Rule-based location extraction from Italian unstructured text

Daniele Caruso, Rosario Giunta, Dario Messina, Giuseppe Pappalardo, Emiliano Tramontana
Department of Mathematics and Computer Science
University of Catania, Italy

Email: {giunta, pappalardo, tramontana}@dmi.unict.it

***Abstract*—Named entity recognition is a wide research topic concerned with the extraction of information from unlabelled texts. Existing approaches mainly deal with the English language, in this paper we present the results of a novel approach specifically tailored to the Italian language. The approach is directed at recognising location names in unstructured texts by several agents based on rules devised for the Italian grammar. Preliminary results show an F1 score up to 0.67.**

***Keywords*-Information Extraction, Named entity recognition, Free Text, Natural Language Processing, Italian Language.**

I. Introduction

Huge amounts of text data are easily available on the World Wide Web. Unfortunately, the great majority of such texts is in the form of unstructured or semi-structured text. Such a reality makes it difficult for both human beings and machines to make a good use of the content of such texts. Information Extraction is concerned with the process of structuring existing texts (both semi-structured and free) so as to single out some parts of text and have them accessed directly by some existing postprocessors [4].

A comprehensive survey of existing approaches [22] show how the Information Extraction community evolved from the seminal approaches since the early '90s, e.g. automatic learning of rules to extract entities [1], maximum entropy models [17], Conditional Random Fields [11], etc. Many, if not all, of these approaches are tested, or developed, on the English language. Moreover, specific analysers have been developed to embed security checks on software programs [10], discover structural properties [3], [12], [14], [18], [19], [23], [24], and perform automatic transformation of programs [2].

We are especially concerned with the problem of named entities extraction from free texts in the Italian language, in particular we are interested in the extraction of location names, i.e. proper nouns of places. Free texts can be of any kind, ranging from dialogues in a movie to fiction prose, thus enacting different constraints, however, in general, a location name is assumed to be written with a capital letter, and common names can be thus considered location names, especially in casual speech, e.g. in *Vediamoci in Dipartimento* (Let's meet at the Department)[1], where the said Department is a shared knowledge between speakers.

Unlike machine learning approaches, both unsupervised and supervised, we proposed a rule-based approach built from simple grammar rules of the Italian language complemented by a dictionary. The process of location names extraction is pursued by means of several specialised agents, each performing an elaboration step and connected in the pipe and filter style (see Figure 1), i.e. the result of the application of a rule removes the bulk of the candidate words, which later have to pass a further screening based, essentially, on a variant of a dictionary comparison.

The text is pre-filtered to remove punctuations symbols and then split into sentences. Each sentence is analysed by up to three rules (See Section III) so to identify word candidates, finally combined to remove false positives. Devised rules are typical Italian language patterns, identifying general contexts where a location can be found, thus the rules are not a simple filtering of words from an existing dictionary.

Preliminary results of the algorithm are encouraging: precision goes up to 0.82 and recall up to 0.92, while the comprehensive F1 score goes up to 0.67.

II. Phase 1: preliminaries

The approach and corresponding tool we have developed works on simple text files, i.e. a web page can be pre-processed beforehand by one of the many converters available to remove HTML tags.

For the devised rules, we make use of an especially compiled Italian lexicon, containing the following classes of words:

- **Articles**. A list of definite articles, e.g. *il* (the).
- **Prepositions**. Both kinds (*semplice* (simple) and *articolata* (composite)) but excluding *con* (with), as it is not used when naming places.
- **Verbs**. A subset of verbs related with places, such as *andare* (to go), *mandare* (to send), *partire* (to leave), *passeggiare* (to (take a) walk).
- **Descriptors**. A list of adverbs frequently related to a place, such as *dentro* (inside), *vicino* (near).
- **Non-places**. Words of various kinds (verbs, adverbs, nouns, etc.) not related to places, but that can appear in grammar structures (defined by the rules we set) as if they were places. E.g. *acido* (sour), *dormire* (to sleep). As

[1]We have decided to use both the original Italian and the translated version of any processed text we show, in order to allow a better appreciation of the proposed approach.

