Using Conditional Random Fields with Constraints to Train Support Vector Machines
Locating and Parsing Bibliographic References

Sebastian Lindner
Institute of Computer Science, University of Würzburg, Würzburg, Germany

Keywords: Bibliography, Classification, Conditional Random Fields (CRFs), Constraint-based Learning, Information Extraction, Information Retrieval, Machine Learning, References Parsing, Semi-supervised Learning, Support Vector Machines (SVMs).

Abstract: This paper shows how bibliographic references can be located in HTML and then be separated into fields. First it is demonstrated, how Conditional Random Fields (CRFs) with constraints and prior knowledge about the bibliographic domain can be used to split bibliographic references into fields e.g. authors and title, when only a few labeled training instances are available. For this purpose an algorithm for automatic keyword extraction and a unique set of features and constraints is introduced. Features and the output of this Conditional Random Field (CRF) for tagging bibliographic references, Part Of Speech (POS) analysis and Named Entity Recognition (NER) are then used to find the bibliographic reference section in an article. First, a separation of the HTML document into blocks of consecutive inline elements is done. Then we compare one machine learning approach using a Support Vector Machines (SVM) with another one using a CRF for the reference locating process. In contrast to other reference locating approches, our method can even cope with single reference entries in a document or with multiple reference sections. We show that our reference location process achieves very good results, while the reference tagging approach is able to compete with other state-of-the-art approaches and sometimes even outperforms them.

1 INTRODUCTION

Due to the increasing number of scientific publications, there is a growing demand to search for similar works and compare new results with previous ones.

There already are certain online publishing systems and special search engines like Google Scholar[1] or CiteSeerX[2] that allow this kind of research. To support searches e.g. for specific authors, first of all the reference section has to be located within all indexed documents. After that, each reference has to be divided into a set of fields e.g. author or journal title. In the remaining part of this paper, we will therefore use the terms labeling, tagging and splitting into fields as synonyms. Because of the diversity of the content and the corresponding reference sections this process is not easy to automate.

Initial approaches to cope with this kind of data mining task e.g. the previous version of CiteSeer used rule based algorithms. CiteSeer therefore applied heuristic rules to big sets of data and became a well-known search engine for references (Bollacker et al., 1998). But the sets of rules were difficult to maintain and to adjust to other domains. In contrast to that, machine learning techniques are much more adaptable to other reference locating and labeling domains (Zou et al., 2010). Generally speaking, supervised machine learning algorithms use labeled training material to build a statistical model, which is then used to tag or label further data. First machine learning approaches used Hidden Markov Models (HMMs) for this task (Hetzner, 2008). Nowadays CRFs are most popular, because they allow the use of dependend features and joint inference over the whole reference and so achieve better results (Gao et al., 2012).

In cooperation with Springer Science+Business Media we develop the web platforms SpringerMaterials[3] and SpringerReference[4] for online document

[1] http://scholar.google.com
[2] http://citeseerx.ist.psu.edu/index
[3] http://www.springermaterials.com
[4] http://www.springerreference.com

Lindner S..
Using Conditional Random Fields with Constraints to Train Support Vector Machines - Locating and Parsing Bibliographic References.
DOI: 10.5220/0004546100280036
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval and the International Conference on Knowledge Management and Information Sharing* (KDIR-2013), pages 28-36
ISBN: 978-989-8565-75-4

publishing. Those platforms have a great amount of bibliographic data with a variety of citation styles. Because of these different citation styles, using some random part of the available references for training to label some other part of the references would not lead to a good labeling performance. However, since the content is organized in books and subject areas, training of separate models can be done with regard to this segmentation. Because the generation of labeled training data for each of this content partitions is a time consuming job, the focus of this paper lies on cases where only a few training instances are available.

Due to the limited amount of training instances, a Conditional Random Field and additional prior knowledge in form of constraints about the bibliography domain in addition to a few labeled instances are used for training. This CRF can then easily be adapted to other domains (citation styles) by generating a few new labeled instances for training and changing the constraints. That way, a new CRF model can be trained for each book or subject area. This results in a significant improvement in overall labeling accuracy.

Afterwards, we explain how the results of this reference labeling process can then be used to locate the reference section in the first place. Normally, the locating of the reference section would be the first step in extracting bibliographic references from documents. But we demonstrate how results of the reference tagging can even be of value for the reference locating process. Therefore, we reuse the features and the output of the reference labeling process in addition to the results of a Named Entity Recognition (NER) analysis, a Part Of Speech (POS) tagging step and the generation of some additional features to locate the reference section in a document with a machine learning algorithm. We compare the use of a Conditional Random Field (CRF) and a Support Vector Machine (SVM) for this task.

