Named Entity Recognition on HTML as a web data extraction subtask

JoÃ£o Mateus de Freitas Veneroso
Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brazil
jmfveneroso@gmail.com

Berthier Ribeiro-Neto
Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brazil
berthier.dcc.ufmg@gmail.com

ABSTRACT

Reliable researcher affiliation data is necessary to allow enquiring about international research group productivity and publication patterns. Public bibliographic databases such as DBLP and Google Scholar hold invaluable data about the academic environment. However, the researcher affiliation information is frequently missing or outdated. We propose a statistical data extraction method to acquire affiliation information directly from university websites and solve the name extraction task in general. Previous approaches to web data extraction either lack in flexibility, because wrappers do not generalize well on cross website tasks, or they lack in precision, because domain agnostic methods neglect useful properties of this particular application domain. Our statistical approach solves the name extraction task with a framework that incorporates both textual and structural features to yield an outstanding tradeoff between generality and precision. We conducted experiments over a collection of 152 faculty web pages in multiple languages from universities in 49 countries and obtained 94.40% precision, 97.61% recall and 0.9597 F-measure at the extraction task.

1 INTRODUCTION

Web information extraction is the task of automatically extracting structured information from unstructured or semi-structured web documents. The input usually consists of web documents containing a number of predetermined entities organized in a similar manner and the information extraction task consists of identifying these entities and organizing them according to a template. In this context, named entity recognition (NER) is a subtask that aims to detect named entities in the text and classify them into predetermined categories such as person names, locations or organizations.

Once we are able to reliably detect relevant entities, we may employ an information extraction algorithm to extract structured data. In this paper, we investigate different NER approaches from the perspective of web information extraction.

HTML documents most often lie in between the structured / unstructured data paradigm, which means that authors take quite a casual approach regarding formal structure. However, DOM hierarchy, element disposition, class names, and other features related to the document structure and indirectly associated with the data itself can be valuable information in the task of identifying entities and determining relationships. Yet we cannot expect these features to be completely constrained by an underlying pattern. Organization patterns tend to follow some guidelines but are in no way subject to strict rules. That is why classical information extraction systems such as automatic wrapper generators usually do not translate well across different websites. Contrastingly, statistical based approaches can be much more flexible.

NER systems are frequently trained and tested on news corpora such as the dataset introduced at the Language-Independent Named Entity Recognition Shared Task at CONLL-2003 [**?**]. State-of-the-art approaches have recently achieved F1-scores of 90.10 [**?**], 90.94 [**?**], and 91.21 [**?**] in this task using different combinations of bidirectional LSTM layers, word and character representations.

Even if focusing on web documents, records also may contain pure text that needs appropriate IE algorithms to extract the inherent data. Most WDE algorithms certainly can exploit the semi-structured nature of web documents, but can surely not restrict ourselves to that, because deeper processing is necessary. When looking through the algorithms and searching for the Swiss army knife, most algorithms can be ruled out for some incapability restricting the algorithms strongly. B.

NER on HTML poses a different type of challenge, because web pages are often very different from the plain text found in news corpora. Named entities may be present inside tables, lists, graphs or other types of visual elements that provide little to no textual information that could give hints about the semantic category of a word. Systems trained on news or similar corpora do not translate well to the web NER task without retraining, since in this task we cannot profit as much from contextual hints or transfer learning approaches such as word embeddings. Besides, deep learning approaches can require a lot of data to reach peak performance.

To test different NER approaches from the perspective of web information extraction, we will explore the task of person names extraction from university faculty websites. Researcher affiliation is often missing from many entries in public databases such as DBLP [1] and the display of information varies significantly between different university websites, so this task can provide a good measure of the expected performance and data need for other web information extraction tasks. Also, with the lack of a large amount of training data, we show that feature engineering and resources such as a gazetteer may be invaluable.

2 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 information 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

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

heavily on web page 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 tradeoff 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 semisupervised 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 in Natural Language Processing and Deep Learning, neural models outperformed previous NER methods. Huang [**?**] introduced the bidirectional Long Short-Term Memory (LSTM) model with a Conditional Random Field (CRF) output layer for NER. Lample [**?**] incorporated Convolutional Neural Network based character representations on top of the architecture. And Ma [**?**] introduced LSTM based character representations. Both obtaining improvements with systems that are less reliant on feature engineering.

