A Focused Crawler for Romanian Words Discovery

Ionut-Gabriel Radu, Traian Rebedea
Faculty of Automatic Control and Computers
University Politehnica of Bucharest
Email: radu.ionutz@hotmail.com, traian.rebedea@cs.pub.ro

***Abstract*—As all natural languages are subject to change over time and as the Web becomes more prevalent, it also constitutes a major source for identifying language evolution. Although these changes affect all linguistic branches ranging from phonetics, lexicon and grammar to semantics and pragmatics, we will focus only on discovering new potential words that entered the Romanian lexicon or alternative forms (lexicalizations) that are frequently used. In this paper we describe the architecture of a system which models the rate of Romanian vocabulary growth based on different statistics gathered by a focused web crawler. In order to validate the proposed system, the paper also presents the main new words identified by the system in online texts written in Romanian.**

Keywords*—*Neologisms Discovery, Focused Crawling, Text Processing, Language Identification

I. Introduction

Since its formation, the Romanian vocabulary has undergone various semantic and morphologic transformations. Although Romanian has developed on a Thraco-Dacian substratum and has been subsequently influenced by various languages (e.g. Slavic, Hungarian, Turkish), its Latin structure has not been modified. Today, due to large amounts of Romanian web content, the rate of change has increased significantly. This observation is also true for many other languages around the world. Our objective is to provide a mechanism to identify new words (e.g. neologisms) that entered the Romanian language using a specialized web crawler.

The paper continues with a section describing the technologies used by the focused crawler developed for identifying new Romanian words. Section 3 presents previous works on three main tasks related to our work: automatic discovery of neologisms, language identification and sentence segmentation. In section 4, we describe the system architecture and the main components of the crawler, while, in section 5 the most important results of using our system for new Romanian words discovery are highlighted. The paper is ending with a summary of our work and several concluding remarks.

II. Main Technologies

In this section, we outline the tools used to discover new Romanian words and the main steps which are required to achieve this goal.

A. *Specialized Web Crawlers*

A web crawler, which is frequently referred to as web spider or web robot, is a program primarily used to scan web pages to create an index of data. Regardless of their purpose, web crawlers consist of three major components: the *seed* which is a set of initial URLs (Uniform Resource Locators), the *closed* set, which includes all visited and expanded URLs and the *frontier* which is the list of unexpanded URLs. Usually, the abstract representation model is a *forest* of direct acyclic graphs and a set of *predicates* that defines the expansion step. General purpose web crawlers typically impose no restrictions on the expanded set of URLs, while the specialized (or focused) crawlers are designed to follow only web pages which are relevant to specific topics. The usual high level architecture of a focused web crawler includes a *scheduler*, a multi-threaded *downloader* and a *priority queue* (see Fig. 1) [1].

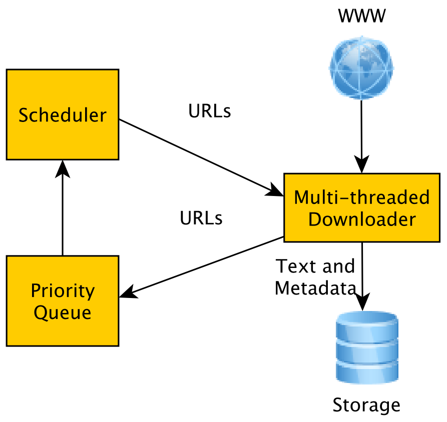

Fig. 1: Focused Web Crawler - High Level Architecture

B. *Text Processing*

In order to accomplish the goal of discovering new Romanian words, we implemented a text processing pipeline consisting of text normalization, language validation, sentence segmentation, sentence-level language identification, word tokenization, and spelling errors detection.

1) Text Normalization: This step involves transforming the text into a standard form which can be processed by the subsequent steps. It usually consists of eliminating irrelevant text such as multiple spaces/tabs/new lines, replacing erroneous diacritics (e.g. *s* with cedilla will be replaced by *s* with comma), etc.