The results in this paper proof that the mentioned reference locating process has a very good accuracy, while the reference tagging can compete with other state-of-the-art approaches and even outperform them. Since the features and the output of the reference parsing step are used to locate the reference section in an article this step is introduced first.

So in the first part of this paper, we demonstrate how references can be parsed. Therefore, first of all an extended constraint model for CRFs is introduced. Next, all used features and constraints needed for this task are described. In addition to that, it is shown how automatically extrated constraints can be used to construct new features for learning.

In the second part of the paper, we propose a new method for the classification of HTML blocks into those containing a reference and those which do not contain a reference. Therefore, a novel feature set for two machine learning algorithms to classify these blocks is introduced. Next, the results for the reference parsing and location step are shown and compared to other previous approaches. Last but not least, we draw conclusions and propose several topics for further research.

2 REFERENCE PARSING

First of all, we formally introduce the task of reference parsing. It is to assign the correct labels $\mathbf{y} = \{y_1, y_2, \ldots y_n\}$ to tokens of an input sequence $\mathbf{x} = \{x_1, x_2, \ldots x_n\}$. Tokens are thereby generated by splitting a reference string on whitespace. For example, the token *Meier* in an input sequence should receive the label *AUTHOR*. For examples of tagged references see Table 1.

This also is a common task in other research areas like Part Of Speech tagging and semantic role labeling (Park et al., 2012). So all techniques shown in this paper can similarly be used in other fields of research as well.

2.1 CRFs with extended Generalized Expectation Criteria

Linear-chain Conditional Random Fields (CRFs) are a probabilistic framework that uses a discriminative probabilistic model over an input sequence $\mathbf{x}$ and an output label sequence $\mathbf{y}$ as shown in equation 1.

$$p_\lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} exp\left(\sum_i \lambda_i F_i(\mathbf{x}, \mathbf{y})\right) \quad (1)$$

In case of a linear chain CRF $F_i(\mathbf{x}, \mathbf{y})$ are a number of feature functions and $Z(\mathbf{x})$ is a normalization factor as described in (Sutton and McCallum, 2006).

Since the goal is to train a model with as few labeled training instances as possible, constraints are used to improve labeling results. An existing concept of Conditional Random Fields (Mann and McCallum, 2010) is therefore extended to allow more complex constraints.

Given a set of training data $T = \left\{\left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \ldots, \left(\mathbf{x}^{(m)}, \mathbf{y}^{(m)}\right)\right\}$ and an additional set of unlabeled references U, the goal of training a Conditional Random Field with extended constraints is to learn the parameters λ_i by maximizing equation 2. While training, these additional unlabeled references

Table 1: Examples of tagged references.

(a) <**author**>N. Benvenuto and F. Piazza.</**author**><**title**>On the Complex Backpropagation Algorithm.</**title**> <**journal**>IEEE Transactions on Signal Processing,</**journal**><**volume**>40(4)</**volume**> <**pages**>967-969,</**pages**><**date**>1992.</**date**>

(b) <**author**>C. Jay, M. Cole, M. Sekanina, and P. Steckler.</**author**> <**title**>A monadic calculus for parallel costing of a functional language of arrays.</**title**> In <**editor**>C. Lengauer, M. Griebl, and S. Gorlatch, editors,</**editor**> <**booktitle**>Euro-Par'97 Parallel Processing,</**booktitle**><**volume**>volume 1300</**volume**> of LNCS, <**pages**>pages 650-661.</**pages**><**publisher**>Springer-Verlag,</**publisher**><**date**>1997.</**date**>

are used to calculate a distance between the current labeling results on these unlabeled references and the provided constraints.

$$\Theta(\lambda, T, U) = \sum_i log p_\lambda \left(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}\right) - \frac{\sum_i \lambda_i^2}{2\sigma^2} - \delta D(q || \hat{p_\lambda}), \tag{2}$$

where q is a given target distribution and

$$\hat{p_\lambda} = p_\lambda(y_k | f_1(\mathbf{x}, k) = 1, \ldots, f_m(\mathbf{x}, k) = 1) \tag{3}$$

with all $f_i(\mathbf{x}, k)$ being feature functions that only depend on the input sequence. The first term in equation 2 is the log-likelihood used by most CRF implementations for training and the second term is a Gaussian prior for regularization. $D(q||\hat{p_\lambda})$ is a function to calculate a distance between the provided target probability distribution and the distribution calculated with the help of the unlabeled training data. To penalize differences in these distributions, δ is thereby used as a weighting factor (Lafferty et al., 2001). This is the part of the equation that takes the constraints into account during training.

