An Adaptive Context-based Algorithm for Term Weighting
Application to Single-Word Question Answering

Marco Ernandes, Giovanni Angelini, Marco Gori, Leonardo Rigutini, Franco Scarselli
Dipartimento di Ingegneria dell'Informazione
Università di Siena, Siena - Italy
{ernandes,angelini,marco,rigutini,franco}@dii.unisi.it

Abstract

Term weighting systems are of crucial importance in Information Extraction and Information Retrieval applications. Common approaches to term weighting are based either on statistical or on natural language analysis. In this paper, we present a new algorithm that capitalizes from the advantages of both the strategies by adopting a machine learning approach. In the proposed method, the weights are computed by a parametric function, called *Context Function*, that models the semantic influence exercised amongst the terms of the same context. The Context Function is learned from examples, allowing the use of statistical and linguistic information at the same time. The novel algorithm was successfully tested on crossword clues, which represent a case of Single-Word Question Answering.

1 Introduction

Term weighting is an important task in many areas of Information Retrieval (IR), including Question Answering (QA), Information Extraction (IE) and Text Categorization. The goal of term weighting is to assign to each term w found in a collection of text documents a specific score $s(w)$ that measures the importance, with respect to a certain goal, of the information represented by the word. For instance, Passage Retrieval systems weigh the words of a document in order to discover important portions of text and to discard irrelevant ones. This is the case of applications as QA, Snippet Extraction, Keyword Extraction, Automatic Summarization.

Common approaches to term weighting can be roughly divided into two groups: statistical and linguistic techniques. The methods in the former group are based on a statistical analysis that extracts, from a document collection, features as word frequencies or information-theoretical measures. TFIDF [Salton and McGill, 1986] is a popular statistically-inspired method currently used in many IR systems to measure the importance of words. A major assumption in this approach is that the words of a document can be considered as unordered and independent elements. Such assumption is shared by many other measures as Information Gain, Gain Ratio, Best-Match (e.g. BM25) or the Chi-Square function [Manning and Schtze, 1999]. Even if the literature presents several interesting adaptations of TFIDF for specific problems (e.g. [Debole and Sebastiani, 2003]), to the best of our knowledge there has been no attempt to compute TFIDF-like term weights that make use of relationships among terms. Latent Semantic Indexing (LSI) [Deerwester *et al.*, 1990] is a statistical approach that does not suffer from the word-independence assumption. LSI tries to measure the principal associative patterns between sets of words and concepts using the Singular Value Decomposition technique, without taking into account word order. Unfortunately LSI cannot be employed in many practical problems. This is the case of QA, because LSI projects document words into a non-linguistic space, whereas QA requires answers in natural language.

On the other side, techniques that are more deeply inspired by natural language theories and applications [Voorhees, 1999], as morphological and syntax analysis, naturally exploit the information provided by word contexts. But their design is very difficult and require a large human effort.

In this paper, we present a novel algorithm for term weighting that aims to combine the advantages of both statistical and linguistic strategies by adopting a machine learning approach. Our method exploits the relationships (i.e. order, distance, etc.) among the words of a document in order to evaluate their semantic relevance to a given question. The intuition behind the proposed model is that the relevance of a term can be computed recursively as the combination of its intrinsic relevance and the relevance of the terms that appear within the same context. The influence exercised by a word on another one is computed using a parametric function, called *Context Function*. This function is trained by examples, thus it is well suited for using together statistical (e.g. term frequency) and linguistic information (morphological, syntactical, lexical), requiring no pre-designed rules.

The main idea underlining the proposed approach is tightly related to the TextRank algorithm [Mihalcea and Tarau, 2004] that has been succesfully applied to keyword extraction and text summarization. More precisely, the context-based algorithm extends TextRank, by allowing a more general modelling of the word relationships and by providing a learning algorithm to define the strength of those relationships.

The Context-based algorithm has been evaluated on a specific problem, i.e. Single-Word Question Answering, where the goal is to find the single correct word that answers a given question. Single-Word QA is particularly suited to evaluate a term weighting algorithm, since it directly exploits the rank

produced by the weights. The proposed method has been applied to improve the performances of a particular answering system (WebCrow [Ernandes *et al.*, 2005]), designed to address crossword clues. The results show that the approach is viable for the Single-Word QA problem and suggest that it could be extended to other fields of Text Processing.

