Combining Classifiers to Extract Information From Bibliographic References

Pedro Carloto de Castro
Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal

***Abstract*—The correct identification of information from fields that form a bibliographic reference is a crucial step to the proper functioning of repositories and search engines that aim to provide organized and accurate information to their users.**

To perform the extraction of information in bibliographic references, several learning models can be applied with success. However, this paper aimed to investigate if this process can be further improved using a combination based on aggregation techniques.

To answer this question, the present work creates and evaluates a system that combines the results from specialized classifiers using different aggregation techniques. Results show that the combination of outputs can improve the performance over individual classifiers.

In addition, a voting method that further improves the results of reference extraction is presented. This mechanism intends to be context independent, since it does not rely on training datasets or the knowledge of the classifiers being used in order to properly function.

I. Introduction

Publications have attributes that characterize them and make them unique. A string that combines these attributes (authors, title and others) is called a bibliographic reference. The process of parsing bibliographic references consists in finding and classifying the various parts that form a bibliographic reference (i.e., its individual fields) through an automatic process. To fulfil the purpose of transforming bibliographic reference strings into a structured format (for example, BibTeX[1]), the information needs to be extracted from the reference string.

Several information extraction techniques allow to extract information from Bibliographic References. These techniques have been applied in various tools and are being used in commercial and institutional sites. These tools are used for all sorts of information related tasks, such as searching or managing information. Generally these tools have been highly optimized and produce high quality structured bibliographic references.

The performance of different classifiers has already been closely analysed in previous works [3], [1] and [14]. Those studies found that typically each classifier is better suited to produce high quality results only for some categories.Also interesting, with different test datasets classifiers achieved significantly different results in the same categories. This motivates the search for a combination technique that adapts to the actual context of the bibliographic reference without relying on the training set that the classifier was trained with.

[1]http://www.bibtex.org/ - last consulted on 19-08-2012

A. Main Contributions

This paper contributes to the information extraction field in two different ways:

- A test of the viability of applying mechanisms of classifier combination to Bibliographic References structuring.
- A voting model that self adjusts to the quality of a classifier during test or even production phase.

The remaining of this document is structured in four main parts. Section II describes the Relate Work that sustains the work done. In Section III the main contribution of this paper is presented. The architecture of the created system, its components and the way they interact is explained. Section IV fully explains the evaluation process and all the results obtained are described and discussed. Finally, Section V concludes by summarizing the work done.

II. Related Work

A. Hidden Markov Models

Hidden Markov Models (HMMs) are well known in the statistics field and have well established training algorithms. These models are based in the assumption that a tag (generally a word) is part of a larger event and that there is a sequential pattern in the whole text.

An HMM can be described as a finite state graph with an initial state, final state and where the middle states can be achieved through transition probabilities. In every input there is also an observation output and for each state there are observation output probabilities. An HHM is referred as a *hidden* model because the state changes with every input and it only matters the current state and, to obtain the next state, only the current state matters (and not the ones before). So the states before the last one are hidden. HMM give us the benefit of only needing to look to the current state and to save the observations outputs. This creates the problem of not knowing the most likely sequence of hidden states that produced our observation sequence. The solution comes with Viterbi Algorithm [2]. It would be possible to *brute force* into the solution, but this algorithm is guaranteed to efficiently find the most probable path of a state sequence in a HMM.

In the context of Information Extraction, Hidden Markov Models have proven to be very efficient [13]. However, Hidden Markov Models present two major problems: the lack of rich representations of observations (features), that is, the lack of representation about non independent attributes about a word, for example a capitalized word in the middle of a text is

usually a noun. The second problem has to do with the fact that HMMs try to solve a conditional problem (predict the next state given the previous one) with a generative model[2].

Zhou and Su proposed an HMM-based chunk tagger [15] that presents above average results in IE. While traditional HMM assumes conditional probability independence, this works assumes mutual information independence. This means that token sequence is independent from its tag sequence. This allows to incorporate several features to improve the IE task: attributes of words(example: capitalization); the semantic classification of important triggers (example: "IN" is a prefix for a book title or journal). This classifier presents f-measure results of 96.6% and 94.1% against the data from MUC6 and MUC7 respectively.

B. Conditional Random Fields Models

In order to ensure tractable inference, HHM and other generative models require independent states. This means that the probabilities in one state only depend on that state. On the contrary, in CRF, probabilities are dependent on the previous sequence of input. This permits CRF to be a model that tracks inference and the same time does not make unwarranted independence assumptions [12].

CRF uses a single exponential model for the joint probability of the entire sequence of labels. This permits a model wide trade between the weights of features produced in the various states [8].

A Conditional Random Field can be classified as an undirected graphical model so it obeys to its probabilistic rules [12]. These models are based on maximum entropy models[3], which permits CRF to use limited training data and construct probabilities from that data. Since the first tests, CRF have shown solid results in the Information Extraction area [10].

For the purpose of our task a special case of CRF is normally used, the linear chain CRF. It corresponds to a finite state machine (FSM) and is used for sequence labelling. This special case will be briefly explained here. Let a state sequence be $\boldsymbol{s}=\boldsymbol{s_1 \ldots s_t}$ and the input sequence $\boldsymbol{x}=\boldsymbol{x_1 \ldots x_t}$. The feature function will be represented by $\boldsymbol{f_k(s_{t-1}, s_t, x, t)}$ where $\mathbf{t}$ represents the current time step. A feature can measure any aspect of a transition between states. The probability of $\boldsymbol{s}$ given $\boldsymbol{x}$ is given by:

$$\mathbf{P(s|x)} = \frac{1}{Z_x} \exp \sum_{t=1}^{T} \sum_{k} \alpha_k f_k(y_{t-1}, y_t, x, t) \qquad (1)$$

Z_x allows normalization, so the sum of probabilities of the various states is one. The last parameter, α_k is the learned weight of the feature. Using the Viterbi algorithm is then possible to efficiently calculate the most probable labelling sequence for a sequence input.

