ISSN 1870-4069

A Web-based Tagger for Named Entities Detection

David Muñoz*, Fernando Pérez-Téllez, David Pinto

Computing Department, Institute of Technology Tallaght, Dublin, Ireland

{David.Efrain@postgrad.ittdublin.ie, fernandopt@gmail.com, davideduardopinto@gmail.com}

Abstract. Information Retrieval constitutes an important task for extracting meaningful data in huge amounts of texts since the majority of information in the world exist electronically, and in unstructured and plain text form. Most data-mining applications assimilate only structured information, this means before any predictive model can be applied the data must be prepared in a special way. This paper presents a web-based tagger system for named entities detection in plain text to help users tagging texts resources. The advantage of using this tool is the possibility of tagging any plain text file and with any type of class for the entities, allowing to treat any domain and thus obtaining a wider recognition of named entities. In addition, a small application shows the facility in the training of NER classifier models as an advantage of corpora constructed with the web-based tagger.

Keywords. Named entity recognition, NER, collaborative tagging, named entities, information extraction, information retrieval.

1 Introduction

The increasing use of computers to generate and administrate information causes the information overload. In this sense, Information Retrieval (IR) is the topic commonly associated with online documents. The IR goal is to measure the similarity between an input document and a document collection, finding the best matches in how similar are the results with the input document. In this way, Named Entities (NE) recognition represents an important aspect in the process of natural language understanding, and their effective identification comes to be significant in Information Retrieval related tasks.

The majority of information exists in unstructured and textual data, however for machines is hard to understand the unstructured data and that is why before any learning method -such as predictive models- can be applied the data must be prepared in a special format.

In the task of Named Entities recognition (NER) The Stanford Natural Language Processing Group provides a software implementation in the training of classifier models. However, the training data needs to be structured in a special way of two tabbed

*Corresponding author. Tel.: +52 1 22 26 26 82; E-mail: devasc26@gmail.com.

columns, in which the first column contains the word and the second column contains the tag assigned to the word. Taking into consideration the conversion of unstructured data into structured data is how the proposed system allows constructing corpora in this specific format for any type of plain text files.

In the last years, the task of extracting meaningful data of text has gained the attention in researcher and industry fields. Collaborative tagging systems have emerged as a solution for avoiding noise on information content since every day there is a lot of information generated and the majority is textual data [15].

The problem of highlighting relevant information over the information overload has gained the attention of many researchers, and as an answer, there are some works presenting alternatives as a solution.

Since information extraction is considered as a limited form of full natural language understanding, where the information we are looking for is known beforehand. It includes two fundamental tasks, named entity recognition and relation extraction [10].

In the task of universal semantic tagging, Abzianidze and Bos [8] contributes to better semantic analysis for wide-coverage multilingual text. The authors said that, besides their application in semantic parsing demonstrated in the PMB project, sem-tags can contribute to other NLP tasks, e.g. POS tagging, or research lines rooted in compositional semantics. In their work, the authors have shown that the tags provide semantically fine-grained information, and they are suitable for cross-lingual semantic parsing.

In the task of Named Entity Recognition (NER) in Tweets, Ritter et al [2] proposed a distantly supervised approach which applies Labeled LDA to leverage large amounts of unlabeled data in addition to large dictionaries of entities gathered from Freebase, and combining information about an entity's context across its mentions. This because classifying named entities in tweets is a difficult task since tweets contain a plethora of distinctive named entity types (Companies, Products, Bands, Movies and more), and almost all these types are relatively infrequent. On the other hand, tweets often lack sufficient context to determine an entity's type without the aid of background knowledge.

In the task of named entity classification, Mohamed and Oussalah [11] presented an approach by using the Wikipedia article info boxes where it has significantly reduced the classifier's processing time since the information inside the info box is structured. The proposed approach achieved a classification accuracy of above 97% with 3600 named entities and CoNLL-2003 shared task NER dataset used to validate the classifier's performance.

Font et al [5] proposed a general scheme for building a folksonomy-based tag recommendation system to help users tagging online content resources. They achieved this by using 3 independent steps: 1) Getting candidate tags, selecting a number of candidate tags for every input tag based on a tag-tag similarity matrix derived from a folksonomy, then 2) Aggregating candidate tags, assigning scores to the candidates of step 1 and merging them all in a single list of candidates tags, and finally 3) Selecting which tags to recommend, automatically selecting the candidates that will be part of the final recommendation by determining a threshold and filtering out those candidates whose score is below the threshold.

In the task of recommending items, the classification and prediction have an important role by analyzing data. It is important to know what classification algorithm to use depending on the application to be developed. As noted by Sheshasaayee and Thailambal, Classification is in supervised Learning of Machine Learning where a set of correctly predicted observation is available [3].

Chavaltada et al proposed a framework for automatic product categorization and explained that each classification method is affected in the efficiency of the model since each method have different the parameters [4].

In the task of text mining, Hawizy et al. [9], proposed a tool for semantic text-mining in chemistry where is possible to extract structured scientific data from unstructured scientific literature such as scientific papers or thesis which uses free-flowing natural language combined with domain-specific terminology and numeric phrases. In their work, they also demonstrated that using text mining and natural language processing tools, it is possible to extract both chemical entities and the relationships between those entities, and then making the resulting data available in a machine processable format.

