infer meaningful patterns from the HTML structure.
- RAPIER **?**, SRV **?**, WHISK **?**, which are NLP-based tools.
- WIEN **?**, Soft Mealy **?** and STALKER **?** which are wrapper induction methods.
- NoDoSE **?** and Debye **?**, which are semi supervised modeling based tools that require some interaction with the user by means of a graphical user interface.

In 2006, Chang et al. **?** complemented the previous surveys with semi-supervised technologies such as Thresher **?**, IEPAD **?** and OLERA **?**. They differed from supervised and unsupervised methods because they either needed only a rough description of data from users for extraction rule generation or some level of post processing that needed user attention. The survey also mentioned newer unsupervised methods such as DeLa **?**, Exalg **?** and Depta **?**.

Most of the early information extraction systems were rule-based with either manual rule description or automatic rule learning from examples, thus they suffered from a lack of flexibility when dealing with noisy and unstructured data. Huge progress in the field of statistical learning led to the development of statistical models that tried to solve this problem.

In 2008, Sarawagi ? produced a survey that classified wrappers into rule-based methods, statistical methods and hybrid models, bringing together the fields of named entity recognition, relationship extraction and information extraction. The rule based methods encompass most of the previous models. The statistical methods convert the extraction task into a token labeling task, identifying the target entities through the assignment of labels as in a typical Named Entity Recognition task. Traditionally, Hidden Markov Models ??, Linear Chain Conditional Random Fields ?, and Maximum Entropy Taggers ? have been the usual choice for linear sequence tagging models. More recently, with the advancement of Natural Language Processing and Deep Learning, neural models outperformed previous NER methods for plain text. Huang et. al. ? introduced the bidirectional Long Short-Term Memory (LSTM) model with a Conditional Random Field (CRF) output layer for NER. Ma and Hovy ? incorporated Convolutional Neural Network based character representations on top of the architecture. And Lample et. al. ? introduced LSTM based character representations.

Surveys by Ferrara et al. ?, Schulz et al. ? and Varlamov et al. ? updated the previous surveys on information extraction methods with some interesting innovations. Some examples are: the Visual Box Model ?, a data extraction system that produces a visualization of the webpage to exploit visual cues to identify data presented in a tabular form; automatic wrapper adaptation ?, a technique that tries to reduce the cost of wrapper maintenance by measuring the similarity of HTML trees and adapting wrappers to the new page structure; AutoRM ?, a method to mine records from a single webpage by identifying similar data regions through DOM tree analysis; Knowledge Vault ?, a method that combines different extraction approaches to feed a probabilistic knowledge base.

Most data extraction systems focus on extracting information from single websites and are therefore unsuited for cross website extraction tasks. Even unsupervised approaches that are domain independent, such as RoadRunner ? and EXALG ? only work well for extracting data from pages generated from a same template.

A statistical approach to unsupervised domain independent Web data extraction was described by Zhu et al ?. The 2D CRF model takes a webpage segmented into data blocks and employs a two dimensional conditional random field model to perform attribute labeling. The model was further improved ? to model record segmentation and attribute labeling as a joint task. Some of the limitations of early unsupervised methods were also tackled by ObjectRunner ? and AMBER ?. These methods work by annotating webpages automatically with regular expressions, gazetteers and knowledge bases. They can rectify low quality annotations and even improve the annotators by exploring regular structures in the DOM during the record segmentation phase.

Web data extraction methods have undoubtedly improved extraordinarily, but as pointed by Schulz et al. **?**, it is difficult to compare the results achieved by competing tools, and many seem to rely excessively on heuristic methods. In that regard, the recent advancements in sequence taggers may provide more robust and flexible extraction tools.

Chapter 8

Conclusion

Machine-learning-based sequence labeling models provide a flexible approach to Web data extraction, in contrast to more traditional methods. In simple cases, a neural named entity tagger may be sufficient to solve the entire data extraction task. In other cases, the sequence tagger remains an important part of the web data extraction system, as it performs attribute labeling on data records with accuracy and flexibility.

In this article, we compared the performance of different sequence models on the task of named entity recognition on HTML, introducing a novel dataset that is publicly available. We found that there are two components to the most successful models: neural based character representations that extract morphological features automatically, and the joint modelling of output labels.

We showed that a BI-LSTM-CRF neural network with LSTM-based character representations can be employed effectively to solve a web data extraction task, achieving an F1-score of 0.8867 with no feature engineering on the faculty listings dataset.

The effective recognition of named entities on HTML is an essential step in most general Web data extraction methods. The accuracy achieved by deep neural architectures even on webpages that are very different from the plain text for which these architectures were initially designed shows the potential for a truly flexible approach to cross domain web data extraction.