[2]A Generative model randomly generates observable data based on hidden parameters

[3]*Entropy of a probability distribution is a measure of uncertainty and is maximized when the distribution in question is as uniform as possible [11].*

Lafferty et al. [8] combined a conditional model with the global normalization of a random field model. The results obtained show that CRF is a more robust model when dealing with inaccurate model assumptions than maximum entropy Markov models [9] and HMMs. Further advantages from CRF models have been pointed in their work: efficiency in training and decoding processes using dynamic programming and the guarantee that the model will find the global optimum.

C. Result Aggregation Techniques

The reference extraction task can be performed by the various models previously described. Moreover, a combination of the above methods has been proven to generally improve the results [6], i.e. it is possible to choose the best result of the classification of a given token through the observation of the results given by all classifiers. This is possible because each classifier has a different design and each one is based in different mathematical foundations, so they will typically generate different errors [6].

This work will be focused on references, so pre-defined classes will be used in the classification process. .

The combination of different information extraction models can be made through various aggregation techniques two of which are described below:

1) Equal Voting: This is the simplest way to construct the aggregation of various results from different taggers to form an unified answer. In equal voting, or simple voting how it is also know, all the classifiers have the same weight so the decision will be made by *majority*. In the case of ties, one solution can be the selection of a random answer (within the results from the classifiers), or of the answer from a specific classifier that had proven to had better results when used alone.

2) Weighted Voting: In this method each classifier has a different weight in the voting process. This weight can be determined in different ways. One possibility is to test individually each one of the classifiers and give more weight to the classifiers that better perform in individual tests [4]. Another possibility is to determine more precisely the merit of each tagger in each individual tag that it makes. This is done by doing a cross-validation[4] of its result on the combiner training set. This process allow us to make specialized taggers that will have more weight in voting when a particular type of situation appears [6]. In order to combine different classifiers in a voting approach, linear interpolation is widely used as a combination function. Using this function the classifiers'class probability distribution is:

$$P(C|w, C_1^n) = \sum_{i=1}^{n} P(C|w, C_i) \times \alpha_i(w) \qquad (2)$$

The n classifiers(C) are combined regarding each word(w), and the weight of each classifier is represented by $\alpha_i(w)$. The output of each classifier on word(w) is represented by the output symbol, C_i.

[4]Cross validation is a technique to estimate the accuracy of a model in a practical case. It is implemented by splitting the data into various sets and subsets(training and testing sets). Then, each model runs a specific set in various rounds.

III. COMBINATION OF MULTIPLE CLASSIFIERS

The work described in this paper addresses two distinct problems in extracting information from Bibliographic references. The first is the combination of results from classifiers specialized in tagging bibliographic references. The second one is the selection of the best classifiers for each attribute without previously knowing the qualities of the classifiers.

Although these are distinct problems, they share the same goal: to improve the quality of the information extracted. The solution proposed will pursue this goal either by trying to solve the first problem or both at the same time. These problems present various challenges that will have to be tackled. It is necessary to: a) compare results that may have different structures; b) have different weights according to a classifier and also to the attribute being compared; c) have a method to dynamically adjust the weights of classifiers based on the answers outputted by it and the other classifiers.

A solution to overcome the above challenges is proposed. This solution allows combining tagged references from several different classifiers. The result of the overall process is a tagged bibliographic reference that combines chunks of text from different classifiers. The evaluation of the performance of each classifier and of the output generated by the combination process is also included in the solution.

This solution, named Reference Chooser, allows the combination of results using techniques based on weight voting and simple voting, both described in Section II-C.

Reference Chooser is intended to be as generic as possible, so that it is able to receive a tagged reference from any classifier as input to select a final answer. However, creating a generic solution poses some difficulties, since classifiers tend to produce different outputs. So, although the architecture of the prototype was made to combine results from a variety of sources, slight adjustments may be necessary when a new classifier, or any other source, is added. Three classifiers were chosen and were integrated with the prototype.

The proposed solution combines the output of different classifiers in an attempt to obtain a tagged reference with better quality than the ones produced by a single classifier. An overview of the solution is shown in Figure 1, which represents the sequence of actions performed by each component.

The main components are the Manager, the Prepare Reference Corpus, the Classifiers, the Normalizing Mechanism and the Combination Mechanism. The Manager controls the whole process. The Prepare Reference Corpus component is responsible for preparing the data such that it can be fed to the Classifiers. The Classifiers will receive the references and generate the tags for each found chunk. The Normalizing Mechanism will standardize the references to enable the Combination Mechanism to execute a voting procedure. The results from the voting procedure are then aggregated in order to form the final reference.

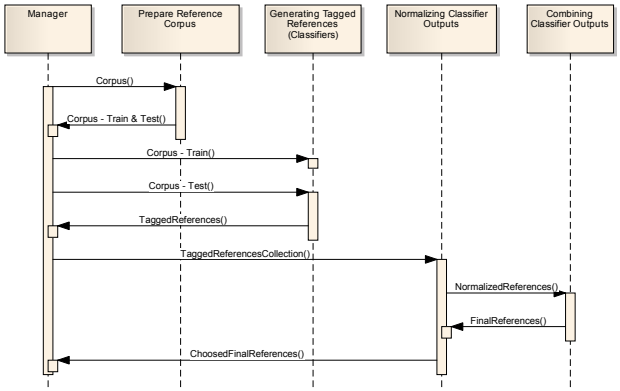


Figure 1. Flow diagram of the proposed solution

A. Architecture Component Description

1) Manager: As depicted in Figure 1, the Manager component controls the entire process. The use of a centralized controller creates a main advantage to the system: the possibility to change any part of the process, making it a versatile system. This makes it possible to: add more classifiers; change the location or type of the inputs, datasets or isolated references; and add new layers, new evaluation steps or additional processes to the flow of events.