The rest of this paper is organized as follows. First, the fundamentals and techniques employed are explained. Then, the proposed system (web-based tagger) and its architecture are explained. After, the case study of a small demonstration where the Stanford format is used to train the classifier. Finally, the critical evaluation and conclusion are presented.

2 Fundamentals and Techniques

This section introduces the named entities recognition in the process of natural language understanding as well as the use of collaborative systems. It also explains the V-fold cross-validation technique as a method for test the NER model´s ability to predict new named entities in datasets of unknown data.

2.1 Named Entities

NER is an important task in the field of information extraction systems since it aims to locate and classify named entities in raw text into categories previously defined (e.g. Person, Location, Organization). In this way, texts can be represented by their named entities. It is emerged in the Sixth Message Understanding Conference in 1995 [11]. Although, sometimes many of the NEs can be ambiguous to be classified in more than one class, e. g. the automotive company created by Henry Ford in 1903, where “Ford” can be referred to many entities (Name, Company, etc.).

On the other hand, NER systems require a large amount of highly accurate training data to perform well at the task named entities recognition [17]. In this way, excellent training data can be achieved by human feedback, since humans can easily differentiate from one context and another, assigning the correct tag to each named entity in the texts.

2.2 Collaborative Tagging

Collaborative tagging consists of assigning tags by users to a set of information resources such as plain text files. After that, tags can be used for many purposes, including retrieval, browsing, and categorization [7].

Halpin et al [6], claimed that there are three main entities in any tagging system: users, items, and tags. They produced a generative model of collaborative tagging in order to understand the basic dynamics behind tagging. So, they showed how tag co-occurrence networks for a sample domain of tags can be used to analyze the meaning of particular tags given their relationship to others tags. In this way, [1] proposed to model data from collaborative tagging systems with three-mode tensors, in order to capture the three-way correlations between users, tags, and items. He said that by applying multiway analysis, latent correlations are revealed, which help to improve the quality recommendations. He also developed a hybrid scheme that additionally considers content-based information that is extracted from items.

2.3 Cross-validation

Cross-validation methods are widely used in machine learning in order to estimate how accurately a predictive model will perform in practice. Cross-validation is based on the idea of data splitting, as in a prediction model usually is given a dataset of known data. So, part of the data is used for fitting each model, and the rest of the data is used to measure the performance of the model [13].

V-fold Cross-validation.
In practical applications, cross-validation is often computationally expensive. So, in order to achieve a smaller computational cost can be applied the method V-fold cross-validation, which depends on an integer parameter V.

Basically, the value of *V* should be small for complexity reasons, but not too small for decreasing the variability of the algorithm [14]. There are three well-known factors to take into account in order to choose *V*:

- *bias*. When *V* is too small, it leads to underfitting and suboptimal selection.
- *Variability*. The variance is a decreasing function of *V*.
- *Computational complexity*. *V*-fold cross-validation needs to compute at least *V* empirical risk minimizers for each model.

In the least-squares regression setting, *V* has to be chosen largely in order to improve accuracy (by reducing bias and variability); or it will generate computational issued arise when *V* is too big. This is why *V*=5 and *V*=10 are very classical and popular choices [13]. For this work, the value of *V* is 10 because the performance accomplished is much better and the computational power is low allowing to make fast experiments.

The explanation of the *V*-fold cross-validation method is illustrated in Figure 1, where the complete dataset is divided into the test data and the training data with the rest of the dataset. Each time with different elements until the final iteration is complete.

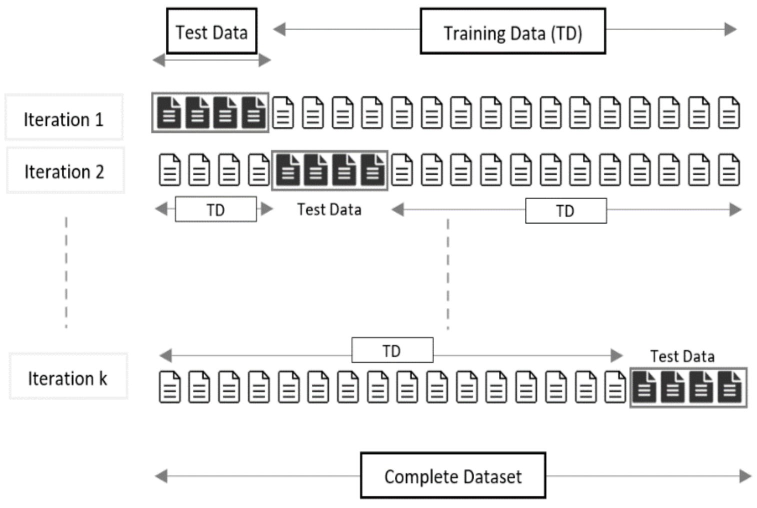


Fig. 1. An illustrative example of V-fold cross-validation.

3 Proposed System

The proposed web-based tagger aims to solve the problem of converting unstructured information to a structured format that can be easily understood by computers, the web-based tool receives plain text files as input and generates output files in a structured format ready to be used in the training of predictive models.