In this paper we compare three different distance metrics against each other. The first one is the Kullback-Leibler (KL), the second the L_2 distance and the third one is a range-based version of the L_2-distance. In the last case, target probability ranges can be specified instead of one specific value. If the calculated probability then falls within this range, the calculated distance is 0.

As implementation we used MALLET (McCallum, 2002), which implements CRFs with Generalized Expectations (GE) as described in (Mann and McCallum, 2010). Instead of restricting the target probability distribution to the use of only one feature (see equation 4), we extended this functionality to support multiple features as well (see equation 3). This way more complex constraints can be defined, which in return improved our labeling results.

$$\hat{p_\lambda} = p_\lambda(y_k | f_i(\mathbf{x}, k) = 1) \tag{4}$$

An example of such a constraint could be that if a name appears at the beginning of a document, it should be labeled *AUTHOR*.

2.2 CRF Features for Reference Parsing

The following enumeration briefly lists the different categories of features used by the reference tagging process. We therefore use a set of binary feature functions similar to the ones used by ParsCit (Councill et al., 2008).

1. **Word based Features.** Indicate the presence of some significant predefined words like *'No.'*,*'et al'*, *'etal'*, *'ed.'* and *'eds.'*.
2. **Dictionary based Features.** Indicate whether a dictionary contains a certain word in the reference string like for author first- and lastnames, months, locations, stop words, conjunctions and publishers.
3. **Regular Expression based Features.** Indicate whether a word in the reference string matches a regular expression like ordinals (e.g. 1st, 2nd...), years, paginations (e.g. 200-215), initials (e.g. J.F.) and patterns that indicate whether a word contains a hyphen, ends with punctuation, contains only digits, digits or letters, has leading/trailing quotes or brackets or if the first char is upper case.
4. **Keyword Extraction based Features.** Indicate whether a word is in a list of previously extracted keywords for a certain label.

For the keyword extraction process the *GSS* measure (Sebastiani, 2002) was used. *GSS* is thereby defined as

$$GSS(t_k, c_i) = p(t_k, c_i) \cdot p(\overline{t_k}, \overline{c_i}) - p(t_k, \overline{c_i}) \cdot p(\overline{t_k}, c_i), \tag{5}$$

where $p(\overline{t_k}, c_i)$ for example is the probability that given a label c_i, the word t_k does not occur under this label.

Table 2 shows a brief excerpt of the extracted keywords. As one can see, many useful keywords could be automatically extracted for the labels.

In addition to the already mentioned features, a window feature with size 1 and a feature that indicates the position of each word in the reference string was used for training the CRF. Window feature in

Table 2: Automatically extracted keywords with their corresponding label.

Keyword	Label
Proceedings	BOOKTITLE
Conference	BOOKTITLE
pp	PAGES
Press	PUBLISHER
ACM	JOURNAL
Journal	JOURNAL
University	INSTITUTION
Proc	BOOKTITLE
Vol	VOLUME

this case means that features of one token (word) are transferred to its neighbors.

2.3 CRF Constraints for Reference Parsing

In order to use Conditional Random Fields with extended Generalized Expectations as shown in equation 2, constraints must be defined for the reference labeling task. The following enumeration shows the types of constraints used and names a few examples of them:

1. **Constraints that depend on a single feature function** e.g. extracted keywords for a label should be tagged with that exact label or words that match a year pattern should definitely be labeled *YEAR*
2. **Constraints that depend on multiple feature functions** e.g. words at the beginning of the reference string and are contained in the dictionary of names should be labeled *AUTHOR*. At the end of the reference string they should be labeled *EDITOR*. Also a number right to word 'No.' should recieve the label *VOLUME*.

An extensive list of all used constraints, their assigned probability distributions and other extracted keywords can be found in our previous paper (Lindner and Höhn, 2012).