2) Prepare Reference Corpus: This component is responsible for splitting the corpus into two parts. The first part, constitutes the training set and is used to train the classifiers; the second part, constitutes the test set, is used to test the classifiers and also the combination of results produced by the prototype.

3) Generating the Tagged References: This component generates the tagged references to be used in the remaining process. The generation of the tagged references is based on several specialized Bibliographic Reference Classifiers that receive non tagged references as input and output the same references already tagged.

To ensure reliable results for the evaluation process, the classifiers are only trained with the test dataset created by the component referred in Section III-A2. After this training process, all classifiers receive the test dataset produced by the same component and tag the references.

4) Normalizing Classifier Outputs: To enable the comparison it is necessary that the chunks from a reference being compared share the same format and do not have features that are meaningless for the voting mechanism but would distinguish them in a 1:1 comparison.

The following example illustrates such problem: classifier A outputs, as an author name, the chunk ”Mike A.” and a second classifier outputs ”A. Mike”. Although these chunks are different, the system should consider them equal because they refer to the same author. This problem can be addressed in a general way for certain situations, but may be dependent on the attribute being compared in other situations. Section III-B1 presents the standardization process and the components created to address this problem.

5) Combining Classifier Outputs: Section II-C reviews several result aggregation techniques. Three of these techniques were used by Reference Chooser to combine outputs: Simple Voting, Weighted Voting and Dynamic Weight Adjusting Voting. This last technique is adapted from Weight Training and is proposed in this paper as an alternative to the other ways of determining the weights that each classifier has in the voting process. Other ways to determine weights were already explained in Section II-C2.

In dynamic weight adjusting voting a classifier does not have a single value for its voting weight. It has a weight value for each possible class (attributes from bibliographic references) that can be tagged to a text chunk. This weight can also be seen as a **confidence value**, since it reflects the confidence gained by each classifier in the respective attributes. The final outputs will be subsequently chosen according to these values. Whether simple voting or weight voting is being used, Reference Chooser uses a point system to differentiate between confidence values and to allow the voting process to occur, this point system ranges from 0 to 5000.

Weight Voting techniques generally adjust the classifier confidence values in advance, before the start of the testing process. The technique proposed in this paper, **Dynamic Weight Adjusting Voting**, differs from this method and also from the method proposed by [6]. It adjusts the confidence values of the classifiers in each cycle of combining a chunck from a reference, i.e., each time a reference is tagged and then combined all confidence values are adjusted. This adjustment is only based on the answers of the classifiers and does not rely on the real correct tag from the dataset. This is crucial to ensure that the system can tackle "real life" challenges.

This process of weight adjustment allows a swift adaptation of the system, making it able to choose an answer based on a confidence value indicating that a classifier is better for that particular attribute - while still benefiting from the input of the other classifiers.

In the proposed voting system all classifiers start with the same confidence values which are adjusted only after the first choice (and in all subsequent choices). The final answer is the one that has the highest confidence value (HCV), that results from the sum of the individual confidence values of the classifiers. If the classifier answered with the HVC it is awarded:

- 10 / (number of classifiers that answered the HVC) points

If the classifier did not answer with the HVC, the following points are deducted from its score:

- 10 / (number of classifiers that did not answer the HVC) points

If all classifiers answered the HCV, no points are awarded or deducted.

Figure 2 presents an example of this process. Classifier A tags Dutch for the title class and has a confidence value of 500 points; classifier B tags Dutch for the title class and has a confidence value of 300 points; classifier C tags American for the title class and has a confidence value of 600 points. The HCV will be Dutch. Since classifier A and B have selected the answer that achieved the HCV. Classifier A and B will receive 5 points each. Classifier C will lose 10 confidence points.

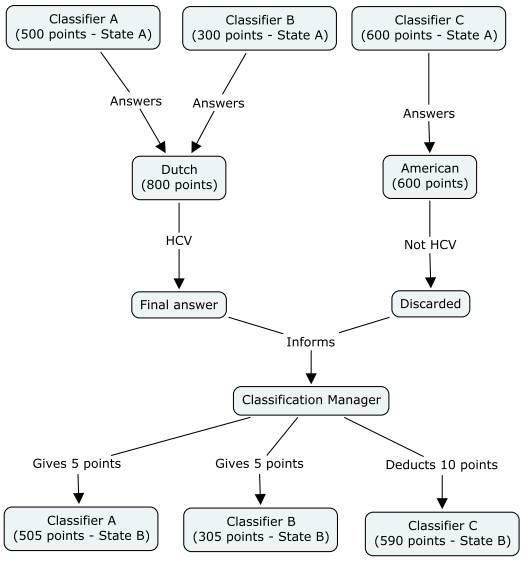


Figure 2. Example of the Dynamic Weight Adjusting Voting

The main advantages of this dynamic weight adjustment are:

- Smooth distribution of points: Every point a classifier looses, other classifier(s) gain(s). This is important to ensure both classifiers who perform well gain advantage and classifiers who answer incorrectly are penalized.
- No scores' inflation: if all classifiers tag the same chunk none receives points. This maintains a balanced system and reinforces the advantages mentioned in the previous point.
- Classifiers may fail without being too penalized: Even if a classifier gets the highest penalty by being the only one not to return the HVC, it can still recover in the long run, since all other classifiers will share the points from returning the HVC.
- This mechanism helps to distinguish the most skilled classifiers from the others. Since the failure of one classifier benefits others, after some cycles the system will be able to identify the best classifiers and rely more on these.
- No need for pre-adjusting confidence values: It is not necessary to adjust the starting confidence values according to the classifiers' performance in a specific attribute. This eliminates the need for individual tests of the classifiers, **but most importantly it allows to adjust a system according to the most common references that it will**

be evaluating instead of relying on the same references that were used to train the classifiers.

B. Prototype Implementation

This section provides details on the implementation of Reference Chooser for the purpose of this paper.