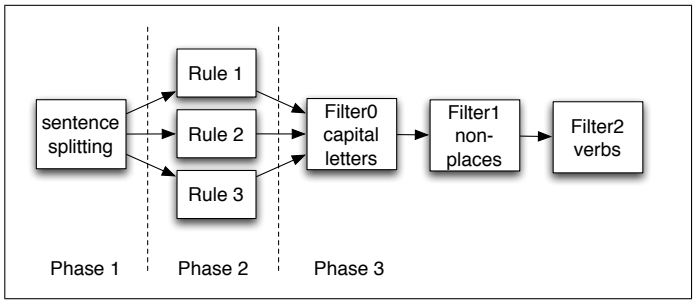

Fig. 1: The agents implementing the pipe and filter model

Verbs	Descriptors	Non-places
abitare (to dwell)	*avanti* (in front of)	*altrimenti* (else)
camminare (to walk)	*dietro* (rear)	*decimo* (tenth)
entrare (to get/come in)	*fianco* (side)	*allora* (then)
uscire (to get out)	*dentro* (inside)	*filosofo* (philosopher)
salire (to go up)	*fuori* (outside)	*molto* (much)
viaggiare (to travel)	*vicino* (near)	*scrivere* (to write)
partire (to leave)	*direzione* (direction)	*camminare* (to walk)
andare (to go)	*esterno* (outer)	*distrarre* (to distract)
indirizzare (to address)	*interno* (inner)	*florido* (prosperous)
raggiungere (to reach)	*lontano* (away)	*bere* (to drink)
risiedere (to inhabit)	*sinistra* (left)	*visitare* (to visit)
visitare (to visit)	*destra* (right)	*ognuno* (everyone)
imboccare (to access)	*adiacente* (adjacent)	*cremisi* (crimson)
arrivare (to arrive)	*vicinanza* (proximity)	*nostro* (ours)
svoltare (to turn)	*ingresso* (entrance)	*riempire* (to fill)
tornare (to go back)	*uscita* (exit)	*durante* (while)
fermare (to stop)	*dirimpetto* (opposite)	*esso* (it)
giungere (to arrive)	*attiguo* (adjacent)	*piatto* (flat)
parcheggiare (to park)		*lucente* (shining)
emigrare (to emigrate)		*spostare* (to move)
decollare (to take off)		*capace* (capable)

TABLE I
SAMPLE VERBS, DESCRIPTORS AND NON-PLACES WORDS

different words may be incorrectly identified as places, the approach assists users in the customisation of the set, by incorporating additional words, so as to exclude (refine) future results.

Table I shows the sample lists of words used in these categories.

In the following sections, such sets will be named after their initial, e.g. we will talk of V as the set of verbs.

Given a text T (read from an input file), the first step is to separate sentences, based on standard Italian grammar rules. T is split at occurrences of one of the symbols in the set of sentence-end punctuation marks, i.e. $\{\text{full-stop}, \text{ellipsis}, \text{exclamation-mark}, \text{question-mark}\}$, all the other punctuation types are removed in order to be processed by the next agents, obtaining a list of sentences. Any other non-letter symbol is ignored, e.g. dollar sign, percent sign, etc.

Each sentence in the input text is further segmented in order to find words, this is accomplished by using the space character as word separator, this applies to any rule we describe. The words found within a sentence are then compared with the entries in the lexicons, according to the different rules described in the next sections.