3 REFERENCE LOCATING

Existing algorithms for extracting information from HTML usually depend on the Document Object Model (DOM) structure of the document. A related field of work that has recently been studied by several researches is the automatic extraction of records from web pages. Such records could for example be the items in an online shopping cart. Most of them are based on identifying similar DOM tree structures in the web page. Using visual information, tree matching and tree alignment, Zhai and Liu were able to successfully extract such structured records (Zhai and Liu, 2006). Fontn et al. even extended these approaches to extract sub-records (Fontan et al., 2012).

On the other hand, data mining algorithms on scanned documents mostly analyze geometric features and layout. Either a top-down approach to recursively divide the whole document into smaller sections e.g. using X-Y cut (Ha et al., 1995), or a bottom-up approach to cluster more and more small components together (Jain and Yu, 1998) is used. Both of these processes terminate when some criteria is met.

In this part of the paper, two machine learning techniques for the identification of a reference section in an HTML document are introduced. Therefore, the performances of a Conditional Random Field (CRF) and a Support Vector Machine (SVM) are compared against each other. First of all the HTML is partitioned into units belonging together (blocks) based on the HTML DOM (Document Object Model) and visual layout information. After that, features for each of these blocks are extracted and used for classification into reference and non-reference blocks.

Zou et al. followed a similar approach by first extracting zones from HTML and then training a SVM with features from these zones (Zou et al., 2010). Since we used our own dataset from SpringerReference, there are some different preconditions in comparison to the approach of Zou et al. They used the MEDLINE 2006 database for reference section locating experiments. In their approach the output of the SVM and the corresponding confidence values for the classification results are used in an equation to determine the best candidates for the first and last reference entry (see equation 6).

$$[t_F^*, t_L^*] = \arg\max_{t_F, t_L} \prod_{t_F \leq i \leq t_L} P(c_i = R) \prod_{0 \leq j < t_F, t_L < j \leq N} (1 - P(c_j = R)), \quad (6)$$

where t_F and t_L are the locations of the first and last reference, respectively. N ist the total number of child zones and $P(c_k = R)$ is the probability of the k^{th} child being a reference zone (Zou et al., 2010).

Because not all of our documents even have a reference section or have multiple reference sections in one single document, their approach is not directly applicable. Their 'repair step' can also not be usefully applied if a document only contains one reference entry. In addition to that, their approach uses features like the left and top position of extracted zones for training. Since some of our articles have multiple reference sections these features can not be transferred to our domain. If we had a further reading section

Table 3: (a) Example of a reference section, (b) HTML source code of the reference section, (c) Named Entity Recognition results for a reference entry, (d) Part Of Speech tagging results for a reference entry (e) Part Of Speech tagging results for a random paragraph, (f) CRF tagging results for a random paragraph

(a) 1. McDuffie, H. H., Dosman, J. A., Semchuk, K. M., Olenchock, S. A., & Sentihilselvan, A. (1995). *Agricultural health and safety: Workplace, environment and sustainability*. Boca Raton, FL: Lewis.
2. Messing, K. (1998). *One-eyed science: Occupational health and women workers*. Philadelphia: Temple University Press.

(b)
```
<ul>
  <li>1. McDuffie, H. H., Dosman, J. A., Semchuk, K. M., Olenchock, S. A., &amp;
   Sentihilselvan, A. (1995). <i>Agricultural health and safety: Workplace,
   environment and sustainability</i>. Boca Raton, FL: Lewis.</li>
  <li>2. Messing, K. (1998). <i>One-eyed science: Occupational health
   and women workers</i>. Philadelphia: Temple University Press.</li>
</ul>
```

(c) McDuffie, H. H., Dosman, **J. A.**, **Semchuk_PERSON** K. M., **Olenchock_PERSON**, S. A., & Sentihilselvan, A. **(1995)_DATE**. *Agricultural health and safety: Workplace, environment and sustainability* **Boca Raton_LOCATION**, **FL_LOCATION**: **Lewis._PERSON**

(d) McDuffie_NNP, H._NNP H._NNP, Dosman_NNP, J._NNP A._NNP, Semchuk_NNP, K._NNP M._NNP, Olenchock_NNP, S._NNP A._NNP, &_CC Sentihilselvan_NNP, A._NN (1995)_CD. *Agricultural*_NNP *health*_NN *and*_CC *safety*_NN: *Workplace*_NNP, *environment*_NN *and*_CC *sustainability*_NN. Boca_NNP Raton_NNP, FL_NN: Lewis_NNP.