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 web page 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 web page 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 thus are 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 the same underlying template. A statistical approach to unsupervised domain independent web data extraction was described by Zhu et al [**?**]. The 2D CRF model takes a web page segmented into data blocks and employs a two dimensional conditional random field model to perform attribute labelling. The model was further improved in [**?**] 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.

In summary, 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 certain heuristics. In that regard, the impressive advancements in NLP may provide for more robust and flexible extraction tools.

3 NAMED ENTITY RECOGNITION MODELS

Most web data extraction systems rely on hand crafted rules or gazetteers to perform attribute annotation. Machine learning approaches to NER can improve annotations of more complex entities and even manage entity detection without resorting to a gazetteer. We explored multiple approaches to the Named Entity Recognition problem in the context of a web information extraction task. We first introduce two traditional approaches, namely: Hidden Markov Models and Linear Chain Conditional Random Fields. And further we explore the neural architectures.

3.1 Hidden Markov Models

A Markov Model is a stochastic model that computes the most probable sequence of states given a limited set of observable states $\S = \{s_1, s_2, ..., s_n\}$. The Hidden Markov Model (HMM) differs from the Markov Model in that it does not observe the states directly, but rather a probabilistic function of those states. For example in NER, the words are observed, however the Named Entity labels associated with these words are not. Formally, we want to compute the most probable sequence of labels $Y = \{y_1, y_2, ..., y_n\}$ for a sequence of observed tokens $X = \{x_1, x_2, ..., x_n\}$.

$$Y* = \arg\max_{Y} P(Y|X) \tag{1}$$

With Bayes theorem, we can write $P(Y|X)$ as:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{2}$$

Since $P(X)$ is the same for all label sequences Y, we can simply maximize:

$$P(Y|X) \propto P(X|Y)P(Y) \tag{3}$$

A HMM makes two assumptions. First, the probability of being in a given state depends only on a fixed number of previous states. That is $P(y_i|y_{i-1}x_{i-1}, y_{i-2}x_{i-2}, ..., y_1x_1) = P(y_i|y_{i-1}, y_{i-2}, ..., y_{i-k})$. In fact, we can get much better results on the NER task by looking at trigrams or quadrigrams ($k = 2$ or $k = 3$) instead of bigrams as a regular HMM. Some label assignments are highly improbable, such as single token named entities separated by a common word, and these kinds of patterns can be caught by a higher order HMM. Second, the probability of a word depends only on its assigned label $P(x_i|y_{i-1}x_{i-1}, ..., y_1x_1) = P(x_i|y_i)$. With these assumptions, the probability $P(Y|X)$ can be approximated by the expression:

$$P(Y|X) \propto \prod_{i=k+1}^{n} P(y_i|y_{i-1}, y_{i-2}, ..., y_{i-k})P(x_i|y_i) \tag{4}$$

The probabilities can be calculated through maximum likelihood estimation from the relative frequencies of labels and features in the corpus. 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.

HMM based taggers have been succesfully apllied in many NLP and IE tasks [**? ? ?**]. They are incredibly fast to train and 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.

3.1.1 Self training. In the particular case of NER on HTML, there are useful features related to the HTML structure that can be used to identify named entities. In a particular website, named entities tend to occur inside the same HTML tags with the same CSS classes. This fact can be used as an additional piece of evidence in the HMM tagger. However, if we train the HMM for a particular set of websites, and then we feed it with new websites, almost surely there will be small correlation between the HTML features estimated in the training set and the distribution of these features in the new websites, simply because websites use very different templates. Therefore, we propose to use a self training strategy to train these HTML features. This is only possible because the HMM can be trained very quickly. The self training strategy is implemented like this:

- Train the HMM in a given training set without the HTML features.
- Compute labels for the new websites with the trained HMM.
- Use these computed labels as a proxy for the actual labels in the new websites and estimate HTML feature frequencies only in the new websites.
- Recompute the labels using the HTML features.

This adds very little overhead to the original model while improving precision and recall modestly.