Section 2 describes the Context-based term weighting algorithm. Section 3 presents the experimental environment used to evaluate the proposed method. In Section 4 we report the experimental results obtained by the system in the Single-Word QA problem. Section 5 provides our conclusions.

2 Context-based term weighting

The proposed method exploits the contexts of words. A word context primarily consists of the text surrounding a given word, but could also include other features, as document titles, hyper-linked documents, and so on. The idea is that, in order to measure the relevance of a word with respect to a certain goal (e.g. a query, a document category), the features of the context in which the term appears are important as well as the features of the word itself. Roughly speaking, a word is important not just when this is statistically strong, but also when the words that belong to its context are important too.

This recursive definition recalls that of social networks [Seeley, 1949], where the status of a person depends on the status of the persons that are somehow related to it. We assume that a text document can be represented by a social network, where the importance of the words can be computed on the basis of its neighbors. More precisely, the weight $s(w)$ of the word w is computed as

$$s(w) = (1-\lambda)d_w + \lambda \sum_{u \in r(w)} s(u)c_{w,u} \ , \qquad (1)$$

where d_w is the default score of w, $r(w)$ is the set of words that belong to the context of w, $c_{w,u}$ is a real number measuring the influence exercised by u over w, and $\lambda \in [0,1]$ is a damping factor. Thus, the score of a word will result from the combination of the default score d_w and its context score $\sum_{u \in r(w)} s(u)c_{w,u}$.

The real numbers d_w and $c_{w,u}$ can be calculated using information about the occurrences and the context of the words (the frequency and the linguistic role of a word, the distance between two words, and so on). The functions used to compute these values can be either predefined or learned by examples in a machine learning fashion. In our work, we assume that d_w is given by an initial scoring system (e.g. TFIDF). We then learn a parametric function for calculating the $c_{w,u}$ whose goal is to improve the initial values.

Since the d_w can be provided by external informative sources, the algorithm is well suited to exploit the weights produced by other algorithms in order to improve their performances. It is worth to notice that both d_w and $c_{w,u}$ can be computed exploiting statistical and linguistic features. From this point of view our method represents a mechanism to integrate both kinds of information.

In order to implement the proposed approach, the following problems have to be addressed: how to define a Context Function that computes the $c_{w,u}$ values, how to compute the term weights $s(w)$ using Eq.(1), how to automatically learn the Context Function from a set of examples and how to evaluate the performances of the system. These issues will be considered in the following sub–sections.

2.1 Context Functions

A document set contains multiple instances of words and each instance (a word-occurrence) is affected by a different context. Therefore, we distinguish between words (that we shall represent with w) and word occurrences (represented with a hat, $\hat{w}$). Here we assume that $c_{w,u}$ can be computed as the sum of the contributions of all the occurrences of w and u

$$c_{w,u} = \sum_{\hat{w} \in occ(w)} \sum_{\hat{u} \in ict(\hat{w},u)} \mathcal{C}_{\mathbf{p}}(\hat{w},\hat{u}). \qquad (2)$$

Here, $occ(w)$ is the set of instances of word w, and $ict(\hat{w},u)$ is the set of the occurrences of u that belong to the context of $\hat{w}$, i.e. $ict(\hat{w},u) = occ(u) \bigcap ctxt(\hat{w})$, where $ctxt(\hat{w})$ is the context of $\hat{w}$. Moreover, given a set of parameters $\mathbf{p}$, $\mathcal{C}_{\mathbf{p}}(\hat{w},\hat{u})$ is the parametric Context Function that establishes the strength of the influence between the instances $\hat{w}$ (the word under evaluation) and $\hat{u}$ (the context word). In our work this function is learned by examples[1].