Figure 2 illustrates the role of the web-based tagger in the construction of high quality corpora in three main stages:

- In order to start the corpus construction process, it requires to provide an entry of plain text files to the system.
- In the second stage, the user begins to highlight the most important concepts in each file, assigning to each concept the tag that best identifies it. Since there are many domains in which is possible to tag concepts, the web-based tool allows the users to work in as many domains as necessary. Once all the relevant concepts in the domain have been tagged by the user, it is possible to export the tagged data in one of the three available formats: *Inline XML, keyword context, and tabbed column* format. When the *Inline XML* format is selected it simply find the indexes of each tagged concept enclosing between its corresponding assigned tags. If the *keyword context* format is selected then the concepts highlighted are extracted as well as the words before and after that concept. If the *tabbed column* format is selected then a tokenization step for each plain text file is made.
- After the information in the plain text document has been processed and formatted, it means that the document is now in a structured format and ready to be used in

machine learning algorithms. Finally, the structured data is available to be downloaded and used.

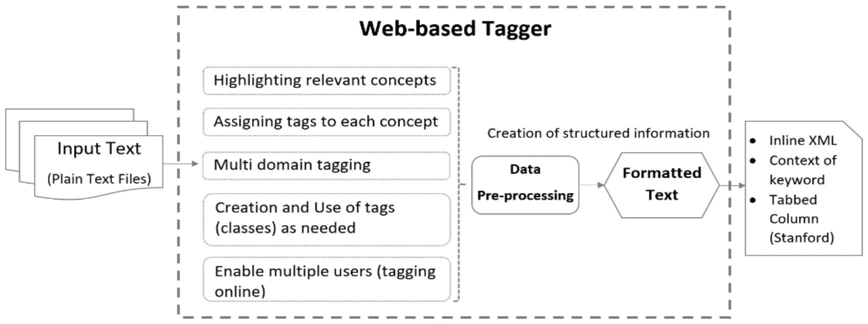

Web-based Tagger
Input Text
(Plain Text Files)
Highlighting relevant concepts
Assigning tags to each concept
Multi domain tagging
Creation and Use of tags (classes) as needed
Enable multiple users (tagging online)
Creation of structured information
Data Pre-processing
Formatted Text
Inline XML
Context of keyword
Tabbed Column (Stanford)

Fig. 2. The role of web-based tagger in the construction of high quality corpora.

The next points in this section describe the system architecture of the web-based tagger and explain more in detail the data output formats generated in order to train each model.

3.1 System Architecture

There are many reasons to have a tagger tool as a web application since it provides access to information from any device with internet connection. Also, it facilitates a huge number of users who can tag different plain text files simultaneously, improving the amount of tagged data.

Since the main task of tagging is to get tagged concepts for users, and in any of the domains the web-based tool needs to allow access to many users and from any place they are. Also, users should download the tagged files when they need it. On the other hand, it needs to facilitate the creation of new tags, since existing a lot of domains in which users can tag concepts.

Following these statements, the system architecture is illustrated in Figure 3. First, users must be authenticated in the Login Screen and depending on whether they are, administrators or taggers, the information, and permissions associated with the user are loaded.

If the authenticated user is an administrator, then its user interface consists of three main functionalities: administration of users, tags, and permissions. If the authenticated user is a tagger, then its user interface is composed of four main functionalities: upload plain text files and after selecting one of them, the user can begin to highlight the important entities until all important concepts have been tagged in the text. Once the named entities in the text have been tagged, the user can download the tagged text in any of the three formats available in the web-based tagger: Inline XML, Tabbed columns, and Keyword context.

The functionalities allowed to each user are explained in the next sections.

Users

As there is a big number of people tagging plain text files, it is necessary to authenticate the user session to know what data is allowed to tag. On the other hand, it facilitates access from any device, it means, once a plain text file was uploaded, the user can use it at any time, in any place, and in any device. There are two types of users:

– Tagger: Taggers are main generators of high quality tagged information since they have a good understanding in finding keywords in textual data. They are allowed to use a lot of different tags, so they are not limited to work in only one or few domains, as well as they are not limited in the number of files. Once they are logged in the system, they are allowed to: upload plain text files, search and select text files, then they can tag keywords in the text and download tagged files. Even, they can download tagged files that belong to other users, so they can use them to generate larger corpora.

  Something important is that each tagger can decide if he makes his file collaborative, this means to be tagged by other users. Thus, the quality of a tagged concept can increase substantially, considering that tags are assigned by different human feedbacks. This is achieved thanks to the system allows reassigning tags to the different concepts that have been tagged.

– The administrator: An administrator is allowed to create, modify and delete users, as well as administrate new tags and assigning tags to each tagger.

  So, tagger users can get tagged texts with high quality in tags and in any domain because there are no class restrictions.

Tags

Since the tags are created by the administrator to be used by taggers in the plain text files, a tag is defined as follows:

– Name. It denotes the class name, e.g. Person, Location, Organization, Skill and Knowledge.

– Id. It denotes a short name reference to the class name, e.g. Per, Loc, Org, Ski, Kno.

– Description. The description is a reference for tagger users, in order to explain what defines a certain class.

– Examples. The examples are provided in order to help users easily identify which concepts belong to a certain type of class.

– Color. The color is designed to highlight the text the concepts that are tagged by users in order to facilitate the reading and identification of each tagged concept.