2) Language Validation: We cannot entirely rely on URL patterns to discern that the web page content is written in Romanian because there are frequent cases where foreign pages contain Romanian texts. Worse still, there are cases when

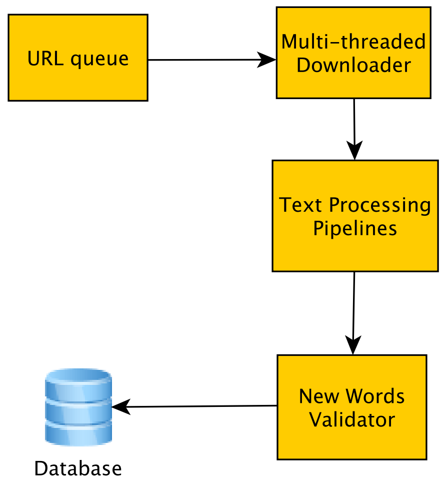

Fig. 2: RWScraper Web Crawler - High Level Architecture

Romanian and text in other languages are mixed in the same web page. In this case, we have to extract only the text with the highest probability to be written in Romanian.

3) Sentence Segmentation: In order to identify the context where new words are found, it is necessary to partition the input text into sentences. In most languages, this problem reduces to the identification of sentence boundaries. There are various approaches to solve this problem, ranging from simple tokenization to probabilistic detection of the first and last words in a sentence.

4) Sentence Level Language Identification: As mentioned earlier, language identification of larger portions of text (e.g. an entire web page or a paragraph) does not always produce the best results. As a consequence, we introduced an additional language identification step suited for classification of smaller blocks of text.

5) Word Tokenization: Another step in obtaining a fully tokenized text is word tokenization. The results of the word tokenization step are two types of tokens: character recognizable tokens (e.g. punctuation, numbers, dates) and morphological structures.

After performing all of these processing steps, our system - called *RWScraper (Romanian Word Scraper)* - is able to solve the following problems:

- Identify Romanian texts;
- Distinguish between proper names and common nouns;
- Create a database with new words along with context information and metadata. In order to identify new words, each word discovered in a Romanian text is looked in the database provided by www.dexonline.ro, which contains definitions from several Romanian dictionaries (DEX, DOOM, etc.);
- Discover the most frequent spelling errors in Romanian online texts.

RWScraper has been designed around the open-source *Scrapy* (www.scrapy.org) framework, which can be used to build custom web crawlers. Scrapy provides the basic infrastructure necessary to extract web content based on several user-defined rules. Fig. 2) presents a high-level architectural design of the system. The URL queue and the multi-threaded downloader, which employs Scrapy, are the main components of the crawling system, while the text processing pipeline and the new words validator are focusing the crawling process.

III. Related Work

In this section we concentrate on other approaches used to identify neologisms and/or new lexicalizations of existing words, together with solutions to the main natural language processing tasks involved in discovery of new words.

A. Neologisms Identification Techniques

Breen [2] proposes several methods for identifying neologisms in Japanese. The first one scans existing Japanese corpora for possible ”new” words, typically by processing the texts through segmentation software and dealing with the ”out-of-lexicon” problem. Then, he considers simulating the Japanese morphological processes to create new possible words and then test for the presence of them in large corpora.

1) Scanning Texts for Neologisms: This process involves segmentation, lemmatization, extraction of possible unrecorded words, examination of the words in the original textual contexts, and development of the reading and the meaning of the words. The scanning was concentrated on words written in *katakana* and rare *kanji* (two of the four writing systems in Japanese). The results showed that approximately 20% of words written in *katakana* were considered ”new”.

2) Generation of Possible Words: This approach involves generating Japanese morphological processes to synthesize potential words, then test if they appear in the lexicon or is in use in corpora. $2,900$ compound verbs were selected from a Japanese lexicon, the two verb portions extracted and $420,000$ potential compound verbs generated. Of these only $22,800$ were found to be in use.

In another study, Paryzek proposes another approach based on retrieval of phrases. This method involves identification of lexical discriminants (e.g. *termed*, *called*, *known as*) and punctuation discriminants (e.g. single and double quotes) [3]. However, this method is able to identify a significantly smaller number of potential new words as it uses a smaller corpus of documents (compared to the Web) and also due to the limited number of lexical discriminant patterns.

B. Language Identification Techniques

Here we outline some of the most important approaches to language identification in arbitrary texts.