1) Attribute Normalizers: As previously mentioned, some chunks of text can be standardized by a generic component, but others need to be standardized taking into account the type of attribute to which they belong.

The **Generic attribute Normalizer** starts by making a simple standardization of the chunks it receives. This is done by removing simple features from each chunk, like punctuation. For the prototype used in this paper the punctuation removed was . (dots), , (commas), (quotation marks) and -(hyphens). This is an important step to ensure that the answers are compared based on their core information and the comparison is not harmed by punctuation. There is also the need for specific normalizers, like the **Author Normalizer** that is specialized in chunks from the authors class. It is required due to the characteristics of one of the classifiers, FreeCite. This classifier performs two changes to the authors, making its output different from the other classifiers, even when they are returning the same result. First, FreeCite presents the authors names in a order, that is often different from the original one: the surname is presented in the beginning even when in the original reference it was at the end; second, it separates the authors in different tags, i.e. it sub tags each author with its own tag. The first problem was solved by ignoring the order in which the names appeared. To solve the second problem, it was necessary to check if the other classifiers also found the authors found by FreeCite, and to make sure that the other classifiers did not find authors that FreeCite did not. If a new classifier was added to the process it could be treated either as a FreeCite type or a regular classifier.

In the prototype developed it was also necessary to create the **Volume Normalizer**, the **Date Normalizer** and the **Page Normalizer**. These all answered similar like the Author Normalizer.

It is important to notice that the architecture presented in this paper is generic enough to support adding new comparators to the comparator flow, in order to support the specificities of classifiers other than those used in the prototype.

C. Input and Output

The prototype presented in this paper has two main inputs. The first input is the collection of tagged references provided by the classifiers. The second input is a file that describes the initial confidence values for each classifier and its attributes. This XML [5] file also defines if the classification system is dynamic, (i.e. if the system gives and removes points to classifiers based on their correct/incorrect tagged answers) or all scores are always static. In this file each classifier is represented by a node with various elements that match the categories of the tagging process. Each element has an attribute that corresponds to the confidence value that a category starts with.

D. Spreadsheet Generator

This component creates a spreadsheet that allows a manual evaluation process. With the support of this spreadsheet, the output from the proposed system and also the output of each classifier can be evaluated in a precise and simple way. The spreadsheet shows all the necessary metrics to evaluate a classifier or a combination method, i.e. f-measure, precision and recall for each attribute, and also the global results.

E. Classifiers

1) FreeCite: FreeCite [6] was one of the classifiers chosen to integrate the prototype due to its good performance on an evaluation study made in the preliminary phase of this work. In this pre-test, FreeCite was trained using the full CORA dataset. The evaluation was made using 176 references randomly chosen from multi-disciplinary engineer publications in various languages, but mainly in English and Portuguese. The classifier obtained an f-measure of 79% between all classes. Its best performance was for the authors and the year attributes, with more than 90% f-measure. In contrast, its worst performance was in book title with an f-measure of about 65%.

A local service had to be created to allow the training of FreeCite with the same dataset used to train the other classifiers. This training process is essential to ensure that any evaluation done is strictly related to the classifier and not to differences between train datasets.

2) LingPipe: LingPipe [7] was used to implement an HMM and a CRF classifier in the solution. LingPipe is a versatile solution for processing text using computational linguistics. It is able to do various tasks, like finding the names of people, organizations or locations in general texts or suggesting corrections to the spellings of queries. For the specific purpose of the prototype used in this paper, LingPipe was adapted to tag and recognize attributes in bibliographic references.

The main reason behind the choice of LingPipe CRF and LingPipe HMM as classifiers to this combination process was due to their wide adoption in the scientific community [8]. This adoption is justified by the fact that LingPipe has pre-prepared packages for bibliographic reference tagging and tutorials on how to configure the system. It has also shown good results in this specific task.

IV. Evaluation

This section is divided in three main sections. In the first section the evaluation methodology is explained. The second section shows the results that are then discussed in the third section.

[5]http://www.w3.org/XML/ - last consulted on 05-05-2013

[6]http://freecite.library.brown.edu/ - last consulted on 05-05-2013

[7]http://alias-i.com/lingpipe/ - last consulted on 02-05-2013

[8]http://alias-i.com/lingpipe/web/citations.html - last consulted on 02-05-2013

A. Methodology

For evaluating Reference Chooser, we used the CORA dataset [9], a data set consisting in 500 bibliographic references. Half of the dataset was used to train each individual classifier, while the other half was used to perform the tests.

For each reference, we evaluated the following attributes: Authors, Title, Book title, Date, Editor, Institution, Journal, Location, Note, Pages, Tech, Volume and Publisher. Although the prototype was also prepared to combine the Number attribute of a bibliographic reference, this attribute was excluded because it is not tagged in the dataset. Furthermore, only the portion relative to the year in the date attribute was used in the evaluation, since one of the classifiers incorporated in the prototype ignores all other information such as the month or the day.

The evaluation process was divided in two main parts. In the first part FreeCite, LingPipe CRF and LingPipe HMM were individually evaluated. In the second part, the prototype configured with these three classifiers was evaluated. The tagged references made by the previous mentioned classifiers were used as an input for the combination process. Several different configurations were made using the prototype.

In each new configuration, the prototype was fed with different starting weights (also known as confidence values). Moreover, depending on the configuration, the confidence values could change or stay static during the process of testing the dataset. Depending on the type of voting in use. Further details were described in Section III-A5.

1) Configuration 1 - Proposed voting method: This configuration represents the voting method with dynamic weight adjustment proposed in Section III-A5. This means that the confidence values for each classifier were adjusted during the process, i.e. changed depending on the result of the voting. In this configuration the same confidence value was assigned for all attributes.

2) Configuration 2: In the second configuration each classifier started with different confidence values. These values were defined based on the output confidence values that were obtained after the execution of Reference Chooser's first configuration. In the second configuration the confidence values were also adjusted during the process.