Bibliography

Abdessalem, T., Cautis, B., and Derouiche, N. (2010). ObjectRunner: lightweight, targeted extraction and querying of structured web data. *Proceedings of the VLDB ...*, 3(2):1585--1588. ISSN 21508097.

Adelberg, B. (1998). NoDoSE—a tool for semi-automatically extracting structured and semistructured data from text documents. *ACM SIGMOD Record*, 27(2):283--294. ISSN 01635808.

Arasu, A., Garcia-Molina, H., Arasu, A., and Garcia-Molina, H. (2003). Extracting structured data from Web pages. *2003 ACM SIGMOD International Conference on Management of Data*, pages 337 -- 348.

Arocena, G. O. and Mendelzon, A. O. (1999). WebOQL: Restructuring documents, databases, and webs. *Theory and Practice of Object Systems*, 5(3):127--141. ISSN 10743227.

Califf, M. E. and Mooney, R. J. (1999). Relational learning of pattern-match rules for information extraction. *Computational Linguistics*, 4:9--15. ISSN 15324435.

Caruana, R., Lawrence, S., and Giles, L. (2000). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pages 381--387, Cambridge, MA, USA. MIT Press.

Chang, C., Chang, C., Lui, S., and Lui, S. (2001). IEPAD: information extraction based on pattern discovery. *Proceedings of the 10th international conference on World Wide Web*, pages 681--688.

Chang, C.-H., Kayed, M., Girgis, M. R., and Shaalan, K. F. (2006). A Survey of Web Information Extraction Systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411--1428. ISSN 1041-4347.

Chang, C. H. and Kuo, S. C. (2004). OLERA: Semisupervised Web-data extraction with visual support. *IEEE Intelligent Systems*, 19(6):56--64. ISSN 15411672.

Chinchor, N. (1998). Overview of muc-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998.*

Crescenzi, V., Mecca, G., and Merialdo, P. (2001). Roadrunner: Towards automatic data extraction from large web sites. *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 109--118. ISSN 10477349.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. (2014). Knowledge vault: a web-scale approach to probabilistic knowledge fusion. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, pages 601--610. ISSN 0893-6080.

Ferrara, E. and Baumgartner, R. (2011). Automatic wrapper adaptation by tree edit distance matching. *Smart Innovation, Systems and Technologies*, 8:41--54. ISSN 21903018.

Ferrara, E., De Meo, P., Fiumara, G., and Baumgartner, R. (2014). Web data extraction, applications and techniques: A survey. *Knowledge-Based Systems*, 70:301--323. ISSN 09507051.

Freitag, D. (1998). Information Extraction from HTML: Application of a General Machine Learning Approach. *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 517--523.

Freitag, D. and McCallum, A. (2000). Information extraction with hmm structures learned by stochastic optimization. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 584--589. AAAI Press.

Freitag, D. and Mccallum, A. K. (1999). Information extraction with hmms and shrinkage. In *In Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 31--36.

Furche, T., Gottlob, G., Grasso, G., Gunes, O., Guo, X., Kravchenko, A., Orsi, G., Schallhart, C., Sellers, A., and Wang, C. (2012a). Diadem: Domain-centric, intelligent, automated data extraction methodology. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, pages 267--270, New York, NY, USA. ACM.

Furche, T., Gottlob, G., Grasso, G., Orsi, G., Schallhart, C., and Wang, C. (2012b). AMBER: Automatic Supervision for Multi-Attribute Extraction. *arXiv preprint*, 1210(5984):1--22.

Grishman, R. and Sundheim, B. (1996). Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 466--471, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hammer, J., Mchugh, J., and Garcia-molina, H. (1997). Semistructured Data : The TSIMMIS Experience. *Proceedings of the First East-European Symposium on Advances in Databases and Information Systems*, pages 1--8.

Hirsch, J. E. (2005). An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences*, 102(46):16569--16572.

Hirschman, L., Yeh, A. S., Blaschke, C., and Valencia, A. (2005). Overview of biocreative: critical assessment of information extraction for biology. *BMC Bioinformatics*, 6:S1 – S1.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735--1780. ISSN 0899-7667.

Hogue, A. and Karger, D. (2005). Thresher : Automating the Unwrapping of Semantic Content from the World Wide Web. *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 86--95.

Hsu, C. N. and Dung, M. T. (1998). Generating finite-state transducers for semi-structured data extraction from the Web. *Information Systems*, 23(8):521--538. ISSN 03064379.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Krüpl, B., Herzog, M., and Gatterbauer, W. (2005). Using visual cues for extraction of tabular data from arbitrary HTML documents. *Special interest tracks and posters of the 14th international conference on World Wide Web - WWW '05*, pages 1000----1001.

Kushmerick, N. (2000). Wrapper induction: efficiency and expressiveness. *Artificial Intelligence*, 118(1-2):15--68. ISSN 00043702.