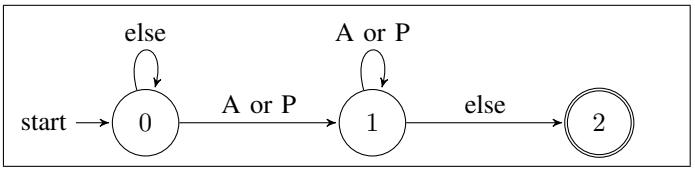

Fig. 2: The FSM implementing Rule 1

III. PHASE 2: RULE-BASED EXTRACTION

We defined three finite state automata, to implement three grammar cases possibly implying the use of a place in the accepting state of the automaton. Each rule identifies a different sentence pattern. The rules are applied at the sentence level, i.e. on a list of words terminated by a punctuation symbol, obtained in phase 1. The tokens (words) are fed to the automaton and, if an accepting state is reached, the current token is marked as a location candidate. If no accepting state is reached no candidate is produced. When a candidate is found, and a sentence still contains some more words, then the automaton restarts from its initial state using the token after the candidate, proceeding until the sentence ends.

The devised rules are independent from one another, so they can be parallelised by running as different agents e.g. on a multicore machine or in different machines coordinated in a Cloud fashion.

The result of each rule application is a list of candidate words, such words are used as input for the next phase (see Section IV) for the definitive labelling. Different rules possibly yields different candidates. Then, a way to use all the said rules is to combine them, hence the candidate words passing the rule filter(s) will be the union of the candidate words determined by each applied rule (see Section V).

A. Rule 1: Da Roma

The first rule, translating "from Rome", is used to identify possible candidate words as a location, and is named, as the other rules, after a typical example of a (part of a) sentence in which a place can be identified.

The automaton (see Figure 2) scans words (tokens) of a given sentence and remains in state 0 until a preposition (P) or an article (A) is found, this condition makes the automaton changes its current state from 0 to 1, and the state remains unchanged unless a different kind of word is found in the next token. Other articles or prepositions do not enable a state change, which is instead triggered by any other kind of word. The final state is reached when a candidate word for a place is found, however many candidates will be ignored afterwards, as described in Section IV.

As a single rule, this yields the highest number of false positives, as the use of an article or a preposition is very common in the Italian language.

B. Rule 2: Vicino a Roma

The second rule accommodates the mentioning of a place name in sentences such as the name of the rule suggests, "Near Rome", as the presence of a Descriptor (see Section II)

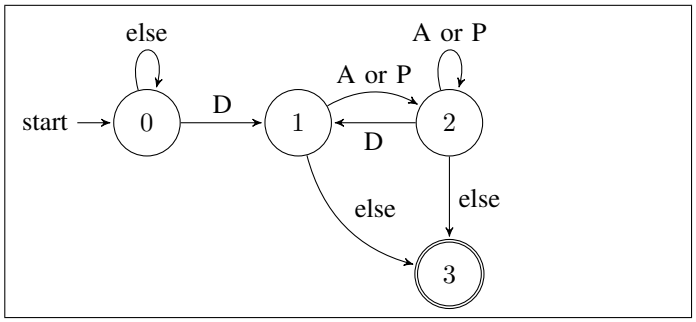

Fig. 3: The FSM implementing Rule 2

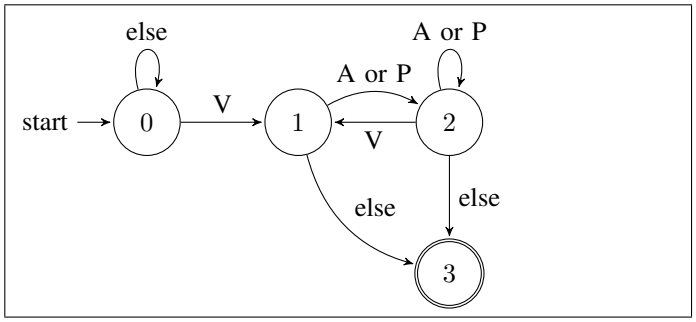

Fig. 4: The FSM implementing Rule 3

is a strong indication that a place will be mentioned in the following text in the same sentence.