3) Configuration 3: The third configuration was designed in a similar way to the second one. However, this time, the input confidence values were the output values that resulted from the execution of Reference Chooser's second configuration.

4) Configuration 4: This configuration is similar to the second configuration, since the confidence values obtained in the end of the execution of Reference Chooser's first configuration were also used as the starting confidence values of the fourth configuration. There was, although, an important difference between the two configurations, as the confidence values were static during the combination process in the forth configuration.

[9]http://people.cs.umass.edu/ mccallum/data.html - last consulted on 01-05-2013

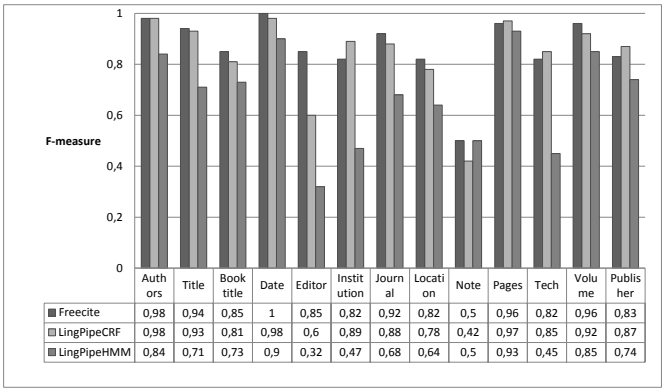
5) Configuration 5 Best for each attribute: In the Best of each attribute configuration the confidence values were based on the f-measure obtained by each classifier in their independent evaluation, such that the classifier who obtained the highest score for an attribute was given a bigger confidence value. The other two were given a smaller value, equal between the two. In this configuration there were no adjustments to the confidence values during the process of evaluation. Due to these fixed weights, the prototype always chose the same classifier to give the answer for a specific attribute.

6) Configuration 6 Simple voting: The sixth configuration designed for this evaluation is Simple voting. In the previous configurations all the voting was weight voting. As fully explained in Section III-A5, this means that the votes from different classifiers did not weight the same. In this configuration, however, all attributes from the three classifiers have the same confidence value and this value does not change during the whole combination process. This means that an answer is chosen based on how many classifiers choose it. If all classifiers choose different answers, the choice is random.

B. Results

This section is divided in two parts. In the first part the results obtained in the individual classifier tests are presented. The results obtained by the execution of different configurations in Reference Chooser are described in the second part.

1) Classifier Results: Figure 3 presents a direct comparison between the F-measures of the evaluated classifiers (FreeCite, LingPipe CRF and LingPipe HMM). Below the chart is a table with the corresponding values of the bars in the chart.



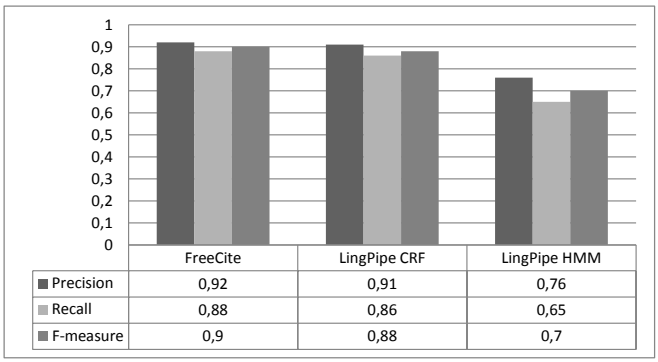
	Authors	Title	Book title	Date	Editor	Institution	Journal	Location	Note	Pages	Tech	Volume	Publisher
Freecite	0,98	0,94	0,85	1	0,85	0,82	0,92	0,82	0,5	0,96	0,82	0,96	0,83
LingPipeCRF	0,98	0,93	0,81	0,98	0,6	0,89	0,88	0,78	0,42	0,97	0,85	0,92	0,87
LingPipeHMM	0,84	0,71	0,73	0,9	0,32	0,47	0,68	0,64	0,5	0,93	0,45	0,85	0,74

Figure 3. F-measure obtained by each attribute in each classifier using the CORA dataset

From the observation of figure 3 some mains aspects should be highlighted:

- The only attribute where FreeCite and LingPipe CRF have equal results is the authors attribute where both classifiers scored a f-measure of 98%.
- The editor attribute has the largest distinction between the results of all classifiers. Specifically, FreeCite score 53% higher than LingPipe HMM.

- FreeCite achieved a f-measure equal or above 95% in the authors, date, pages and volume attributes
- LingPipe CRF achieved a f-measure equal or above 95% in the authors, date and pages attributes
- LingPipe HMM did not achieve a f-measure equal or above 95% in any category.



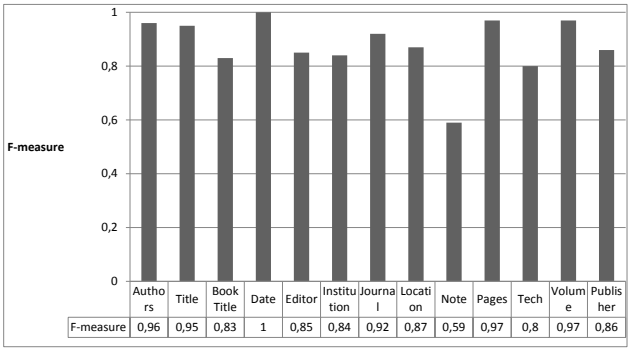
	FreeCite	LingPipe CRF	LingPipe HMM
Precision	0,92	0,91	0,76
Recall	0,88	0,86	0,65
F-measure	0,9	0,88	0,7

Figure 4. Global results obtained by the classifiers using the CORA dataset

Figure 4 compares the global results of the three classifiers. It is possible to observe in this figure that both FreeCite and LingPipe CRF achieved high results in both precision and recall. The difference between the global f-measure of these two classifiers is 2%, with the higher value belonging to FreeCite. LingPipe HMM results are not that high. For example, the difference between this classifier and LingPipe CRF is 18% in f-measure.