3.2 Linear Chain Conditional Random Fields

A Linear Chain Conditional Random Field (CRF) is the discriminative analog to the HMM, it was first introduced by [**?**]. It is a distribution $P(Y|X)$ that takes the form:

$$P(Y|X) = \frac{1}{Z(x)} \prod_{t=1}^{T} exp\left(\sum_{k=1}^{K} \theta_k f_k(y_{t-1}, y_t, X)\right) \tag{5}$$

where θ is the parameter vector that we are going to learn, $f_k(y_{t-1}, y_t, X)$ are feature functions over the current timestep t_y, the previous timestep y_{t-1}, and the observation vector X. And the partition function $Z(x)$, takes the form:

$$Z(x) = \sum_{Y} \prod_{t=1}^{T} exp\left(\sum_{k=1}^{K} \theta_k f_k(y_{t-1}, y_t, x)\right) \tag{6}$$

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 like in the case of HMMs.

CRFs are more general than HMMs because the transitions from y_{t-1} to y_t can also depend on the vector of observations X. This flexibility of feature functions allow for a wide range of possibilities. Recently, CRFs have been succesfully employed as the output layer in complex neural architectures bringing improvements over models that classify labels independently.

3.3 Neural Models

Recurrent neural networks (RNN) have been succesfully employed on numerous NLP tasks such as language modelling, POS tagging, speech recognition and NER. Different from feedforward neural networks, RNNs can retain information in their internal state, making them more suitable for processing sequences, and consequently for solving text related tasks. In NER, the network outputs one label for each input vector as Figure YYY.

<INSERT FIGURE YYY>

Although RNNs are theoretically capable of learning long term dependencies, in practice it might be difficult. Long short term memory networks (LSTM) were introduced by Hochreiter and Schmidhuber [**?**] with this problem in mind and have been popularized since them. LSTM cell implementations vary slightly in the literature. Our implementation has the following definition:

$$\Gamma_i^{<t>} = \sigma(W_i[x^{<t>}, h^{<t-1>}] + b_i)$$
$$\Gamma_f^{<t>} = \sigma(W_f[x^{<t>}, h^{<t-1>}] + b_f)$$
$$c^{<t>} = \Gamma_f^{<t>} \circ c^{<t-1>} + \Gamma_i^{<t>} tanh(W_{c,x}x^t + W_{c,a}a^{t-1} + b_c)$$
$$\Gamma_o^{<t>} = \sigma(W_o[x^{<t>}, h^{<t-1>}] + b_o)$$
$$h^{<t>} = \Gamma_o^{<t>} \circ tanh(c^t)$$

where σ is the logistic sigmoid function. Γ_i, Γ_f, and Γ_o are the input, forget and output gates, respectively. And W with a subscript is the weight matrix corresponding to that gate. Variable $c^{<t>}$ is the cell state and $h^{<t>}$ is the hidden state, with the vector $[x^{<t>}, h^{<t-1>}]$ being formed by concatenating the current input vector $x^{<t>}$ and the hidden vector from the previous timestep $h^{<t-1>}$. A visual description of our LSTM cell is provided in figure XXX. This implementation differs from the LSTM cell described in Huang et al. [**?**] in that the input gate Γ_i and the forget gate Γ_f do not take input from the previous cell state $c^{<t-1>}$ and the output gate does not take input from the current cell state $c^{<t>}$. The cell state and the control gates are what makes the LSTM cell better at modelling long term dependencies in comparison to a regular RNN.

XXX Insert figure explaining equations XXX

In NER related tasks, both past and future words are important when deciding the label at time t, but a regular LSTM network only takes past features in consideration. A bidirectional LSTM solves this problem by stacking two regular LSTMs, and feeding them with observations in opposite directions. The first LSTM receives the forward states and the second LSTM receives the backward states. The hidden states from both networks can be concatenated at each time step to produce the output label. With this architecture, Bidirectional LSTMs have information from all input features in the sequence at any given timestep.

3.3.1 LSTM-CRF. Huang et al. [**?**] proposed a bidirectional LSTM with a CRF layer (LSTM-CRF) on the output to tackle the sequence tagging problem. The CRF layer jointly decodes labels for the whole sentence instead of predicting each label individually. This architecture achieved an F1 score of 90.10 in the ConL2003 dataset, in contrast to 85.17 for a bidirectional LSTM without the CRF layer. In our experiments, the LSTM-CRF architecture used a bidirectional LSTM with 100 hidden states, no peepholes and an 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.