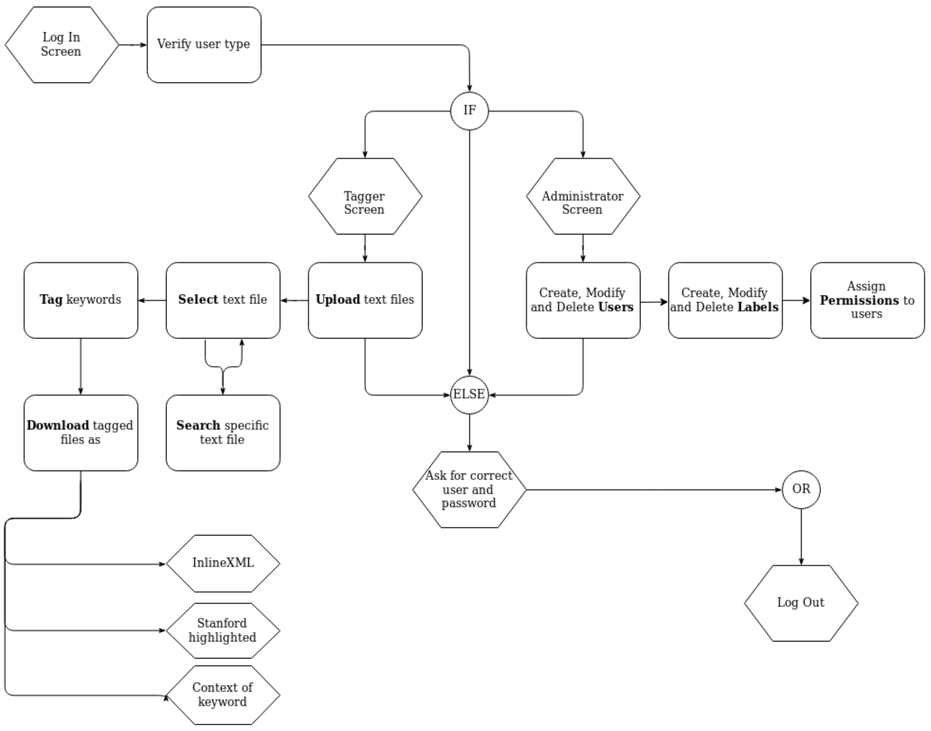
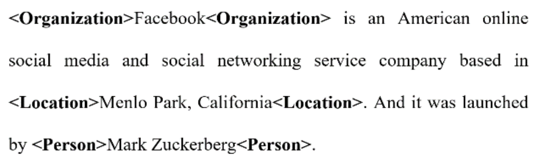


Fig. 3. The system architecture of web-based tagger.

Resources

The web-based tagger allows users to upload plain text files and immediately taggers can start to tag their files. When a file is uploaded a copy of it is generated to differentiate the file that is being displayed in real time from the one that will be tagged internally for later downloading. So, there are three types of formats in which a user can export a tagged text:

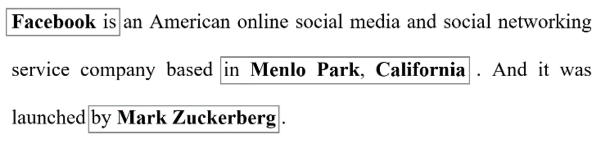
– Inline XML. It contains all the original text, with the exception that each item tagged is between the tags assigned to it. Since XML is a well-known and used format because it makes information easily accessible. An illustrative example showing the XML format is presented in Figure 4.



<Organization>Facebook**<Organization>** is an American online social media and social networking service company based in **<Location>**Menlo Park, California**<Location>**. And it was launched by **<Person>**Mark Zuckerberg**<Person>**.

Fig. 4. An example of an Inline XML format.

- Stanford - Tabbed columns. It contains the data prepared to be used in the training of a classifier model provided by Stanford NER [12]. The first column contains the words or tokens of the document and the second column represents the class to which it belongs if it belongs to one, in other cases the value of the second column is 0. An example of the format is presented in Table 1.
- Keyword context. It is a special format in which there are words that precede or proceed the tagged keyword since there are some keywords with different meanings depending in the context they are being used for. It is important to mention that the words that proceed and precede a tagged keyword, are only taken into account if these are not a punctuation marks since a punctuation mark indicates the beginning or end of another idea. An illustrative explanation the Keyword context format is shown in Figure 5.



Facebook is an American online social media and social networking service company based in **Menlo Park**, **California** . And it was launched by **Mark Zuckerberg**.

Fig. 5. An example of Keyword context format showing the importance of punctuation marks indicating the beginning or end of another idea.

Once the plain text files have been tagged any user can download the generated files, even to train immediately a classifier. This is important because in this way users can download datasets generated by other users (supervised training data) and experiment with this corpora as well as modify them if necessary.

The web-based tagger saves a lot of time since it makes the information accessible for everyone. As an advantage, the second format is ready to be used in NER Stanford training. At this point, the corpus can be constructed more quickly and even with more data available to train the classifier.

Table 1. Example of NER Stanford (Tabbed column) format avail-able to download in the web-based tagger.

Token	Class (label)
Facebook	Organization
Is	0
An	0
American	0
online	0
social	0
media	0
and	0
social	0
networking	0
service	0
company	0
based	0
in	0

Menlo	Location
Park	Location
,	0
California	Location
.	0
And	0
it	0
was	0
launched	0
by	0
Mark	Person
Zuckerberg	Person
.	0

3.2 Datasets

The number of examples used in this work is presented in Table 2. The table presents the number of job descriptions employed to train the NER Stanford Classifiers as well as the testing data, it also presents the number of classes and the number of examples per class. For this experiment, 50 different job descriptions related to the computing field were employed, and each one was tagged with at least one of the 3 classes: Knowledge, Skill, and Value.