1) Common Words Methods: This methods usually store the most frequent words for each language in a database. Prior to classification, the text is tokenized, and then compared to all words obtained in the first step. Using a scoring system, the word list with the most occurrences in that text indicates the language of the document [4]. This method is best suited for large documents and languages where tokenization is a trivial task [5].

2) Unique Letter Combinations: This method is closely related to common words approach. The difference lies in the fact that the database contains sequences of letters, not necessarily valid words. As in the previous case, the main disadvantage is the poor performance on short texts. The main advantage of this method over the common words approach is that it does not require word tokenization [6].

3) Language Identification Using N-Grams: This method starts from the premise that every language has several specific frequently used character n-grams. Let Σ be a finite set called *alphabet*, and Σ^* the set of all finite strings from alphabet Σ. A language L is defined as a subset of Σ^*. An n-gram is a sequence of n characters from $\Sigma \cup \{\$, \#\}$ (where \$ and $\#$ denote the word boundaries) that appear in at least one word in L.

In the first phase, we chose a *corpus* for each natural language L. Then, we perform normalization by deleting all non-alphabetic characters and redundant spaces. After this step, the corpus should have the following form: $w_1\#w_2\#...\#w_n$, with $w_i \in L$ and $\#$ is the blank character. In the next step for all $i \in \{2, ..., n\}$, we extract all possible i-grams for each word. Subsequently, the list of n-grams along with their frequencies are stored in an ordered dictionary. Usually, lower rank n-grams are first in the dictionary. Later, this process is repeated for each language L. For a particular language L, the n-gram ordered dictionary is called *n-gram profile*.

In order to identify the language of a given text, we create the n-gram profile for the text, and, compare it with all language profiles obtained in the previous steps, by computing the distance between them. The language profile that produces the smallest distance gives the language of the document.

The distance between two language profiles is calculated as follows. Let $p_1 : G_1 \to N$, and $p_2 : G_1 \to N$, $i \in \{1, 2\}$ be the language profile function, where G_i is the set of n-grams for each language. Let $\sigma : N \to S$ be a sampling function, where $S \subset N$ is a finite set, usually $S = \{0, 1, 2, ..., k\}$, that is monotonically increasingand $\sigma(0) = 0$.

We extend G_1, G_2, p_1, p_2 as follows: $G'_1 = G_1 \cup (G_2 \setminus G_1)$, $G'_2 = G_2 \cup (G_2 \setminus G_1)$, $p'_1 : G'_1 \to N$, with $p'_1(w) = 0, \forall w \in G_2$, and $p'_2 : G'_2 \to N$, with $p'_2(w) = 0, \forall w \in G_1$.

The distance D_{12} between two language profiles is defined as:

$$D_{12} = \sum_{w_1 \in G'_1, w_2 \in G'_2, w_1 = w_2} \sigma(|p'_1(w_1) - p'_2(w_2)|)$$

One of the most important advantages of language identification using n-gram approach is the high accuracy and that it does not require any linguistic knowledge [7].

4) Markov Models for Language Identification: The occurrence of letters in a word can be regarded as a stochastic process, and hence the word can be represented as a Markov chain where letters are states. Given a text document in a specific language, the transition and initial probabilities for all Markov chains representing all words in the text are calculated and the set of all probabilities is regarded as Markov model for that language. As in the case of unique letter combination approach, Markov language identification does not require text segmentation [8].

5) Teahan's Method: This method is an improvement to the Markov models approach. Being a supervised learning algorithm, this language identification method consists of a learning phase and an identification phase. The advantage of Teahan's approach is that it does not require text preprocessing, such as word tokenization [9].

C. Sentence Segmentation

Sentence segmentation is defined as the task of dividing the text into phrases/sentences. There are several classes of algorithms for sentence segmentation.

1) Regular Expresions and Heuristic Rules: The most widely used sentence segmentation method is a regular grammar, usually with a limited lookahead. It can range from simple implementations (regular expressions for sentence boundaries pattern) to sophisticated ones, that use exception lists (abbreviations) and surrounding context to determine sentence boundaries [10].