Figure 3 shows the Finite State Machine (FSM) to find a candidate place. The automaton starts by reading the words one by one and does not change its initial state (0) until a descriptor is found, then it changes the current state to 1. From state 1 a transition can take place to the state 2, when an article or a preposition is found, or directly to the accepting state 3 in any other case. From state 2 it is possible to return to state 1, if another descriptor is found, or stay in the same state, if more articles or prepositions are found. Finally, the accepting state can be reached by reading any other kind of word.

The accepting state identifies a candidate word as a place, as *Roma* in the rule name.

C. Rule 3: Andando a Roma

While the previous rule uses descriptors as a way to identify a possible place name, this rule is concerned with verbs, such as in the rule name, "Going to Rome".

The FSM implementing the rule is shown in Figure 4. The behaviour of the automaton is the same of Rule 2, where instead of a descriptor a, possibly conjugated, verb is used. The verbs included are only verbs related to movement, and thus usually related to places, such as staying in a place or moving to and from a place. Since a verb can be found in a conjugated form, the check is performed using the Italian version of the stemming algorithm Snowball [21].

The automaton scans the tokens and remains in the state 0 until a verb is found and the current state is changed to state 1. The automaton may change from state 1 to state 2 (and vice versa), by reading a preposition or an article (or reading a verb). Any other words will make the automaton to change into the accepting state 3, i.e. pointing at the current word as a possible candidate for a place name.

IV. Phase 3: Non-place words removal

The candidate words yielded by the application of a rule are further filtered before being labelled as a place name. A candidate, to be considered a place name and thus evaluated as a positive result, has to pass the following filters.

- Filter1: The candidate word is checked against the Non-places lexicon (N). If it exists in N, then it is regarded a False Postitive (FP) and hence discarded. E.g. in the sentence *Andare alla capitale* (To go to the capital city), Rule 2 will suggest *capitale* as a possible place, however it is a common name and thus it will be discarded.
- Filter2: After passing the previous filter (Filter1), all the remaining candidate words are filtered to avoid identifying (conjugated) verbs as places. Once again, the check is performed using a stemming algorithm [21]. To check if a candidate word is a verb, it is stemmed and then concatenated with the three possible suffixes used in Italian verbs (*-are*, *-ere* and *-ire*) so to get the infinitive form of the verb, which is then searched for in the Non-places lexicon. E.g. if the word (a verb) appears in the lexicon, it is discarded as it is not a place name. E.g. in the sentence *Ella esce camminando* (She gets out walking), Rule 3 stays in the state 0 for *Ella*, then goes into state 1 reading the verb *uscire*, however the next token is not an article nor a preposition, thus *camminando* is proposed as a place candidate. However, in this filtering, such a place candidate is recognised as a conjugation of the verb *camminare* and finally discarded. The remaining candidate words are promoted as results.

Any word passing the said filters are labelled as a place name, however such a result may be a True Positive (TP) or a False Positive (FP).

Unstructured text may not be reliably using orthographic conventions, as a text could be a professionally proof-read book or an informal automatic transcription, thus it may or may not use capital letters to address location names. As far as we described our approach, we did not make any assumption on such an orthographic convention, however, experimentally, we found better results when such a convention is satisfied, thus we also provide a further filter:

- Filter0: If the candidate begins with a lower case character, it is not deemed a location name, while it is output as a result if it starts with a capital letter.

As the name suggests, this filter has to be applied before Filter1 and Filter2, as the user sees fit, based on the text to be processed.

V. Discussion

In the next subsections we review the rules and how they relate to the actual grammar writings we examined, and then show the results of the labelling experiments made on different texts.

A. Rule's Assessment

Real case examples. Table II shows several fragments of sentences recognised by our approach, for both TP and FP results, as specified in column 3. The words in italics are the tokens consumed by the automaton for the current rule, while the bold word is the one identified as a place.

Lexicons. As the rules are based on several lexicons, the completeness of such lexicons is essential for a good recognition. In the sentence *"dentro la stanza"* (inside the room), the rule 2 will candidate "stanza", which should not be proposed as a result, as it is not a proper location name. It is a responsibility of the Non-places filter (Filter 1, see Section IV) to recognise that the word is not a location, however, if "stanza" is not in the N lexicon, it will be selected and thus proposed as TP while being a FP.