Laender, A., Ribeiro-Neto, B. A., and S.Teixeria, J. (2002a). A brief survey of web data extraction tools. *ACM SIGMOD Record 31(2)*, pages 84--93.

Laender, A. H. F., Ribeiro-Neto, B., and da Silva, A. S. (2002b). DEByE - Date extraction by example. *Data Knowl. Eng.*, 40(2):121--154. ISSN 0169-023X.

Lafferty, J. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282--289. Morgan Kaufmann.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. *CoRR*, abs/1603.01360.

Lawrence, S., Lee Giles, C., and Bollacker, K. (1999). Digital libraries and autonomous citation indexing. *Computer*, 32(6):67–71.

Leek, T. R. (1997). Information extraction using hidden markov models.

Li, J. and Gray, R. M. (2000). *Image Segmentation and Compression Using Hidden Markov Models*. Kluwer Academic Publishers, Norwell, MA, USA. ISBN 0792378997.

Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503--528.

Liu, L., Pu, C., and Han, W. (2000). XWRAP: an XML-enabled wrapper construction system for Web information sources. *Proceedings of 16th International Conference on Data Engineering*, pages 611--621. ISSN 1063-6382.

Ma, X. and Hovy, E. H. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR*, abs/1603.01354.

McCallum, A. (2005). Information extraction: Distilling structured data from unstructured text. *Queue*, 3(9):4:48--4:57. ISSN 1542-7730.

McCallum, A., Freitag, D., and Pereira, F. C. N. (2000). Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 591--598, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Muslea, I., Minton, S., and Knoblock, C. (1999). A Hierarchical Approach to Wrapper Induction. *Proc. of the Third Annual Conf. on Autonomous Agents, ACM*, pages 190--197.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *In EMNLP*.

Rabiner, L. R. (1990). Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267--296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ramshaw, L. A. and Marcus, M. P. (1999). *Text Chunking Using Transformation-Based Learning*, pages 157--176. Springer Netherlands, Dordrecht.

Rastegar-Mojarad, M., Liu, S., Wang, Y., Afzal, N., Wang, L., Shen, F., Fu, S., and Liu, H. (2018). Biocreative/ohnlp challenge 2018. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '18, pages 575--575, New York, NY, USA. ACM.

Ribas, S., Ribeiro-Neto, B., de Souza e Silva, E., Ueda, A. H., and Ziviani, N. (2015). Using reference groups to assess academic productivity in computer science. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 603--608, New York, NY, USA. ACM.

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition. ISBN 0408709294.

Sahuguet, A. and Azavant, F. (1999). Building light-weight wrappers for legacy Web data-sources using W4F. *Proceedings of the 25th VLDB Conference*, 99:738--741.

Sarawagi, S. (2008). Information extraction. *Foundations and Trends in Databases*, 1(3):261--377. ISSN 1931-7883.

Schulz, A., Lässig, J., and Gaedke, M. (2016). Practical web data extraction: Are we there yet? — A short survey. *IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2016*, pages 562----567.

Shi, S., Liu, C., Shen, Y., Yuan, C., and Huang, Y. (2015). AutoRM: An effective approach for automatic Web data record mining. *Knowledge-Based Systems*, 89:314--331. ISSN 09507051.

Soderland, S. (1999). Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1):233--272. ISSN 0885-6125.

Sundheim, B. M. (1991). Overview of the third message understanding evaluation and conference. In *Proceedings of the 3rd Conference on Message Understanding*, MUC3 '91, pages 3--16, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tjong Kim Sang, E. F. and De Meulder, F. (2003a). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142--147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tjong Kim Sang, E. F. and De Meulder, F. (2003b). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142--147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Varlamov, M. I. and Turdakov, D. Y. (2016). A survey of methods for the extraction of information from Web resources. *Programming and Computer Software*, 42(5):279--291. ISSN 0361-7688.

Wang, J. and Lochovsky, F. H. (2003). Data extraction and label assignment for web databases. *Proceedings of the twelfth international conference on World Wide Web - WWW '03*, page 187. ISSN 00320862.

Zhai, Y. and Liu, B. (2005). Web data extraction based on partial tree alignment. *Proceedings of the 14th international conference on World Wide Web - WWW '05*, page 76. ISSN 10414347.

Zhu, J., Nie, Z., Wen, J.-R., Zhang, B., and Ma, W.-Y. (2005). 2d conditional random fields for web information extraction. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 1044--1051, New York, NY, USA. ACM.

Zhu, J., Nie, Z., Wen, J.-R., Zhang, B., and Ma, W.-Y. (2006). Simultaneous record detection and attribute labeling in web data extraction. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 494--503, New York, NY, USA. ACM.

Appendix A

HTML sentence segmenter

Lorem ipsum dolor.