These global results are based on the micro-average measure, which takes in consideration the amount of chunks tagged from each category and is not simply an average.

2) Reference Chooser Results: Figure 5 represents the f-measure obtained in the execution of the first configuration of Reference Chooser.



	Authors	Title	Book Title	Date	Editor	Institution	Journal	Location	Note	Pages	Tech	Volume	Publisher
F-measure	0,96	0,95	0,83	1	0,85	0,84	0,92	0,87	0,59	0,97	0,8	0,97	0,86

Figure 5. F-measure obtained by each attribute in the first configuration of Reference Chooser using the CORA dataset

From figure 5 the next observations can be highlighted:

- With the exception of note, all attributes achieved a f-measure equal or above 80%.
- The attributes authors, title, date, pages and volume achieved a f-measure equal or above 95%.

Table I represents the output file that was created by Reference Chooser (from now on, referred as RC) in the end of the execution of the first configuration. The content of this file is explained in Section III-C. In table I all the attributes evaluated are represented in the rows; in the columns each classifier is represented with its number of correct answers and final classification for each attribute. A correct answer means that the classifier chose the answer that obtained the highest confidence value (HCV) in the voting process, as explained in Section III-A5. The Classification represents the confidence values that the classifiers had in the end of the process of the CORA test dataset. The last column Total Answers represents the total number of times that a voting process occurred for that type of attribute.

An important observation that can be made from Table I is which classifiers achieved the highest confidence value in each attribute:

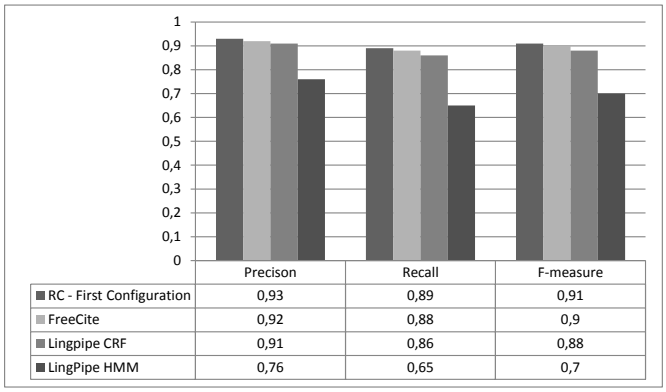
- FreeCite achieved the highest confidence value in Title, Editor, Journal, Location, Note, Pages, Volume and Publisher.
- LingPipe CRF achieved the highest confidence value in Authors, Book title, Institution and Tech.
- LingPipe HMM achieved the highest confidence value in Date.

	FreeCite		LingPipeCRF		LingPipeHMM		
	Correct Answers	Classification	Correct Answers	Classification	Correct Answers	Classification	Total Answers
Authors	524	1760	531	1865	465	875	538
Title	235	2110	212	1765	136	625	245
Booktitle	86	1575	92	1665	65	1260	105
Date	243	1505	287	1480	280	1515	243
Editor	17	1645	4	1450	1	1405	18
Institution	18	1520	21	1565	11	1415	23
Journal	91	1755	82	1620	49	1125	91
Location	52	1670	38	1460	32	1370	60
Note	5	1525	2	1480	3	1495	5
Pages	141	1590	137	1530	127	1380	145
Tech	16	1535	19	1580	6	1385	19
Volume	96	1665	89	1560	70	1275	96
Publisher	42	1610	34	1490	28	1400	42

Table I
CONFIDENCE VALUES OBTAINED AFTER THE EXECUTION OF REFERENCE CHOOSER WITH THE FIRST CONFIGURATION

Figure 6 shows a direct comparison between the individual results of the three classifiers and the results of RC - first configuration. This chart compares the precision, recall and f-measure of the global results obtained.

These results show that RC - first configuration had a slightly higher f-measure score when compared to those produced by the individual classifiers. This figure also shows that the differences occur with the same expression in Precision and Recall. LingPipe HMM stood out negatively as it perfomed worst that the remaining classifiers and RC - first configuration.



	Precison	Recall	F-measure
RC - First Configuration	0,93	0,89	0,91
FreeCite	0,92	0,88	0,9
Lingpipe CRF	0,91	0,86	0,88
LingPipe HMM	0,76	0,65	0,7

Figure 6. Comparison between the three classifiers and Reference Chooser with the first configuration

Figure 7 compares the f-measure obtained by the execution of the first four configurations of RC, for each attribute. From this figure it is important to notice some main aspects:

- With the exception of the location, note and the publisher attribute, the remaining attributes have the same f-measure.
- The first configuration of Reference Chooser has a slightly better performance in location and note attributes.

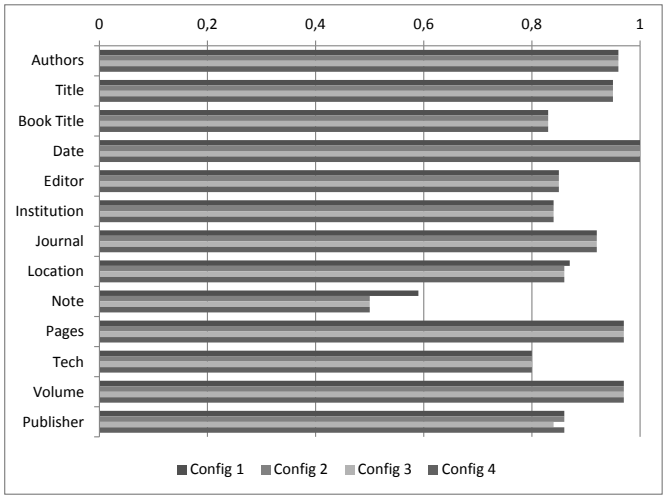


Figure 7. Comparison between the f-measure results from the attributes obtained from configuration 1 through 4

Figure 8 shows the comparison of the global results obtained by the first four configurations used in Reference Chooser. It is possible to see that the first configuration achieved a minor improvement compared to the other configurations.