Repetitions. A simple observation of the FSMs shown in Figures 2 to 4 may lead to a traversal of the states recognising illicit sentences for the Italian grammar. E.g. *"Andando camminare per per Roma"* (Going to walk to to Rome) in which the application of Rule 3 would candidate "Roma" as a place name. Our preliminary studies show that ungrammatical sentences, such as the previous example, are not so frequent unless we factor in informal languages, such as instant messaging or poetic prose/verses.

However, the same rules are capable of recognising ungrammatical sentences appearing in both formal and informal speech. A phrase such as *"Andando a... a... a Roma"* (Going to... to... to Rome) would make the FSM in Figure 4 pointing at "Roma" as an accepting state, even if the sentence is not grammatically correct. As such a sentence can be typical in speeches, e.g. when one speaks while recalling something, an automatic transcription may report such sentences and thus we left the loops in the rules.

Sentence patterns. The rules we are proposing can be considered arbitrary, even if intuitively correct. Thus, before making the actual experiments in labelling, we studied the result of the application of the rules alone on a set of unstructured texts so as to check if such grammar structures had the needed responsiveness degree. I.e. we are interested in the possible paths any automaton may take, given real written texts and not just simple cases (such as the ones in the titles of subsections in Section III, which are correct but also very basic).

Rules have been tested on different kinds of textfiles, both prose and dialogue transcriptions, for a total of 1.2 million of characters. The results are shown in Table III. In each line, the first column is the rule, the second is the sentence pattern found by the rule, the third column is the number of instances of the pattern found in the test corpus. The sentence pattern is identified by the transition in the automaton, e.g. VPA (Verb, Preposition, Article) identifies a sentence such as *Viaggiare per l'Italia* (To travel in Italy), which is decomposed as *Viaggiare*$_V$ *per*$_P$ *l'*$_A$ *Italia*, where the words before *Italia* are being catalogued respectively as [V]erb, [P]reposition and [A]rticle.

Rule	Sentence	Result
1	Il processo che si svolge *a* **Milano**	TP
	(The trial taking place in Milano)	
	I treni a lunga percorrenza *per la* **Sicilia**	TP
	(Long distance trains to Sicily)	
	Il vertice che si terr oggi *a* **Bruxelles**	TP
	(The meeting taking place in Bruxelles)	
	Se *il* **Ministro** in indirizzo non intenda intervenire	FP
	(If the addressed Minister does not mean to intervene)	
2	Scappa *verso il* **Canale**	TP
	(Runs away towards the Channel)	
	All'*interno della* **Basilica** Palladiana	TP
	(Inside the Basilica Palladiana)	
	Mi fanno sedere *accanto a* **Carlo**	FP
	(They let me sit beside Carlo)	
	Sono operative *presso le* **DIGOS** di tutto lo Stato	FP
	(They are operational in the DIGOS (offices) of the State)	
3	*Passando per* **Piazza** Del Popolo	TP
	(Proceed through Piazza Del Popolo)	
	La prima volta che *vedo* **Palermo**	TP
	(The first time I see Palermo)	
	Non ho più *visto* **Carlo**	FP
	(I have not seen Carlo)	
	La soglia richiesta per *entrare in* **Parlamento**	FP
	(The threshold required to get into the Parliament)	

TABLE II
SAMPLE RESULTS USING DIFFERENT RULES

Rule	Sentence Pattern	Occurrences
Rule 1	A	1076
	P	4174
	AA	1
	AP	1
	PA	56
	PP	4
	APA	1
Rule 2	D	58
	DA	41
	DP	73
	DADP	1
	DPDP	2
Rule 3	V	82
	VA	26
	VP	190
	VPA	1
	VPV	1
	VAVP	2
	VPVP	1

TABLE III
DIFFERENT SENTENCE KINDS

Given a rule, a Sentence Pattern such as A is more general than any pattern having A as a suffix e.g. PA. Thus, all the occurences of PA form a subset of the occurences of A. For the experiments (Section V-B) the automata are set to found the longest match.