2) Segmentation using Regression Trees: Riley [11] proposed a method using regression trees for sentence boundaries detection according to several lexical features, such as probability of word preceding '.' occurs at the beginning or end of sentence, length of the word preceding and after the '.', case of word preceding and after the '.' (uppercase, lowercase, capital, number), punctuation after '.', and abbreviation class of word finishing with a '.'. This method uses the context on either side of the punctuation mark to compute probabilities. The results showed an accuracy of 99.8% for the Brown corpus.

IV. System Architecture

A. Problem Description

RWScraper is an integrated solution primarily designed for discovering new Romanian words. The source of discovery is the set of all available Romanian texts in the World Wide Web. We restricted the search for relevant Romanian documents to a limited number of domains. We opted of this solution because of several factors stated below:

- As the crawling process goes deeper, the probability of encountering foreign language texts grows and, thus, our system would consume most of its time on processing invalid texts (identifying texts not written in Romanian).
- The volume of garbled text is huge, many of the online texts written in Romanian lacking diacritics.
- We restrict our search to several formats: plain text, HTML and PDF.
- We start from a list of trusted sources in terms of correctness of the Romanian language use.

Apart from its main feature, RWScraper also offers the context in which the new words were discovered and a database of proper names identified in the crawled documents.

In the following subsections we will present the design overview and approaches to solve the encountered natural language processing tasks.

B. System Design Overview

As stated before, RWScraper is built upon the open source and easily customizable web crawling system *Scrapy*. The architecture of Scrapy consists of three key elements: items, spiders, and processing pipelines.

The *items* define the data we want to scrape. There is no restriction on the type of content accepted for an item. The main goal of these objects is to provide a way to define all metadata for a web resource in one place. *Spiders* are objects responsible for defining rules to restrict the crawled content to our area of interest. *Pipelines* deal with text processing tasks that act on the crawled web resources. The output of a pipeline is the input for the next one. RWScraper consists of five main pipelines which are called in the following order. First, the text normalization pipeline receives an item encapsulating the raw text downloaded from an URL and performs a couple of preprocessing tasks in order to reduce the garbled information. Then, the Romanian language validator rejects those texts with a low Romanian language score. In the sentence segmentation pipeline, the resulted text is split into phrases according to a segmentation algorithm. The sentence level processing task has two primary goals: sentence level Romanian language validation and word tokenization. At last, the word level processing pipeline aims to detect some frequent Romanian spelling errors such as replacing *â* with *î* or solving the problem of missing whitespaces.

In order to validate if we discovered a new Romanian word, first we search for it in the *dexonline.ro* database. This way we identify existing words in the Romanian language. To reduce the number of searches in the database, we extracted from a large corpus containing $33 million$ Romanian texts (European Parliament sessions transcripts) the most frequent $3,000$ words regardless of their form, constructed a dictionary of fixed size and designed a cache-like mechanism based on it.

In the next subsections we detail the text processing algorithms used by each pipeline.

C. RWScraper Language Validation Approach

First of all, we divide the texts into two categories: diacritics free texts - **DIAFREE** and genuine Romanian texts - **GEN**. They will be treated somehow differently by the language identification process.

Let $\Sigma = A \cup \{-, \#\}$, where $\#$ denotes the blank character, and A the set of all Romanian lower characters, be the *extended* alphabet and $T \subset \Sigma^*$ the set of all input texts. D is the set of all Romanian diacritics, and $X = \{k, q, w, y\}$ the set of Romanian rare characters. We also denote w_i the i-th character from left to right, w_{-i} the i-th character from right to left, and $len(w)$ the number of characters in a word or text, w.

We restrict the input texts to the following set, T', that is not allowed to start or finish with a space or to contain two subsequent space characters: $T' = \{w \in T | w_0 \neq \#, w_{-1} \neq \#, w_i w_{i+1} \neq \#\# \forall 1 \leq i < len(w) - 1\}$.

Before we effectively employ the language identification algorithm, it is required to perform two preprocessing tasks. First, the input text is changed to be in the canonical form permitted by T' (texts starting and ending with any character from $\Sigma \setminus \{\#\}$ and without two consecutive blank characters). Then we decide if the text is **DIAFREE** or **GEN**.