There are a total of 18,766 tokens of which there exist 2,097 examples with the "Knowledge" tag, 956 examples with the "Skill" tag and 234 examples with the "Value" tag. The remaining 15,477 tokens have no category so in the second column the value for the label is 0.

All of this data is in the format of Stanford – Tabbed Columns provided by the same web-based tagger, and it consists in a tsv file containing the 50 job descriptions in a continuous text with the class labeled for each word, where the first column is for every single word and the second is for the belonging label of each word.

The 50 job descriptions are all related to the IT sector and all of them were obtained from the website: https://www.jobs.ie/. Each one was stored in plain text files.

Table 2. Job descriptions employed and the number of examples belonging to each class.

50 Job descriptions	
Class	Examples
Knowledge	2,097
Skill	956
Value	234
No assigned class	15,477

3.3 Tagger Interface

The main screen of the web-based tagger system is where users can tag texts as well as download them in the corresponding format to train the classifiers. This interface corresponds to the tagger user and a fast explanation of the design is showed in Figure 6.

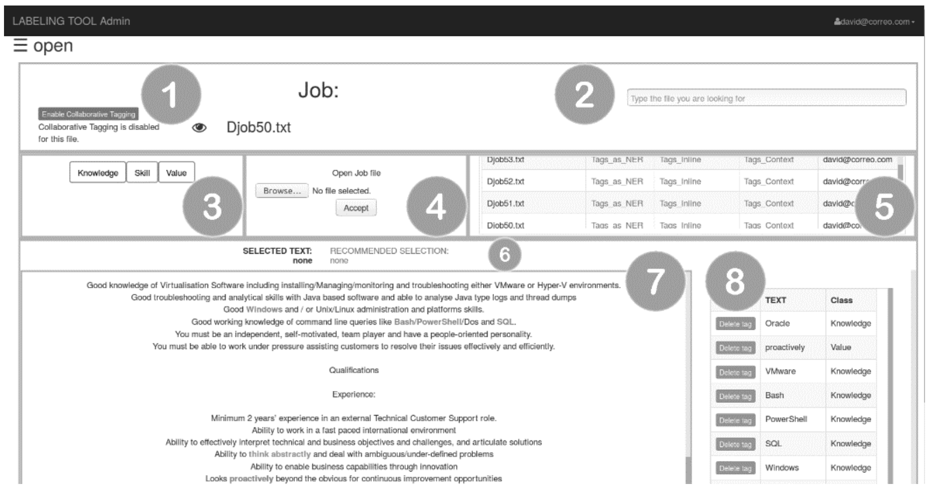
The first section, allows the user to visualize the current text in the edition and it also shows if the file is enabled to be editable in a collaborative way.

Section two contains an autocomplete field for searching a specific text file in the system as the number of files increase quickly.

Section three shows the allowed tags to be used for the user, these indicate the domain in which a user is working.

In section four, a specific text file can be loaded into the system by selecting from a local computer.

In section five, a list of text files existing in the web-based system is displayed as well as the available formats to download by only selecting the desired file.



Collaborative Tagging is disabled for this file.

Djob53.txt	Tags_as_NER	Tags_Inline	Tags_Context	david@correo.com
Djob52.txt	Tags_as_NER	Tags_Inline	Tags_Context	david@corr
Djob51.txt	Tags_as_NER	Tags_Inline	Tags_Context	david@c
Djob50.txt	Tags_as_NER	Tags_Inline	Tags_Context	david@co

SELECTED TEXT: none RECOMMENDED SELECTION: none

Good knowledge of Virtualisation Software including installing/Managing/monitoring and troubleshooting either VMware or Hyper-V environments.
Good troubleshooting and analytical skills with Java based software and able to analyse Java type logs and thread dumps
Good Windows and / or Unix/Linux administration and platforms skills.
Good working knowledge of command line queries like Bash/PowerShell/Dos and SQL.
You must be an independent, self-motivated, team player and have a people-oriented personality.
You must be able to work under pressure assisting customers to resolve their issues effectively and efficiently.

Qualifications

Experience:

Minimum 2 years' experience in an external Technical Customer Support role.
Ability to work in a fast paced international environment
Ability to effectively interpret technical and business objectives and challenges, and articulate solutions
Ability to think abstractly and deal with ambiguous/under-defined problems
Ability to enable business capabilities through innovation
Looks proactively beyond the obvious for continuous improvement opportunities

	TEXT	Class
Delete tag	Oracle	Knowledge
Delete tag	proactively	Value
Delete tag	VMware	Knowledge
Delete tag	Bash	Knowledge
Delete tag	PowerShell	Knowledge
Delete tag	SQL	Knowledge
Delete tag	Windows	Knowledge

Fig. 6. Tagger interface.

In section 6, appears the selected keyword or concept in the plain text file at the moment of editing, and a recommendation is offered since people can omit one or more characters at the moment of selecting the keyword.

In section 7, the currently selected text file is displayed allowing to observe in real time the tagged keywords by highlighting them in the corresponding color of the tag assigned, it makes easy readable the content for taggers.

Finally, in section 8, a list with the tagged keywords is provided and the option for deleting a tag for a specific keyword.

4 Case Study and Discussion

This section presents an application nugget (small demonstration) of the web-based tagger system in order to show its operation and highlight the advantage to generate more efficient corpus and open access to any domain. The first step is to create a tagged dataset of job descriptions in the IT sector, all of this made by human feedback since humans can differentiate better between two different contexts.