The preliminary study reported in Table III shows just the number of occurrences for each sentence pattern, it does not show the percentage of TPs or FPs, as this is just a way to check the different transitions in the proposed automata.

B. Experiments

The rules detailed in Section III have been developed in a tool and have been tested on different kinds of unstructured texts: (i) theatrical dialogue transcriptions (texts T2, T3, T4),

(ii) official stenographic transcriptions of political debates (T5) and (iii) news articles (T1). In the two latter cases, the transcriptions are properly capitalized, and thus the Filter0 (see Section IV) has been used in the experiments, while the other texts were all in lower cases and thus only Filter1 and Filter2 have been used in phase 3 (see Section IV).

All the texts used for the experiments have been manually labelled for the location names. In the experiments, all the combinations of the rules have been tested, as shown in the second column. E.g. Rule "2&3" means to put together as a mathematical union the set of candidates gathered by Rule 2 with the set of candidates gathered by Rule 3, using such an union for the filtering agents in phase 3.

The precision metric is computed as a correctness measure, using also the number of False Negatives (FN), as $\frac{TP}{TP+FP}$, while the recall is computed as a completeness metric as $\frac{TP}{TP+FN}$. The F1 score gives the harmonic mean of precision and recall.

There are cases where a rule fails to identify any TP, however this is expected. When an input text does reference a place name by e.g. a motion verb, then only Rule 3 can be able to recognise such places, while Rule 2, concerned with the usage of descriptors, will never be applied.

The results show an interesting F1 score, going up to 0.67 with an average of 0.38. The precision metric goes up to 0.82 in the best case, with a minimum value of 0.27 and an average of 0.45. The recall shows also good results, having a maximum value of 0.92 and an average of 0.51.

While there are cases where very few location names are identified, we deem such preliminar experiments worth expanding, as one of the limitations is the small number of labelled text which we have dealt with.

Text	Rules	TP	FP	FN	F1	precision	recall
T1	1	39	49	9	0,57	0,44	0,81
	2	1	1	47	0,04	0,50	0,02
	3	0	3	48	n/a	n/a	n/a
	1&2	39	50	9	0,57	0,44	0,81
	1&3	39	52	9	0,56	0,43	0,81
	2&3	1	4	47	0,04	0,20	0,02
	1&2&3	39	53	9	0,56	0,42	0,81
T2	1	33	49	6	0,55	0,40	0,85
	2	2	1	37	0,10	0,67	0,05
	3	9	2	30	0,36	0,82	0,23
	1&2	33	50	6	0,54	0,40	0,85
	1&3	34	51	5	0,55	0,40	0,87
	2&3	11	3	28	0,42	0,79	0,28
	1&2&3	34	52	5	0,54	0,40	0,87
T3	1	13	7	6	0,67	0,65	0,68
	2	0	0	19	n/a	n/a	n/a
	3	3	2	16	0,25	0,60	0,16
	1&2	13	7	6	0,67	0,65	0,68
	1&3	14	9	5	0,67	0,61	0,74
	2&3	3	2	16	0,25	0,60	0,16
	1&2&3	14	9	5	0,67	0,61	0,74
T4	1	56	136	5	0,44	0,29	0,92
	2	0	4	61	n/a	n/a	n/a
	3	9	15	52	0,21	0,38	0,15
	1&2	56	140	5	0,44	0,29	0,92
	1&3	56	151	5	0,42	0,27	0,92
	2&3	9	19	52	0,20	0,32	0,15
	1&2&3	56	155	5	0,41	0,27	0,92
T5	1	74	190	84	0,35	0,28	0,47
	2	5	4	153	0,06	0,56	0,03
	3	4	8	154	0,05	0,33	0,03
	1&2	76	194	82	0,36	0,28	0,48
	1&3	75	198	83	0,35	0,27	0,47
	2&3	9	12	149	0,10	0,43	0,06
	1&2&3	77	202	81	0,35	0,28	0,49