Our study of *ro_eu_parliament* corpus revealed that 6.40% of the characters in a Romanian text are diacritics. Thus in average, at every 100 characters, approximately 6 of them should be diacritics. From all Romanian texts with more than 500 characters, we analyzed 100% of them contained at least one character from D. One of the problems with this approach is that 4.14% of texts contained *ș*, *â*, and *î*. Unfortunately, there are also other languages that possess this set of diacritics or part of it. We also noted that Romanian is the only language that uses *ț* and *ă*. So, in order to identify a Romanian text, we must take into account only those texts that contain at least one *ț* and *ă*. Returning to our study, if we analyze 50 texts (1823 phrases) with more than 600 characters, 99.72% of them (1818 phrases) have at least one *ț* and *ă*. So, our assumption states that *if a text has over 600 characters and has no ț and ă are found, then it is* ***DIAFREE****, otherwise is* ***GEN***.

1) Profile Function: The next step in our algorithm is to calculate the *reference profile* of the Romanian language. Our profiles P are of the following pattern: $P = \langle N_1, N_2, N_3, F, \mu \rangle$, where $N_1 = \{(u, f) | u \in \Sigma, f \in [0, 1]\}$ is the unigram frequency, $N_2 = \{(b, f) | b \in \Sigma \times \Sigma, f \in [0, 1]\}$ the bigram frequency, $N_3 = \{(t, f) | t \in \Sigma \times \Sigma \times \Sigma, f \in [0, 1]\}$ the trigram frequency, $F = \{(w, f) | w \in Words(T), f \in [0, 1]\}$ and $\mu = \sum_{w \in Words(T)} \frac{len(w)}{|Words(T)|}$.

In case of **DIAFREE** texts, when calculating N_1, N_2, N_3 and F, we only consider characters from $\Sigma \setminus D$.

Additionally, we have $\sum_{(n,f) \in N_i} f = 1, i \in \{1, 2, 3\}$ and $\sum_{(w,f) \in F} f = 1$. We define the *profile function*, $\Phi : T \to P$, where T is the set of all preprocessed texts and P is the set of all language profiles of the form $\langle N_1, N_2, N_3, F, \mu \rangle$.

2) Evaluation Function: In language identification algorithms, the evaluation function takes a new text profile and measures how far it is from the reference language profile. Our approach performs evaluation of profiles as follows:

- **Step 0:** Let $r = \langle RN_1, RN_2, RN_3, RF, r\mu \rangle$ and $p = \langle N_1, N_2, N_3, F, \mu \rangle$ the reference and the input text profiles. We extend p in order to have all pairs that appear in the reference profile: $\bar{p} = \langle \bar{N_1}, \bar{N_2}, \bar{N_3}, \bar{F}, \mu \rangle$ with $\bar{N_i} = N_i \cup (RN_i \setminus N_i), i = \overline{1, 3}, \bar{F} = F \cup (RF \setminus F)$.
- **Step 1:** Calculate $Err_i = \sum_{(w_k, f_k) \in \bar{N_i}, (w_k, f_l) \in RF} |f_k - f_l|, i = \overline{1, 3}$ $Err_f = \sum_{(w_k, f_k) \in \bar{F}, (w_k, f_l) \in RF} |f_k - f_l|$ and $err\mu = |\mu - r\mu|$
- **Step 2:** Calculate $\sigma = vErr^T$, where $v = \langle k_1, k_2, k_3, k_4, k_5 \rangle$ and $Err = \langle Err_1, Err_2, Err_3, Err_f, err\mu \rangle$
- **Step 3:** Calculate $s = \sigma - B$, where B is the *bonus* value. We chose the bonus $B = 0.05$ for **GEN** and $B = -0.05$ for **DIAFREE** ones. The final decision is based on a discrimator. The value of the discriminator was chosen after a set of tests conducted on over 100 texts in 9 different languages, as will be detailed in section 5.

D. Sentence Segmentation

The next step of RWScrapper is sentence segmentation. Our algorithm is based on regular expressions and predefined lists. We use two lists: one containing abbreviations, and another one containing a list of words that are rarely found at the end of sentence (some prepositions, conjuctions, and adjectives).