4.1 Data preprocessing

It has been implemented considering the principles of V-fold cross-validation method, in which the original dataset (50 job descriptions) is partitioned in k equal subsamples. Then, of the k subsamples, only one of them is taking as validation data for testing the model and the remaining k-1 subsamples are used as training data (corpus). The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation time.

Table 3. Partitions employed in the training data with V = 10.

50 Job descriptions	
k time	Training data (partitions) 70%
1	f_0-f_{34}
2	f_{15}-f_{49}
3	f_0-f_{19}, f_{46}-f_{49}
4	f_0-f_{19}, f_{21}, f_{23}, f_{25}, f_{27}, f_{29}, f_{31}, f_{33}, f_{35}, f_{37}, f_{39}, f_{41}, f_{43}, f_{45}, f_{47}, f_{49}
5	f_0-f_{20}, f_{22}, f_{24}, f_{26}, f_{28}, f_{30}, f_{32}, f_{34}, f_{36}, f_{38}, f_{40}, f_{42}, f_{44}, f_{46}, f_{48}
6	f_5-f_{39}
7	f_{10}-f_{44}
8	f_0-f_{29}, f_{35}-f_{39}
9	f_0-f_4, f_{10}-f_{14}, f_{20}-f_{24}, f_{30}-f_{49}
10	f_0-f_{24}, f_{26}, f_{28}, f_{30}, f_{32}, f_{34}, f_{36} f_{38}, f_{40}, f_{42}, f_{44}

As explained in fundamentals section, 10 has been chosen as the value of V because it performs better since the value of V is small for complexity reasons but not too much for decreasing the variability of the algorithm.

For this work, the 70% of job descriptions were taken to train the classifier and the remaining 30% to evaluate its performance. This test was executed 10 times by randomly taking 70% each time and using the remaining 30% in each case for its evaluation.

To identify a specific job description, file the notation used is f_x where f means "file" and x is the number of job description used. The first job description is identified with x=0 and the last one with x=49. The partitions used each time in the process are shown in Table 3. The first training data consists of the set of job descriptions from 0 to 34. The second training dataset is composed of job descriptions from 15 to 49.

4.2 Classifier training

When an NER classifier needs to be trained it requires a special tabbed columns format where the first column contains the word and the second column has the tag assigned for that word. The web-based tagger has generated the corpus in this format with the tagged files by users. Since Stanford NER is a Java implementation, the training of a classifier is easy to accomplish with only one command line.

Table 4. Number of iterations and evaluations by each calssifier.

Classifier	Iterations	Evaluations
1	107	125
2	107	126
3	101	118
4	107	125
5	113	126
6	119	134
7	114	126
8	105	115
9	107	122
10	104	116

As the value chosen for V is 10 the number of classifiers is 10 as well, and the training datasets are different every time. Each of this workouts took a different number of iterations and evaluations to each classifier to be ready. The values corresponding to the total number of iterations and evaluations for each classifier are shown in Table 4.

4.3 Results and Discussion

In order to measure the performance of the classifiers, the Stanford parser takes into consideration three measures: Precision, Recall, and F1. However, it is necessary to know some values before calculating these results.

A true positive is a token that the classifier has identified as a specific class that we have also designated with that class. If the classifier designates a token with a class that we have not assigned to that token, then it is seen as a false positive. A false negative is a token that we have identified with a class and the classifier fails to identify as that class. Finally, there exists the true negatives that measure the proportion of actual negatives that are correctly identified.

Precision is calculated by taking the total number of true positives and dividing that number by the combined sum of true positives and false positives.

The Recall is calculated by dividing the number of true positives by the sum total of true positives and false negatives. The F1 is a combination of Precision and Recall.

Table 5. Results of each calssifier.

Execution	Class	Precision	Recall	F1
	K	0.8081	0.5573	0.6596
1	S	0.6615	0.3333	0.4433
	V	0.7857	0.2200	0.3438
	K	0.9864	1.0000	0.9932
2	S	1.0000	0.9907	0.9953
	V	1.0000	1.0000	1.0000
	K	0.5455	0.5415	0.5435
3	S	0.5082	0.3196	0.3924
	V	0.5385	0.1628	0.2500
	K	0.7037	0.6111	0.6541
4	S	0.5667	0.3208	0.4096
	V	0.5556	0.2041	0.2985
	K	0.6549	0.5107	0.5739
5	S	0.6290	0.3250	0.4286

	V	0.6250	0.2273	0.3333
	K	0.7177	0.6224	0.6667
6	S	0.6984	0.3964	0.5057
	V	0.5625	0.2045	0.3000
	K	0.7864	0.6030	0.6826
7	S	0.7636	0.3925	0.5185
	V	0.7500	0.2308	0.3529
	K	0.7448	0.5838	0.6545
8	S	0.6667	0.3750	0.4800
	V	0.6667	0.2273	0.3390
	K	0.5689	0.5556	0.5621
9	S	0.5918	0.2589	0.3602
	V	0.6250	0.2222	0.3279
	K	0.7300	0.5319	0.6154
10	S	0.6508	0.3628	0.4659
	V	0.7273	0.2000	0.3137

Recall and Precision serves as an indicator of the success of our classifiers. In Table 5 the values for Precision, Recall and F1 measures corresponding to each class of the 10 times executed are presented.

For a better understanding of the results obtained in Table 5, a separate chart for each measure is presented in figures 7 to 9.