As fully explained in Section IV-A, in the fifth configuration - Best for the attribute, the highest confidence values were given to the attributes that achieved the best f-measure in the individual classifier evaluation. The values of these attributes

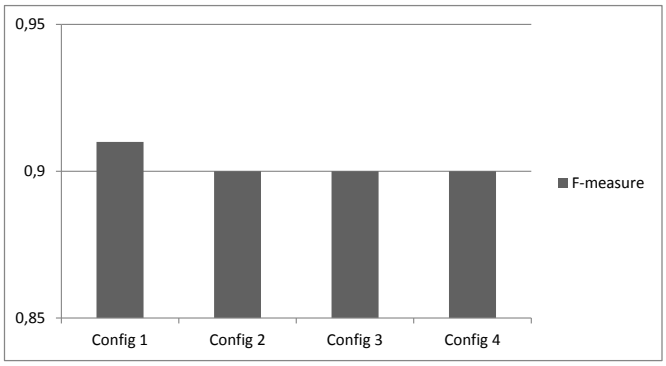


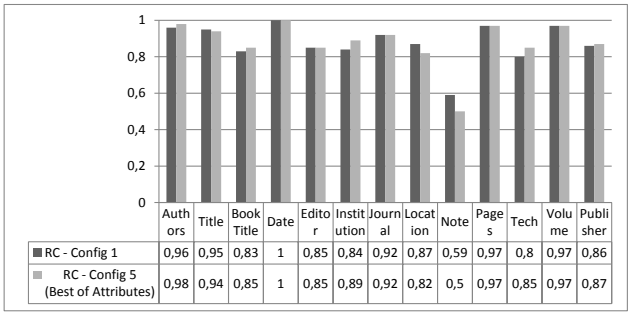
Figure 8. Comparison between the global f-measure results from configuration 1 through 4

can be seen in figure 3. FreeCite was given highest confidence values in the attributes authors, title, book title, date, editor, journal, location and volume. LingPipe CRF was given highest confidence value in institution, pages, tech and publisher and LingPipe HMM was given highest confidence value in note.

In Figure 9 the f-measure values obtained in the end of the execution with this configuration are compared to the ones obtained in the first configuration of Reference Chooser.

As expected, the f-measure achieved in each attributes in Best of Attribute configuration (config. 5) was exactly the same as the best attributes in the classifier individual evaluation. Compared to the first configuration of Reference Chooser the relevant aspects are:

- Best of attributes outperformed the first configuration in the following attributes: Authors, Book title, Institution, Tech and Publisher.
- The first configuration outperformed the best of attributes in the following attributes: Title, Location, Note.



	Authors	Title	Book Title	Date	Editor	Institution	Journal	Location	Note	Pages	Tech	Volume	Publisher
RC - Config 1	0,96	0,95	0,83	1	0,85	0,84	0,92	0,87	0,59	0,97	0,8	0,97	0,86
RC - Config 5 (Best of Attributes)	0,98	0,94	0,85	1	0,85	0,89	0,92	0,82	0,5	0,97	0,85	0,97	0,87

Figure 9. Comparison between the f-measure results from the attributes obtained from configuration 1 and 5

Figure 10 compares the same configurations as above but this time the global results are shown. It is possible to observe in this chart that the results are very similar, with only a 1% difference in precision. This difference also corresponds to a difference of 1% in the final f-measure.

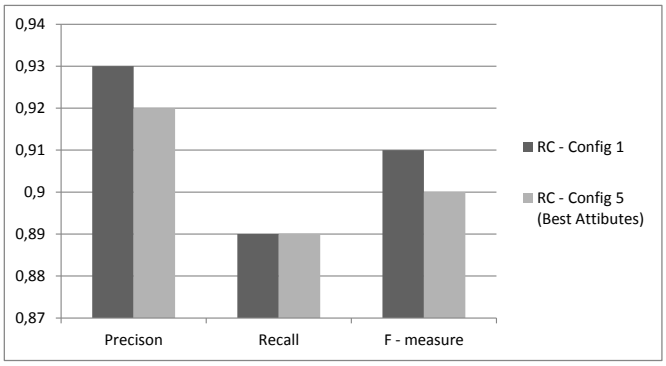
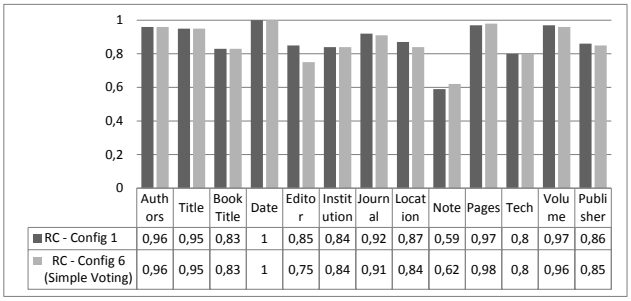


Figure 10. Comparison between the global f-measure results from configuration 1 and 5

Figure 11 shows a comparison between Simple Voting Configuration (config. 6) and the first configuration. From this figure it can be noted that:

- Simple voting outperformed the first configuration in the following attributes: Note and Pages
- The first configuration outperformed the simple voting in the following attributes: Editor, Journal, Location and Volume



	Authors	Title	Book Title	Date	Editor	Institution	Journal	Location	Note	Pages	Tech	Volume	Publisher
RC - Config 1	0,96	0,95	0,83	1	0,85	0,84	0,92	0,87	0,59	0,97	0,8	0,97	0,86
RC - Config 6 (Simple Voting)	0,96	0,95	0,83	1	0,75	0,84	0,91	0,84	0,62	0,98	0,8	0,96	0,85

Figure 11. Comparison between the f-measure results from the attributes obtained from configuration 1 and 6

Figure 12 compares the first configuration with Simple Voting (config. 6) in global results. It is possible to observe in this chart that the results are very similar, with only a 1% difference in recall. This difference corresponds to a 1% difference in the final f-measure.