The context of a word occurrence $\hat{w}$ can be defined as the set of words that are contained in the same document and within the surround of $\hat{w}$ with dimensions $k_L + k_R$ (left margin and right margin). Thus, if "$\ldots \hat{u}_{-k_l-1}\, \hat{u}_{-k_l} \ldots \hat{u}_{-1}\, \hat{w}\, \hat{u}_1 \ldots \hat{u}_{k_r}\, \hat{u}_{k_r+1} \ldots$" is a text passage of any document, then $ctxt(\hat{w}) = \{\hat{u}_{-k_l}, ..., \hat{u}_{-1}, \hat{u}_1, ..., \hat{u}_{+k_r}\}$ holds. In this formulation we introduce the strong assumption that direct semantic relations between words can be registered only within their surrounds. This assumption is plausible and widely adopted (e.g. snippets in search engines). On the other hand, the approach is easily extended to the cases when the context includes other features as the document title, the section title, the text of the anchors pointing to the document, and so on.

The most delicate point of the system is the definition of the function $\mathcal{C}_{\mathbf{p}}(\hat{w},\hat{u})$. This establishes how word couples can influence one another on the basis of features extracted from the words and from the relationships between words. Theoretically, the influence function can exploit any sort of information related to $\hat{w}$ and $\hat{u}$: information-theoretical (e.g. in text categorization: information gain, gain ratio), morphological (i.e. part-of-speech classes of w and u), syntactical (i.e. syntax role of w and u) or lexical (e.g. WordNet distance between w and u). The features that have been used in our preliminary experiments are exclusively statistical (see Tab. 1). Interestingly, even with this reduced list of basic features, the system displays notable performances.

Moreover, how do we merge all this information into one single output? The most general approach consists of implementing $\mathcal{C}_{\mathbf{p}}(\hat{w},\hat{u})$ by a modeling tool that has the universal approximation property, i.e. it allows to realize any function with any desired degree of precision. These tools include neural networks, polynomials, and rationales. While we plan to

[1] The novelty of our approach with respect to TextRank [Mihalcea and Tarau, 2004] consists of the introduction of the context function and the learning algorithm. In fact, TextRank uses an equation similar to (1) to define the word scores, but the influence factors $c_{w,u}$ are predefined values computed using word co-occurrences.

use neural networks in the future, for the introductory scope of this paper we preferred to exploit a simpler approach both for the implementation of $\mathcal{C}_{\mathbf{p}}(\hat{w}, \hat{u})$ and for the learning algorithm. Such a simplification allows to evaluate the proposed method on the term weighting problem, avoiding that the results are altered by the complexity of the system.

We defined the influence function as

$$\mathcal{C}_{\mathbf{p}}(\hat{w}, \hat{u}) = \prod_{i=0}^{i=n} \sigma(\alpha_i x_i + \beta_i)\,, \tag{3}$$

where x_i is the value associated with the i-th feature, the α_i, β_i are the model parameters, $\mathbf{p} = [\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n]$, and σ is the logistic sigmoid function $\sigma(x) = 1/(1 + e^{-x})$. Notice that the logistic sigmoid is a monotone increasing function that approaches 0, when $x \to -\infty$, and approaches 1, when $x \to \infty$. Thus, each term $\sigma(\alpha_i x_i + \beta_i)$ of Eq. (3) is a sort of soft switch related to the i-th feature and controlled by the parameters α_i and β_i. When $\alpha_i > 0$, the switch is "ON" (i.e. close to 1), if x_i is large and positive and it is "OFF" (i.e. close to 0), if x_i is large and negative. More precisely, the parameter α_i controls the steepness and the direction of the soft switch, while β_i controls the point where the switch assumes a medium value $1/2$. Finally, the whole function $\mathcal{C}_{\mathbf{p}}(\hat{w}, \hat{u})$ can be described as a simple boolean AND operation which is "ON" only if all the switches are "ON".

2.2 Computing and Learning Context Weights

For a correct definition of the weights, System (1) must have a unique solution. Stacking all the variables, Eq. (1) can be written as

$$\mathbf{S} = (1 - \lambda)\mathbf{D} + \lambda \mathbf{C}\mathbf{S}\,, \tag{4}$$

where $\{w_1, ..., w_n\}$ is the set the words of the dictionary, $\mathbf{S} = [s(w_1), ..., s(w_n)]'$ is the vector of the weights, $\mathbf{D} = [d_{w_1}, ..., d_{w_n}]'$, and $\mathbf{C} = \{c_{w,u}\}$ is a square matrix containing the word influence factors $c_{w,u}$.