In Figure 7 the values obtained for Precision can be appreciated, were the Precision measure is the degree to which those tokens identified as Knowledge, Skill or Value by a given classifier are indeed that entity. In the first two iterations the performance of the NER classifier is excellent, however, the performance decrease in iterations 3 to 5, and after that, between iterations 6 to 10 the results obtained show a performance remaining constant around 70% of Precision measure.

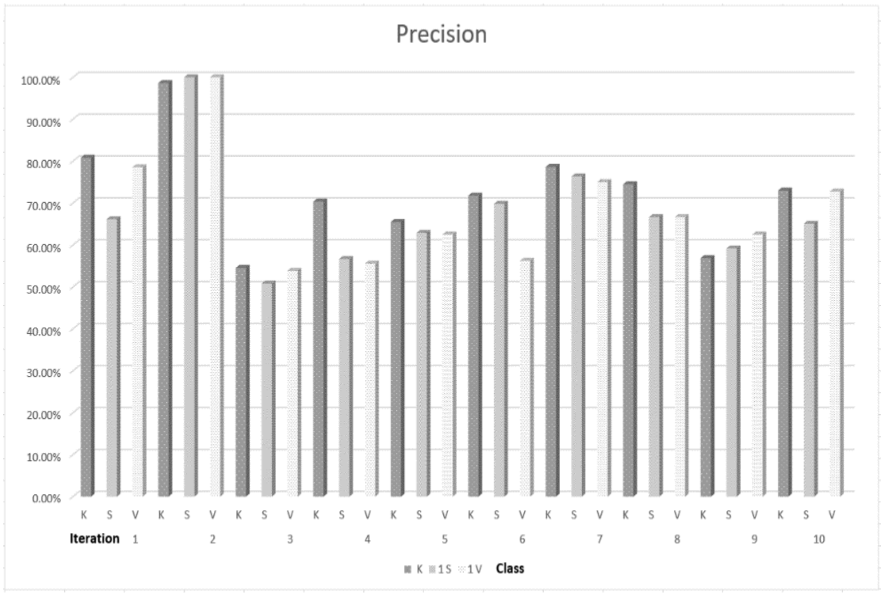


Fig. 7. Precision performance in 10 data tests.

It is also possible to observe the great performance for class Knowledge since most of the keywords reported as Knowledge is in fact Knowledge.

On the other hand, the Recall measure is an indicator of the degree to which a given classifier is able to find all of the tokens that we have identified as Knowledge, Skill or Value. In Figure 8 the values obtained for Recall measure are lower than in the Precision measure, with the exception of iteration 2 that is excellent. It is important to highlight the great performance for Knowledge class. In every iteration, the best performance is always achieved for Knowledge class and it can be possible as a result of more examples provided in this class. However, the results obtained in the Value class are far to be excellent since the examples are only a few pieces provided.

After the Precision and Recall values have been obtained, it is possible to combine those values into an F1 measure which serves as an abstract index of both. In Figure 9 the values obtained for the F1 measure can be appreciated. Again the best results are achieved for Knowledge class in each iteration, while for the Value class the results are low.

After analyzing the results obtained in every iteration and in every measure, it is evident to observe a great performance in precision measurement and always showing excellent results in Knowledge class, this means that most of the keywords identified as Knowledge are in fact Knowledge. Not the same with the Recall measure and Value class, as the number of provided examples is the minimum. The Skill class values are always in the middle of the results, presenting a few changes between a test and another.

The reported accuracy rates presented are good enough, demonstrating an excellent learning in the Knowledge class since wide examples were provided. That why the identified entities as Value in the unknown texts were poor.

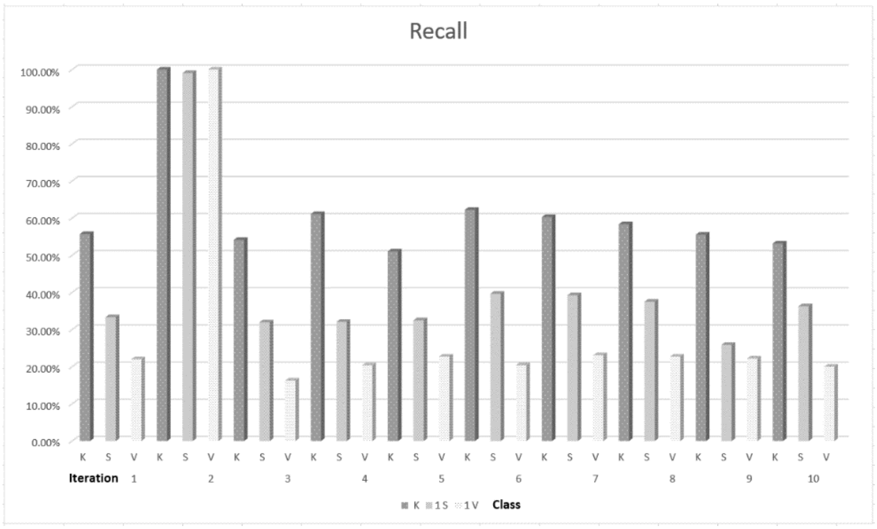


Fig. 8. Recall performance in 10 executions.