(e) The_DT appearance_NN of_IN any_DT coast_NN, the_DT Arctic_NNP included_VBD, depends_VBZ on_IN the_DT alterations_NNS that_WDT have_VBP occurred_VBN to_TO the_DT geologic_JJ base_NN it_PRP inherited_VBD.

(f) The_TITLE appearance_TITLE of_TITLE any_TITLE coast_TITLE, the_TITLE Arctic_TITLE included_TITLE, depends_NOTE on_NOTE the_NOTE alterations_NOTE that_NOTE have_NOTE occurred_NOTE to_NOTE the_NOTE geologic_NOTE base_NOTE it_NOTE inherited_NOTE.

after the first paragraph in a long document, the location information of this block would not be different in comparison to other normal blocks. So the position of a block with a reference in the document is not significant for the learning process.

3.1 Extraction of Blocks

Table 3 shows an example of two references (a) and their corresponding source code (b). As one can see in this case an unordered list is used to arrange the two references. But there are many other ways to structure the same information, for example by using a table `<table>`, a new paragraph `<p>` for each entry or just line breaks `<br>` between text.

In order to get blocks of content belonging together, one can not just use the DOM and extract all elements that have no child elements. For example in Table 3 (b) the title of the two citations is italic. Extracting elements without children as blocks would lead to very small blocks, which have too few relevant information for a correct classification.

To cope with that, a bottom-up approach is used to successively merge inline DOM nodes. These are nodes that do not introduce line breaks. For this we use the HTML parser jsoup[5]. This parser has a predefined lists of tags that introduce line-breaks like `<div>` or `<p>` and those, which are inline elements like `<i>`, `<span>` or simple text. In the example of Table 3 (b) the italic titles would be merged with the surrounding rest of the reference string, because the text and the italic HTML node are inline elements. The `<li>` element however, introduces a line-break. So each of these two references would end up in an own block. Figure 1 shows examples of extracted blocks. If the whole document basically only contains text and is structured by `<br>` and inline elements that use Cascading Style Sheet (CSS) information, the extraction of the blocks can be quite difficult.

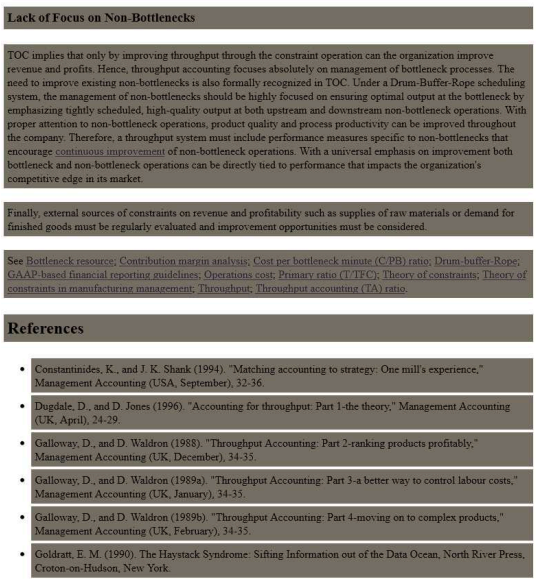

Figure 1: Examples for extracted blocks.

[5]http://jsoup.org/

3.2 Reference Block Classification

First of all, the features for the classification process are introduced. Afterwards it is shown, how two different machine learning techniques can be used to classify blocks into those, which contain a reference and those which do not contain a reference. Therefore, a SVM approach is presented and then compared to a CRF for classification.

Initially, a set of features is generated by other machine learning modules. The Stanford Named Entity Recognizer (Finkel, 2007) is used to extract persons, dates, locations and organizations. This software in turn utilizes a Conditional Random Field to extract these entities. An example output for a reference block is given in Table 3 (c). In this table the corresponding label to a word is appended by _LABEL. In comparison to blocks that do not belong to a reference section, the number of extracted entities is much higher. So the number of entities is a clear indicator for a reference entry.

Next the Stanford Log-linear Part-Of-Speech Tagger (Toutanova et al., 2003) is applied to each block. This uses a Cyclic Dependency Network for its labeling process. Table 3 (d) and (e) show the output of this tagging process for a reference block and a random other block from our corpus. Almost all words in the reference string are tagged as NN (Noun, singular or mass) or NNP (Proper noun, singular). In the case of a random non-reference paragraph however, the number of conjunctions or verbs is much higher like VBD (Verb, past tense), VBZ (Verb, 3rd person singular present), VBP (Verb, non-3rd person singular present), VBN (Verb, past participle) or VBG (Verb, gerund or present participle). So the number of verb forms in a block is another indicator for classification. Other labels that appear in this table are: CC (Coordinating conjunction), DT (Determiner), IN (Preposition or subordinating conjunction), JJ (Adjective).