E. Sentence Level Language Identification

The method described earlier gives accurate results for texts with more than 600 characters. Unfortunately, there are online texts which contain mixed paragraphs in Romanian and other languages. Therefore, if the Romanian score for the entire paragraph is not good enough, we also employ language identification at sentence level. The algorithm we propose has a similar idea to the previous approach. The difference lies in the number of metrics used in the language profiles. In this case, the following metrics are used:

- bigram and trigram frequencies;
- common words frequency;
- diacritics frequency: $\frac{c_w(d)}{len(w)-c_w(\#)}$, where $d \in D$, w is a string (possibly a sentence), $c_w(x)$ is the number of appearances of x in w;
- rare characters frequency: $\frac{c_w(x)}{len(w)-c_w(\#)}$, for $x \in X$;
- double consonant frequency;
- single quote frequency ('); we observed that many foreign languages use the apostrophe instead of the hyphen to link the clitics.

V. Results

Next we present several important results of RWScraper. First, we analyze the performance of the Romanian language validation module.

A. Accuracy of Language Validation Methods

In order to measure the accuracy of language validation methods, we conducted two types of tests:

- **off-line tests**: we chose 105 texts in different languages to perform accuracy tests.
- **real environment tests**: we computed the estimate frequency of foreign words discovered by RWScraper.

The 105 chosen texts are divided into: 20 Romanian with diacritics (RO1 - RO20), 20 Romanian without diacritics (RO21- RO40), 20 Italian, 15 English, 10 Spanish, 5 Latin, 5 French, 5 Turkish texts, 3 Catalan texts, and 2 Aromanian (dialect of Romanian, also known as Macedo-Romanian). The size of the texts varied from $9KB$ to $2.5MB$, the average size being $253.4KB$.

As shown in Table I, the off-line tests revealed some interesting results regarding language identification using the method described in section 4. The genuine Romanian texts (with diacritics) obtained an average score of 0.61, while Romanian texts without diacritics had an average of 0.74.

TABLE I: Average Romanian Language Score for Texts in Different Languages

RO	RO (-d)	IT	EN	ES	FR	TR	LA	CT	AR
0.61	0.74	0.80	1.03	1.03	0.87	1.07	1.17	0.92	0.88

TABLE II: Most Frequent New Words Containing î

Word	Number of Appearances
întodeauna	54
neîmplinirea	29
înduhovnicire	19
rîul	19
întodeuna	14
neînstare	11
întâi-născuți	11
voinî	10
întra-adevăr	8
împreună-slujitorul	6

These tests helped us determine the discriminant which can tell us that every text with a score below that value is Romanian. We set the discriminant to 0.77 since no texts with a score lower than that contained more than 50% foreign words. We also notice that Italian and Aromanian have the closest score to this discriminant. This could be explained by the fact that these two languages have a similar n-gram distribution. Latin and French, on the other hand, have higher scores, primarily due to the presence of 'q' in many words.

In the real environment test of the employed language identification method, from all $53,723$ newly discovered words, 5.27% of them show uncommon characteristics such as: the presence of k, q, w, y, double consonants and rare n-grams; all of them being susceptible to come from other languages.

B. *â/î* Correction

Analyzing all the new words discovered by RWScrapper ($53,723$ words), we observe that only 332 (0.61%) of them contain at least one î, while â was found in 623 words (1.15%). Of a total of 955 words that contain at least one â/î, 11.41% of them have î instead of â, while 3.35% have â instead of î. These results showed that a small portion of Romanian texts on Web uses *â/î* diacritics. The most frequent 10 new words discovered by RWScraper containing î are presented in Table II.

C. Clitics Detection Results

In Romanian, clitics are divided into two categories: hyphen-separated (e.g. *c-are*) and apostrophe-separated clitics (e.g. *d'ale*). The most frequent new words containing these two clitics characters are presented in Table III and Table IV. Of a total of $53,723$ new words discovered by RWScraper, 670 of them contain at least one apostrophe, while $11,247$ contain at least one hyphen.