TABLE IV
EXPERIMENTAL RESULTS

VI. RELATED WORK

Information extraction has come to be a hot research topic, especially since the availability of huge amounts of data publicly available. An excellent survey on Information Extraction is [22], where the author reviews all the significant existing approaches with a great amount of details. While many different approaches have been proposed, however to the best of our knowledge, little to no effort has been put towards the Italian language.

In [20] named entities are extracted and related to classified newspaper advertisements (in French), using different techniques. They make use of a lexicon to store already known entities, thus once a word is found in an advertisement and in the lexicon it can be automatically tagged as the lexicon suggests. They also use regular expressions for entities such as telephone numbers. Finally, a word spotting algorithm is used to compute a score for unrecognised words, based on the context (i.e. other specialised lexicons). While we also make use of a lexicon, we use it to exclude a candidate, after a rule has yielded one. It would be a trivial and brute force approach to recognise a location name using a lexicon with all existing location names (apart from homonymy), instead the rules we propose allow the discovery of names not already inserted in a lexicon.

The approach presented in [1] shows some similarities with ours. The authors start with sample patterns containing named entities, then identify actual instances of named entities, found names are searched for to automatically identify new patterns and reiterate the process.

A different approach has been proposed in [5], and try to identify named entities by short sequences of words, analysing n-grams statistics obtained on Internet documents. Their Lex method is a semi-supervised learning algorithm based on the assumption that a sequence of capitalised words compound the same name when such a n-gram appears to be statistically more frequent than simple chance.

A data mining approach is presented in [25], especially crafted for geographical names. The algorithm searches for specific keywords and patterns manually constructed and related to geographical names, such as *island of* or *archipelago*. The results are used to train a classifier with respect to the found instances of a pattern.

VII. CONCLUSIONS

We have presented an algorithm devised specifically for the Italian language, based on rules built upon its grammar. The

rules represent grammar pattern, implemented by finite state machines, typically used in both written and spoken language, thus several agents can be coordinated in a pipe and filter style to get an unstructured input text to be filtered by the rules to get candidate places. Preliminary results are promising, as the F1 score reaches a maximum of 0.67, whereas the highest precision and recall are 0.82 and 0.92, respectively.

As possible future work, we aim to connect with our previous research in which we have proposed to improve the modularity of a software system by letting classes assume roles on some design patterns [6]–[9]. The work presented here can foster an approach whereby the automatic processing of the italian language used for program comments can assist in the selection of roles for classes. Moreover, semantic analysis of text can take advantage of neural networks [15] and as a further work a possible approach would aim to recognise text fragments using a soft computing approach [13], [16].

ACKNOWLEDGEMENT

This work has been supported by project PRIME funded within POR FESR Sicilia 2007-2013 framework.

REFERENCES

[1] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of ACM Conference on Digital Libraries (DL)*, pages 85–94, New York, NY, USA, 2000. ACM.

[2] F. Bannò, D. Marletta, G. Pappalardo, and E. Tramontana. Tackling consistency issues for runtime updating distributed systems. In *Proceedings of International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8. IEEE, 2010.

[3] A. Calvagna and E. Tramontana. Delivering dependable reusable components by expressing and enforcing design decisions. In *Proceedings of Computer Software and Applications Conference (COMPSAC) Workshop QUORS*, pages 493–498. IEEE, July 2013.

[4] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Trans. on Knowl. and Data Eng.*, 18(10):1411–1428, Oct. 2006.

[5] D. Downey, M. Broadhead, and O. Etzioni. Locating complex named entities in web text. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2733–2739. Morgan Kaufmann Publishers Inc., 2007.