In addition to these, we use the output of our own reference parser as input for classification. While it is able to correctly label most reference sections, its output is very different for a non-reference section as shown in Table 3 (f). All words are labeled as titles or notes. Not even one author is found in the string. So the output of the CRF reference tagger can also be of value to tell a reference and a non-reference section apart.

On top of the machine learning outputs, a number of other features are used for classification. All features generated by the reference parsing process are reused in this reference locating step as well. Meaning features for dictionaries, important keywords, regular expressions and so on.

Furthermore, we used the following features for each block:

- text length
- number of words
- average word length
- number of punctuation characters
- average number of punctuation character (normalized by the number of words)

The number of occurrences of each of the previously mentioned features is then used for training. To take the position of these features in the block into account, each of the features is also combined with a position number (from 1 to 6) that indicates where the feature appeared. It is for example expected to find many author names at the beginning of a reference entry and only a few at the end.

Next we use the Selenium web browser automation framework[6] to determine the y-coordinate of each block. This framework is able to render an HTML page in a browser and provides an interface to get all visual and DOM based information for this page. The blocks are then ordered according to their vertical position and the features of block neighbors are added to the current block. Because it is expected that reference blocks are found in groups, the classification results can be increased this way. The same is true for blocks that are not reference entries. We could retrieve additional layout information about each block from rendering it in a browser, but since we have multiple reference sections in some documents, additional visual information were not useful.

As Support Vector Machine implementation we used the Weka Data Mining software (Hall et al., 2009). It implements a SVM with Sequential Minimal Optimization and some additional improvements to increase training speed (Keerthi et al., 2001). The SVM is then used to classify each block into the categories reference or non-reference. Since a CRF's purpose is to label an input sequence and not a single block, a whole document and all blocks in it are used as an input sequence for training. Unfortunately the Mallet CRF does not support continuous values for training, so first all features have to be discretized. This is done with the help of an algorithm by Fayyad and Irani (Fayyad and Irani, 1993), which is also implemented by the Weka framework.

[6] seleniumhq.org

4 EVALUATION OF REFERENCE LOCATING

To evaluate the reference locating, we collected a random set of 1,000 articles from our SpringerReference corpus that had a reference section. These were then split into 500 articles for training and 500 for testing. The 500 testing articles contained 42,023 blocks of which 8,983 were reference blocks. From these 41,764 could be correctly classified by the SVM approach and 259 got mislabeled. This means 99.3837% of the blocks got the correct classification, while only 0.6163% blocks were classified wrong.

The CRF however did not perform this well. It only achieved 91.3% accuracy. Perhaps the CRF is not able to extract the most important features out of such a variety of features. Additionally, since the CRF uses each document as only one training instance and treats all the blocks as an input sequence, 500 training instances might just not be enough training material. Another problem might be that the algorithm used for discretizing did not split the value ranges into usefull segments.

The results are slightly worse in comparison to those of Zou et al. (Zou et al., 2010). Only 8 blocks out of 22,147 got misclassified by their approach using the MEDLINE 2006 database. As already mentioned before, their approach would not be as successful on our data. In contrast to the MEDLINE documents, many of our documents have zero or only one reference. Despite that, there are articles that have more than one reference section e.g. a further reading section. Using a 'repair step' to find the first and last entry of a reference section would not be possible here. Either because there is only one entry or all entries between two different reference section would be classified as a reference section through this step. So we concentrated our effort on generating better features for learning, instead of trying to increase results through an additional 'repair step' at the end.

5 EVALUATION OF REFERENCE PARSING

As a reference parsing test domain we used the Cora reference extraction task (McCallum et al., 2000) to compare our approach to previous ones. This set contains 500 labeled reference with 13 different labels like *AUTHOR, BOOKTITLE, DATE, EDITOR, INSTITUTION, JOURNAL, LOCATION, NOTE, PAGES, PUBLISHER, TECH, TITLE, VOLUME* .

5.1 Labeling Results

Generally the same test approach as described in (Chang et al., 2007) was used. 500 reference instances were split into 300 for training, 100 for development and 100 for testing in the evaluation process. From this training set we take a varying number of samples from 5 to 300 for training. For semi-supervised learning we also use 1000 instances of unlabeled data, we took from the FLUX-CiM and CiteSeerX databases. These can be obtained from the ParsCit[7] website.