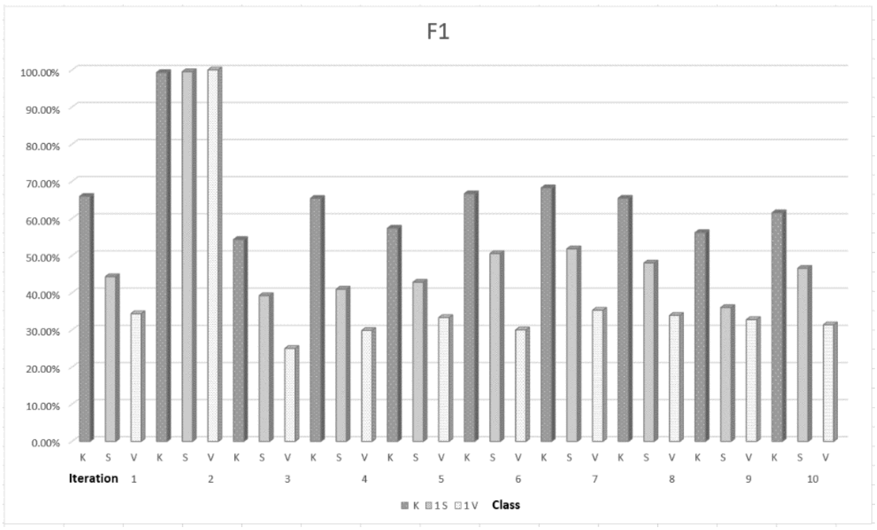


Fig. 9. F1 performance in the complete test.

This experiment shows the advantage of using the CRFClassifier of Stanford in the task of named entities recognizing. And thanks to the different dimensions of examples provided in the three classes it is clear to observe the great performance when a classifier model is trained with a considerable amount of examples for each class.

5 Conclusion

The proposed system allows the construction of high-quality corpora because the texts are tagged with human feedback, in this sense each concept or keyword is tagged under the decision of people which are able to differentiate between one context and another without a problem. The presented work in this paper offers the possibility to generate better corpora in any domain and even faster since these can be constructed in a collaborative way.

As a fast demonstration of the possible applications with the generated corpora in the Tabbed Column Stanford format an application nugget was performed and the performance measures obtained in each class present the excellent use of the CRFClassifier when named entities detection are required to extract relevant information on textual data.

Future experiments will consider the detection of the main characteristics in the domain of job descriptions and Résumés with a huge number of examples for every class since the number of examples are important to improve the results. Identifying the relevant aspects of a new job position in a specific company and the best applicant can reduce the time spent at the moment for finding an applicant.

The web-based tagger can be used to highlight the relevant information in any domain in plain text files. Once the most important data has been tagged the generated corpora can be employed in training classifiers to identify named entities in any kind of text.

References

1. Nanopoulos, A.: Item recommendation in collaborative tagging systems. IEEE transactions on systems, man, and cybernetics, NO. 4, 760–/71 (2011)
2. Ritter, A., Clark, S., Mausam., Etzioni, O.: Named Entity Recognition in Tweets: An Experimental Study, In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 1524–1534 (2011)
3. Sheshasaayee, A., Thailambal, G.: Comparison of Classification Algorithms in Text Mining. International journal of pure and applied mathematics, 425–433 (2017)
4. Chavaltada, C., Pasupa, K., Hardoon, D.: A comparative study of machine learning techniques for automatic product categorization. Springer international publishing, 10–17 (2017)
5. Font, F., Serrà, J., Serra, X.: Folksonomy-based tag recommendation for collaborative tagging systems. Inter-national Journal on Semantic Web and Information Systems, 1–27 (2013)
6. Halpin, H., Robu, V., Shepherd, H.: The complex dynamics of collaborative tagging. WWW 2007, 211–220 (2012)
7. Bischoff, K., Firan, C., Nejdl, W., Paiu, R.: Can all tags be used for search? In: Proceedings of the 17th acm conference on information and knowledge management, 193–202 (2008)
8. Abzianidze, L., Bos, J.: Towards Universal Semantic Tagging (2017)
9. Hawizy, L., Jessop, D.M., Adams, N., Mur-ray-Rust, P.: Chemical Tagger: A tool for semantic text-mining in chemistry. Journal of Cheminformatics 3 (17) (2011)
10. Allahyari, M., Safaei, S., Pouriyeh, S., Trippe, E., Ko-chut, K., Assefi, M., Gutierrez, J.: A brief survey of text mining: classification, clustering and extraction tech-niques. KDD Bigdas (2017)
11. Mohamed, M., Oussalah, M.: Identifying and Extracting Named Entities from Wikipedia Database Using Entity Infoboxes, (IJACSA) International Journal of Advanced Computer Science and Applications 5 (7), 164–169 (2014)
12. NLP. Stanford Named Entity Recognizer. The Stanford natural language processing group. Accessed online on May 2018. https://nlp.stanford.edu/software/CRF-NER.html
13. Arlot, S.: V-fold cross-validation improved: V-fold penalization, 1–17 (2008)
14. Geysser, S.: The predictive sample reuse method with applications. Journal of the American Statistical Association 70 (350), 320–328 (1975)
15. Weiss, S. M., Indurkhya, N., Zhang, T.: Fundamentals of Predictive Text Mining, Springer Verlag, 113–137 (2010)
16. Saquib, S., Siddiqui, J., Ali, R.: Classifications of Recommender Systems: A review. Journal of engineering and technology review, 132–153 (2017)
17. Tardif, S., Curran, J. R., Murphy, T.: Improved Text Categorisation for Wikipedia Named Entities (2006)