3.3.2 LSTM-CRF + CNN character representations. Ma and Hovy [**?**] proposed adding a convolutional neural networks (CNN) layer on top of a bidirectional LSTM-CRF to encode character-level information. Those character representations are combined with word level representations and then fed to the Bidirectional LSTM. This architecture possesses the capacity to learn morphological token features that are extremely useful on the NER task, since similar named entities often present morphological similarities. This improvement obtained an F1 score of 91.21 at the ConLL2003 dataset. In our experiments, the LSTM-CRF architecture with CNN character representations used 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 had 30 dimensions that were randomly initialized.

3.3.3 LSTM-CRF + LSTM character representations. Lample et al [**?**] suggested using an a bidirectional LSTM to model character-level representations on a bidirectional LSTM-CRF. Combining both the forward and backward LSTM representations to form the character representation. This character representation is also combined with a word representation. The forward LSTM is expected to be a better representation of the suffix of a token, and the backward LSTM is expected to be a better representation of the prefix of a token. This differentiates the architecture from the CNN based approach, 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 embeddings was implemented by a bidirectional LSTM with 25 hidden states, producing character representations of size 50. The character embeddings also had 30 dimensions that were randomly initialized.

3.3.4 Network training. The neural models were trained with Stochastic Gradient Descent over 50 epochs using a learning rate of 0.01 and momentum of 0.9. 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, they have a vocabulary size of 400.000 words. In the case of NER on HTML, word embeddings work as a rather satisfactory gazetteer. Named entities of a same type have similar embeddings, so with good word embeddings we

can achieve exceptional performance with little training data and without a gazetteer. Figure XXX illustrates the differences between the three neural models mentioned previously.

XXX Insert picture here explaining the models Figure 1 XXX

4 NER ON HTML DATASET

We constructed a novel dataset to evaluate the performance of multiple NER models at the web information extraction task. The task consists of finding researcher names in faculty listings from multiple web pages. This would be the first step in linking researcher profiles from university websites to their entries in public databases such as DBLP [2]. Unlike many information extraction datasets, each web page in the dataset comes from a different university, 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 name extraction from many different sources without requiring any supervision between different websites.

This task would be similar to the task of labeling authors in comments from many different sources or labeling authors in articles collected from many publishing platforms. Another similar task would be NER on tweets. Because of the character limitation present on tweets, we end up with constraints comparable to the ones found in the NER on HTML task, so we expect similar systems to have comparable performances on both of these tasks.

We collected 145 computer science faculty pages from 42 different countries in multiple languages, although the English version was prefered when it was available. We gathered faculty web pages randomly in proportion to the number of universities in each country[3]. 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 50 tokens were also split according to punctuation.

A proper HTML segmenter poses many challenges by itself as discussed for example in [**?**]. However, we wanted to evaluate models without relying on any sophisticated data record segmentation system. In many cases, entity annotation may precede the segmentation phase in web data extraction methods, so annotators have to be able to work with raw HTML data. Because of the type of documents present in the dataset, many sentences contained only one or two tokens, while other documents contained long texts. Finally, all tokens were tagged using the IOB scheme put forward by Ramshaw and Marcus [**?**] with a modification. O stands for Outside, B-PER is the beginning of a person name, I-PER inside a person name and PUNCT is any punctuation sign other than ".". Many name representations contain punctuation signs and the PUNCT label helps predicting label transitions properly, especially when using simpler algorithms like HMMs. A name for example never ends with a punctuation sign, but it may be split by a punctuation sign.

Data file	Documents	Sentences	Tokens	Names
Training	92	28958	124984	6382
Validation	28	9117	39748	2258
Test	25	5790	26995	1949

Table 1: Number of HTML pages, sentences and tokens in each data file

4.1 Data

The dataset was divided in a training, validation and test set. Table 1 contains a description of the data files. The validation set was used in the early stopping validation strategy on neural model and CRF training, while results were evaluated by comparing results in the test set.

4.2 Features

Twelve categorical features were also present with each token in the dataset. They are presented in table 2.

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
9	URL
10	Is capitalized
11	HTML tag + parent
12	CSS class

Table 2: Features present in the bla dataset

The unaccented lowercase token was used as the key for the Glove-100 embedding lookup. A gazetteer was constructed from a researcher list extracted from DBLP with 1.595.771 names. Feature 2 represents an exact match of a sequence of tokens to any of the 1.595.771 names, 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 in university websites. Features 6 to 9 represent a simple regular expression match to an email, number, honorific or URL, respectively. Feature 10 represents if a word is capitalized.