The labeling results are shown in Table 4. The results report the token based accuracy i.e. the percentage of correct labels and are calculated as averages over 5 runs. Column **Sup** contains the results for a CRF with the same features as previously described but with no constraints. Column **GE-KL** shows the results for our Generalized Expectation with Multiple Feature approach using the KL-divergence and the last column those for the L_2-Range distance metric **GE-L_2-Range**.

Table 4: Comparison of token based accuracy for different supervised / semi-supervised training models for a varying number of labeled training examples N. Results are an average over 5 runs in percent.

N	Sup	PR	CODL	GE-KL	GE-L_2-Range
5	69.0	75.6	76.0	74.6	75.4
10	73.8	-	83.4	81.2	83.3
20	80.1	85.4	86.1	85.1	86.1
25	84.2	-	87.4	87.2	88.4
50	87.5	-	-	89.0	90.5
100	90.2	-	-	90.4	91.2
300	93.3	94.8	93.6	93.9	94.1

In this paragraph, our method is compared to other state-of-the-art semi-supervised approaches. Column **CODL** shows the results for the constraint-driven learning framework (Chang et al., 2007). Here the top-k-inference results are iteratively used in a next learning step. In Posterior Regularization (column **PR**) the E-Step in the expectation maximization algorithm is modified to take constraints into account (Ganchev et al., 2010). Dashes indicate that the referenced papers did not contain values for a comparison.

As one can see, our approaches can compete with other leading semi-supervised training methods. While our approaches perform slightly worse than the others with a very limited amount of training data, one of our approaches outperforms the other techniques with $N = 25$ and $N = 50$ training instances. One recognizes that the introduction of constraints greatly improves labeling results. For $N = 20$ the improvement

[7] http://aye.comp.nus.edu.sg/parsCit/

of **GE-L_2-Range** in comparison to (column **Sup**) is 6 percentage points. In our experiments a CRF using GE constraints with multiple feature functions and $L_2 - Range$ as distance metric has the best labeling results.

The results also suggest that the positive influence of constraints decreases with an increasing number of training instances N. The traditional CRF (column **Sup**) is then able to determine proper weights for features without user provided constraints. In the case of relatively many training instances $N = 300$ complex constraints even seem to have slightly negative effects in comparison to use of simpler constraints in other approaches (column **PR**).

Table 5 shows precision, recall and the F_1 measure for each separate label with 15 training instances using GE with multiple feature functions and L_2-Range as distance metric.

Table 5: Precision, recall and F1 measure for label accuracy with $N = 15$ for a CRF with GE-L_2-Range.

Label	Precision	Recall	F_1
AUTHOR	98.5	98.6	98.6
DATE	95.0	82.5	88.3
EDITOR	92.3	52.8	67.2
TITLE	85.5	98.2	91.4
BOOKTITLE	84.3	84.1	84.2
PAGES	82.6	90.0	86.1
VOLUME	75.8	73.5	74.6
PUBLISHER	73.7	35.0	47.5
JOURNAL	71.9	66.6	69.1
TECH	67.5	25.7	37.2
INSTITUTION	62.4	43.4	51.2
LOCATION	51.4	60.0	55.4
NOTE	15.6	10.0	12.2

As Table 5 indicates, there is a big difference in the F_1 value for all labels. Because some labels like *AUTHOR* occur much more often in the training set and we set up more constraints for these labels, the performances in these cases are better. In contrast to that, labels like *NOTE* achieve a rather poor accuracy. However it is important to have a good performance for more common labels to achieve a good overall labeling performance.

It also has to be mentioned that in cases with only few reference instances the results can have a high standard deviation. The diversity of strings in this small portion of training material might simply be to high for proper training.

6 CONCLUSIONS AND FUTURE WORK

We proposed a new method for locating and parsing bibliographic references in HTML documents. We pointed out how HTML content can be grouped into blocks based on the Document Object Model. Afterwards, we have shown how these then can be used for classification into the categories 'is a reference' or 'is not a reference'. Therefore, the features and output of the reference parsing step are used to locate the bibliographic reference section.