D. RWScraper Results

In this subsection, we will present a series of statistics collected from the data obtained by RWScraper. First of all, we

TABLE III: The First 10 Most Frequently New Words Containing Hyphen

Word	Number of Appearances
nici-nici	1505
dobânda-cheie	625
orașului-ruină	206
w-end	205
semi-deschiși	203
luare-aminte	107
non-existentă	106
romano-bizantină	104
social-culturală	76
post-decembristă	76

TABLE IV: The First 10 Most Frequently New Words Containing Apostrophe

Word	Number of Appearances
douăș'cinci	384
dom'le	313
d'ale	207
pân'la	16
p'aia	13
d'antan	11
nu'ș	10
defetisto-je-m'en-fiche-ist	10
d'aia	10
tre'să	9

started from a set of 198 URL seeds and followed all the links where the domain is restricted to the 198 predefined domains, the type of file is PDF/HTML/TEXT, and the top level domains are .ro/.md/.org.

Then, we ran RWScraper for more than 50 hours scanning $264,328$ online documents, out of which only $12,555$ contained new words. From this set of texts, we extracted $698,341$ phrases and only $47,363$ phrases contained new words. In the end, we discovered $53,724$ new words.

From all the $53,724$ new words, RWScraper found $21,343$ proper names. The remaining tokens are common words and they are divided into the following main categories: misspelled words (approximately 35%), technical words (approximately 15%), and argotic words (approximately 10%). Clitics, regionalisms, archaisms, alternative forms for existing words account for the rest. The top 10 most common Romanian new words discovered by RWScraper are presented in Table V.

VI. Conclusions

RWScraper is a simple new Romanian words discovery system which was developed to also permit the users the possibility to classify the newly-discovered words and to write definitions for them. The project has also managed to create a large database of Romanian words extracted from the World Wide Web, offering us interesting statistics about common proper names, frequent spelling mistakes and newly-invented words.

To sum up, the system has successfully identified several thousand new words in Romanian online texts, together with their frequency of appearance. However, there are several elements that could be further improved, such as the accuracy of the NLP components used by the system and a more pertinent analysis of the words identified by the system.

TABLE V: Most Frequent 10 New Words Found by RWScraper (excluding Proper Names)

Word	Number of Appearances
click	5620
nici-nici	1505
inbox	1162
folder	821
usage	755
neputând	699
dobânda-cheie	625
blog	604
drone	598
check-in	576

Achnowledgement

This work has been funded by the Sectorial Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/132397.

References

[1] C. Castillo, "Effective web crawling," in *ACM SIGIR Forum*, vol. 39, no. 1. ACM, 2005, pp. 55–56.

[2] J. Breen, "Identification of neologisms in japanese by corpus analysis," in *E-lexicography in the 21st century: New challenges, new applications: proceedings of eLex 2009, Louvain-la Neuve, 22-24 october 2009*, 2010, pp. 13–21.

[3] P. Paryzek, "Comparison of selected methods for the retrieval of neologisms," *Investigationes Linguisicae*, vol. 16, pp. 163–181, 2008.

[4] S. Kranig, "Evaluation of language identification methods," *Bakalárska práca, Universität Tübingen, Nemecko*, 2005.

[5] G. Grefenstette, "Comparing two language identification schemes," 1995.

[6] G. Churcher, "Distinctive character sequences," *Personal communication*, 1994.

[7] W. B. Cavnar, J. M. Trenkle *et al.*, "N-gram-based text categorization," *Ann Arbor MI*, vol. 48113, no. 2, pp. 161–175, 1994.

[8] D. Tran and D. Sharma, "Markov models for written language identification," in *Proceedings of the 12th international conference on neural information processing*, 2005, pp. 67–70.

[9] W. J. Teahan, "Text classification and segmentation using minimum cross-entropy." in *RIAO*, 2000, pp. 943–961.

[10] D. D. Palmer, "Satz-an adaptive sentence segmentation system," *arXiv preprint cmp-lg/9503019*, 1995.

[11] M. D. Riley, "Some applications of tree-based modelling to speech and language," in *Proceedings of the Workshop on Speech and Natural Language*, ser. HLT '89. Stroudsburg, PA, USA: Association for Computational Linguistics, 1989, pp. 339–352. [Online]. Available: http://dx.doi.org/10.3115/1075434.1075492