C. Discussion

The viability of the new voting process can be seen from two different perspectives. The first one is through the direct comparison of the results obtained. The second perspective is to understand if the confidence values of the prototype are adapting towards a state where the attributes that achieved the highest f-measures in the individual classifier evaluation were favoured.

It was observed in Section IV-B that the execution of Reference Chooser with the first configuration achieved a higher f-measure than individual classifier's global results. The results of this configuration were also superior to those of the fifth configuration comparing to the fifth configuration Best for attribute and sixth configuration Simple voting.

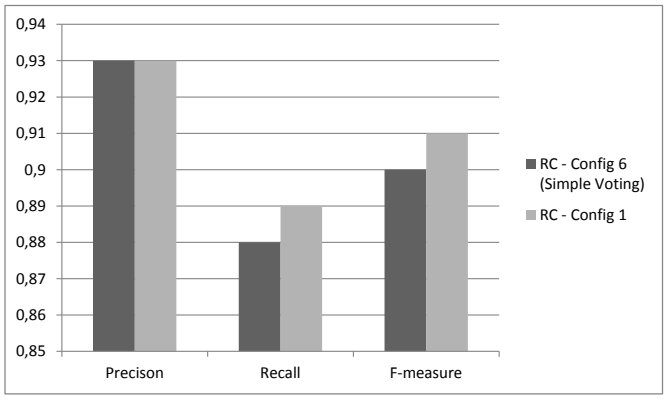


Figure 12. Comparison between the global f-measure results from configuration 1 and 6

From the second perspective the system is actually favouring 9 of the 13 attributes that achieved the highest f-measure in individual evaluation. This can be considered a positive aspect, since in most cases the combination process tends to favour the best classifier for each attribute.

FreeCite, LingPipe CRF and LingPipe HMM were the three classifiers used for the combination process which seems to represent a good choice to use in a combination process. The results achieved by FreeCite and LingPipe CRF show that they are very effective in tagging bibliographic references (Figure 4), which leads to the good results achieved by the prototype. Furthermore, although both use the same learning model - CRF, their results in each attribute are different (Figure 3) which make them distinct classifiers [7]. On the other hand, LingPipe HMM uses a different model for tagging, Hidden Markov Model. This diversity leads to Reference Chooser having more options for generating the final result, which enriches its output.

In the fifth configuration, as expected, the precision and recall results from each attribute matched the results from the attributes in their individual evaluation.

The results show that the first configuration of Reference Chooser achieved a higher f-measure than the use of the best attributes from each classifier. However, it is not possible to say that one is clearly better than the other, since the fifth configuration achieved better results in some attributes. What is important to highlight is that the fifth configuration needed to be pre-tweaked to have this performance. In the case of the first configuration it self adjusted to give these results.

In simple voting all classifiers share same confidence values so their vote will always weight the same. There is a small difference between the f-measure of simple voting and the results of the first configuration, and also the first configuration had better f-measure in three more attributes than simple voting. Nevertheless, a possible explanation for these results is that the adjustable confidence values are only affecting a small

percentage of the choices made by the combination process. FreeCite and LingPipe CRF agree in most of the answers. As a consequence, their confidence values tend to be similar, making them responsible for most of the decisions in the combination process. This then becomes a process similar to simple voting where only some combinations actually benefit from weighted voting. Nevertheless, since some combinations still benefit from a more complex process, and the others are not harmed by it, a complex process is advantageous.

V. Conclusions

The present document proposes a system that combines Bibliographic References from optimized classifiers and allows the combination of results in a broad range of ways. This system was configured to combine results in different ways, which included Simple Voting, Weight Voting and a special version of weight voting which consisted in adjusting the confidence values from the classifiers during test or production phases. The development and testing of Reference Chooser allowed the drawing of two main conclusions:

- **The combination of results is an effective way to improve the results obtained by individual classifiers**. However, compared to the application of combination techniques in other areas of Information Extraction these results are less significant. This can be due to the high f-measure of individual classifiers being used in this paper compared to those used by [6] which may have create a ceiling effect in the results.
- **A voting mechanism with dynamic weight adjustment allows the use of weight voting without the need of pre-evaluating the classifiers** and adjusting the confidence values accordingly. At first, this process may appear to be disadvantageous for the performance of the combination process, due to the lack of previous information. However, the work of this paper proves that it can actually improve the quality of results, even if slightly.

References

[1] Kamal M Ali and Michael John Pazzani. *On the link between error correlation and error reduction in decision tree ensembles*. Citeseer, 1995.

[2] J. Andrew. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[3] S. Džeroski and B. Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54(3):255–273, 2004.

[4] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics, 2003.

[5] D. Freitag. Machine learning for information extraction in informal domains. *Machine learning*, 39(2):169–202, 2000.

[6] H. Halteren, J. Zavrel, and W. Daelemans. Improving accuracy in word class tagging through the combination of machine learning systems. *Computational linguistics*, 27(2):199–229, 2001.

[7] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Decision combination in multiple classifier systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(1):66–75, 1994.

[8] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 282–289. Citeseer, 2001.

[9] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598. Citeseer, 2000.

[10] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics, 2003.

[11] C.E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[12] Hanna M. Wallach. Conditional random fields: An introduction. Technical report, University of Pennsylvania CIS, 2004.

[13] R. Weischedel, R. Schwartz, J. Palmucci, M. Meteer, and L. Ramshaw. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19(2):361–382, 1993.

[14] Lei Xu, Adam Krzyzak, and Ching Y Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(3):418–435, 1992.

[15] G.D. Zhou and J. Su. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics, 2002.