In addition to that, we used other machine learning techniques like Part Of Speech tagging and Named Entity Recognition to obtain further input for our reference location process and so improved classification results. Next, a Conditional Random Field and a Support Random Machine for the classification task were compared against each other. In our setup the SVM approach achieved much better results than a similar approach using a CRF. Even though the reference location process by Zou et al. achieves slightly better results, we have shown that their approaches can not be transferred to our documents from the Springer-Reference corpus. Their 'repair step' can not be applied to our document domain, because it assumes many references in one section and only one reference section per document. Despite the difficulties of our test domain, our approach yields very good locating results.

Furthermore, we introduced a new method of parsing references with constraints. The labeling results proof that the proposed semi-supervised machine learning algorithm can compete with other state-of-the-art approaches. It even outperforms other approaches with a certain amount of training instances.

Both the reference locating and the reference parsing approach can easily be adapted to other domains of data. For example, the CRF with constraints approach could also be used to build a custom Named Entity Recognizer for historical texts. Since many NER approaches already use CRFs in the background, constraints could improve results while only needing a few training instances. The proposed classification algorithms could for example be used to determine if a page contains a shopping cart and so used for data mining purposes.

In future we would like to evaluate how bibliographic databases can be used to correct citations and even augment them with missing information as proposed by Gao et al. (Gao et al., 2012). The reference information can then even be used to calculate a relatedness for documents. This information could for example be used in an automatic link generation process for disambiguation. Since a method for the automatic extraction of keywords was proposed for feature extraction, we would like to concentrate future effort in

the automatic extraction of further features. On top of that, we are trying to not only include constraints in the learning phase of a Conditional Random Field, but also in the inference step. We believe that this could even improve labeling results.

REFERENCES

Bollacker, K. D., Lawrence, S., and Giles, C. L. (1998). CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proceedings of the second international conference on Autonomous agents*, pages 116–123. ACM.

Chang, M.-W., Ratinov, L., and Roth, D. (2007). Guiding semi-supervision with constraint-driven learning. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 280–287.

Councill, I. G., Giles, C. L., and Kan, M.-Y. (2008). ParsCit: An open-source CRF reference string parsing package. In *International Language Resources and Evaluation*. European Language Resources Association.

Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Thirteenth International Joint Conference on Articial Intelligence*, volume 2, pages 1022–1027. Morgan Kaufmann Publishers.

Finkel, J. R. (2007). Named entity recognition and the stanford NER software.

Fontan, L., Lopez-Garcia, R., Alvarez, M., and Pan, A. (2012). Automatically extracting complex data structures from the web. *International Conference on Knowledge Discovery and Information Retrieval (KDIR 2012)*.

Ganchev, K., Graa, J., Gillenwater, J., and Taskar, B. (2010). Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

Gao, L., Qi, X., Tang, Z., Lin, X., and Liu, Y. (2012). Web-based citation parsing, correction and augmentation. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, pages 295–304. ACM.

Ha, J., Haralick, R. M., and Phillips, I. T. (1995). Recursive XY cut using bounding boxes of connected components. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 952–955. IEEE.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Hetzner, E. (2008). A simple method for citation metadata extraction using hidden markov models. In *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 280–284. ACM.

Jain, A. K. and Yu, B. (1998). Document representation and its application to page decomposition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(3):294–308.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., and Murthy, K. R. K. (2001). Improvements to platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13(3):637–649.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probablistic models for segmenting and labeling sequence data. *In Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001)*, pages 282–289.

Lindner, S. and Höhn, W. (2012). Parsing and Maintaining Bibliographic References. *International Conference on Knowledge Discovery and Information Retrieval (KDIR 2012)*.

Mann, G. S. and McCallum, A. (2010). Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, 11:955–984.

McCallum, A. (2002). Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

McCallum, A., Nigam, K., Rennie, J., and Seymore, K. (2000). Automating the contruction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163.

Park, S. H., Ehrich, R. W., and Fox, E. A. (2012). A hybrid two-stage approach for discipline-independent canonical representation extraction from references. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, JCDL '12, pages 285–294, New York, NY, USA. ACM.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Sutton, C. and McCallum, A. (2006). *Introduction to Conditional Random Fields for Relational Learning*. MIT Press.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Zhai, Y. and Liu, B. (2006). Structured data extraction from the web based on partial tree alignment. *Knowledge and Data Engineering, IEEE Transactions on*, 18(12):1614–1628.

Zou, J., Le, D., and Thoma, G. R. (2010). Locating and parsing bibliographic references in HTML medical articles. *International Journal on Document Analysis and Recognition*, 2:107–119.