[6] R. Giunta, G. Pappalardo, and E. Tramontana. Using Aspects and Annotations to Separate Application Code from Design Patterns. In *Proceedings of Symposium on Applied Computing (SAC)*. ACM, 2010.

[7] R. Giunta, G. Pappalardo, and E. Tramontana. Aspects and annotations for controlling the roles application classes play for design patterns. In *Proceedings of Asia Pacific Software Engineering Conference (APSEC)*. IEEE, 2011.

[8] R. Giunta, G. Pappalardo, and E. Tramontana. AODP: refactoring code to provide advanced aspect-oriented modularization of design patterns. In *Proceedings of Symposium on Applied Computing (SAC)*. ACM, 2012.

[9] R. Giunta, G. Pappalardo, and E. Tramontana. Superimposing roles for design patterns into application classes by means of aspects. In *Proceedings of Symposium on Applied Computing (SAC)*. ACM, 2012.

[10] R. Giunta, G. Pappalardo, and E. Tramontana. A redundancy-based attack detection technique for java card bytecode. In *Proceedings of International WETICE Conference*, pages 384–389. IEEE, 2014.

[11] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[12] M. Mongiovi, G. Giannone, A. Fornaia, G. Pappalardo, and E. Tramontana. Combining static and dynamic data flow analysis: a hybrid approach for detecting data leaks in Java applications. In *Proceedings of Symposium on Applied Computing (SAC)*. ACM, 2015.

[13] C. Napoli, G. Pappalardo, and E. Tramontana. A hybrid neuro-wavelet predictor for qos control and stability. In *Proceedings of AIxIA*, volume 8249 of *LNCS*, pages 527–538. Springer, 2013.

[14] C. Napoli, G. Pappalardo, and E. Tramontana. Using modularity metrics to assist move method refactoring of large systems. In *Proceedings of Complex, Intelligent and Software Intensive Systems (CISIS)*. IEEE, 2013.

[15] C. Napoli, G. Pappalardo, and E. Tramontana. An agent-driven semantical identifier using radial basis neural networks and reinforcement learning. In *Proceedings of XV Workshop "Dagli Oggetti agli Agenti"*, volume 1260. CEUR-WS, 2014.

[16] C. Napoli, G. Pappalardo, and E. Tramontana. Improving files availability for bittorrent using a diffusion model. In *Proceedings of International WETICE Conference*, pages 191–196. IEEE, 2014.

[17] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI workshop on machine learning for information filtering*, volume 1, pages 61–67, 1999.

[18] G. Pappalardo and E. Tramontana. Automatically discovering design patterns and assessing concern separations for applications. In *Proceedings of Symposium on Applied Computing (SAC)*. ACM, 2006.

[19] G. Pappalardo and E. Tramontana. Suggesting extract class refactoring opportunities by measuring strength of method interactions. In *Proceedings of Asia Pacific Software Engineering Conference (APSEC)*, pages 105–110. IEEE, December 2013.

[20] R. A. Peleato, J.-C. Chappelier, and M. Rajman. Automated information extraction out of classified advertisements. In *Natural Language Processing and Information Systems*, pages 203–214. Springer, 2001.

[21] M. F. Porter. Snowball: A language for stemming algorithms, 2001. *URL http://snowball. tartarus. org/texts/introduction. html*, 2009.

[22] S. Sarawagi. Information extraction. *Found. Trends databases*, 1(3):261–377, Mar. 2008.

[23] E. Tramontana. Automatically characterising components with concerns and reducing tangling. In *Proceedings of Computer Software and Applications Conference (COMPSAC) workshop QUORS*. IEEE, 2013.

[24] E. Tramontana. Detecting extra relationships for design patterns roles. In *Proceedings of AsianPlop*. March 2014.

[25] O. Uryupina. Semi-supervised learning of geographical gazetteers from the internet. In *Proceedings of the HLT-NAACL 2003 Workshop on Analysis of Geographic References - Volume 1*, pages 18–25. Association for Computational Linguistics, 2003.