Feature 11 represents the HTML enclosing tag and its parent concatenated. Feature 12 represents all CSS classes concatenated. These features are not very useful in a general sense, because every HTML document has a different format, so only because a named entity occurs inside a given HTML tag in a document we cannot say it is more likely to be the case in other documents. However, these features can be trained with the self-training strategy for HMMs

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

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

described in section 3.1.1. For all other models these features were ignored.

5 EXPERIMENTS

We conducted two experiments to evaluate the best models for named entity recognition on HTML in the context of web data extraction. In the first experiment, we used the whole training set (90 documents). In the second experiment, the training set was split into five non overlapping sets with only 18 documents and the validation set was reduced to 5 documents, but the test set was kept unchanged. The goal of this second experiment was to evaluate models when training data is scarce, since labelling data can be a very time consuming task.

Evaluation of model performance in this task was measured with the precision, recall and F1 scores (Van Rijsbergen, 1975). 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:

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{7}$$

Feature	Description
hmm-1	Regular HMM
hmm-2	HMM with $k = 2$
hmm-3	HMM with $k = 3$
crf	Linear chain conditional random fields
lstm-crf	LSTM-CRF model [**?**]
lstm-crf-cnn	LSTM-CRF with CNN character representations [**?**]
lstm-crf-lstm	LSTM-CRF with LSTM character representations [**?**]

Table 3: Features present in the bla dataset

Named entities were only considered to be correct if they were a complete match of the corresponding entity in the dataset. The models evaluated are presented in Table 3. We evaluated the models using the complete set of features, presented in table 2 and no features besides word embeddings for neural models and CRFs, and unaccented words for HMMs. To represent models with the complete set of features we added a "+F" suffix to the model name. Additionally, HMMs were evaluated with and without the self training strategy. To represent HMMs that use the self training strategy we added the suffix "+ST" to the model name. Results were reported for the validation and test sets in experiment 1 and only for the test set in experiment 2. In the second experiment, precision, recall and F1 scores were averaged over the five runs. The standard deviation is presented next to the measures.

5.1 Experiment 1: complete dataset

A named entity was only considered correct when it was an exact match of the corresponding entity in the data file. Models were trained on each of the five sets separately and tested in the same test set, and each model was trained on each of the five sets and evaluated on the same validation and test sets.

Model	Validation			Test		
	Precision	Recall	F1	Precision	Recall	F1
hmm-1	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
hmm-2	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
hmm-3	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
crf	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
lstm-crf	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
lstm-crf-cnn	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
lstm-crf-lstm	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
hmm-1+F	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
hmm-2+F	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
hmm-3+F	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
hmm-3+F+ST	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
crf+F	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
lstm-crf+F	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
lstm-crf-cnn+F	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527
lstm-crf-lstm+F	0.5282	0.9226	0.6718	0.5333	0.8409	0.6527

Table 4: Precision, recall and F1 for the complete dataset

Model	Test		
	Precision	Recall	F1
hmm-1	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
hmm-2	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
hmm-3	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
crf	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
lstm-crf	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
lstm-crf-cnn	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
lstm-crf-lstm	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
hmm-1+F	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
hmm-2+F	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
hmm-3+F	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
hmm-3+F+ST	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
crf+F	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
lstm-crf+F	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
lstm-crf-cnn+F	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)
lstm-crf-lstm+F	0.5333 (0.111)	0.8409 (0.111)	0.6527 (0.111)

Table 5: Precision, recall and F1 for the small dataset

5.2 Experiment 2: reduced dataset

were trained

consisted of training all models with the whole training set (90 documents) and the second experiment trained them on five subsets with 18 documents

Table X presents the results for X models in the NER-HTML dataset. Results for each model are presented when incorporating all features and no features.

Deep learning models are better blabla.

HMMs are competitive but demand feature engineering blabla.

Token morphology is necessary.

Label dependency is necessary.

6 CONCLUSION

Our name extraction method achieved 94.40% precision, 97.61% recall and 0.9597 F-measure on a corpus of 152 faculty directory web pages from 49 different countries with 11782 researcher names.

The model is tuned for the particular problem of name extraction, but we believe this result can be generalized to solve other data extraction