It can be easily proved that, provided $\|\lambda\mathbf{C}\| < 1$ holds for any matrix norm $\|\cdot\|$, Eq. (4) has a unique solution which can be computed iterating the system[2] $\mathbf{S}^{t+1} = (1-\lambda)\mathbf{D} + \lambda\mathbf{C}\mathbf{S}^t$, i.e. $\mathbf{S} = \lim_{t\to\infty} \mathbf{S^t}$ for any initial $\mathbf{S^0}$. In practice, the computation can be stopped when the difference between the scores at iteration $t + 1$ and iteration t are below a specified small value ϵ: $||S^{t+1} - S^t|| < \epsilon$. The method is called Jacobi algorithm [Golub and Loan, 1996] for the solution of linear systems. It is worth to mention that this technique is extremely efficient and can be applied even on sparse matrices containing billions of variables [Page *et al.*, 1998].

The learning procedure can be implemented by any optimization algorithm that minimizes an error function $e_{\mathbf{p}}$, where $\mathbf{p} = [p_1, \ldots, p_n]$ denotes the set of parameters of the Context Function $\mathcal{C}_{\mathbf{p}}(\hat{w}, \hat{u})$. The function $e_{\mathbf{p}}$ measures the performance of the weighting system on a given dataset and it will be discussed in details in the following section. In our method the training algorithm repeats the following two steps for a predefined number of times:

1. compute the gradient $\frac{\delta e_{\mathbf{p}}}{\delta \mathbf{p}}$ of the error function with respect to the parameters;
2. adapt the parameters $\mathbf{p}$ by resilient method [Riedmiller and Braun, 1993].

The resilient parameter adaptation is a technique that allows to update the parameters using the signs of the partial derivatives of the error function. It has been experimentally proved that resilient is often more efficient than common gradient descent strategies [Riedmiller and Braun, 1993].

The gradient is approximated by a simple brute force algorithm. Each component of the gradient $\frac{\delta e_{\mathbf{p}}}{\delta p_i}$ is given by the corresponding incremental ratio, which requires to calculate the error function for $n + 1$ times at each training epoch. It is worth to mention that the proposed learning algorithm is a simplified version of the one described in [Gori *et al.*, 2005]. The latter procedure can be used efficiently even for models having a large number of parameters. Such a fact is important, since it guarantees that our implementation of the context function $\mathcal{C}_{\mathbf{p}}(\hat{w}, \hat{u})$ can be efficiently replaced by more complex and general models as, e.g. neural networks.

In our implementation we decided to learn, along with the set of parameters $\mathbf{p}$, the damping factor λ (see Eq. (4)), which, intuitively, establishes the level of confidence attributed by the system to the information extracted from the context. With this incorporation, a couple of improvements are expected. Firstly, the experiments that would be required to establish the best λ are avoided. Secondly, the new weights $\mathbf{S}$ are guaranteed not to be outperformed by the default scoring value $\mathbf{D}$. In fact, if the Context Function turns out to be deleterious for term ranking, λ can be set to 0 and hence the weighting system replicates the ranking performances of $\mathbf{D}$. During the experiments this case has not been observed.

2.3 Evaluating term weighting performances

The criterion to evaluate a term weighting system depends on the specific task that is addressed by the application. In QA-like problems, as the one proposed in this paper, the performance of the system depends on the position that the correct answer assumes in the candidate answer list. Two positional and cumulative measures can be used for QA evaluation: the Mean Reciprocal Rank of correct answers (MRR), which is a standard evaluation measure in the QA community, and the Success Rate (SR) at N, in which we annotate the probability of finding a correct answer within the first N candidates. Formally, given a set of questions $Q = \{quest(1), ..., quest(q), ..., quest(n)\}$ and denoted by $pos(a^q)$ the position of the correct answer a^q for question $quest(q)$, we have

$$\text{MRR}(Q) = \frac{1}{n}\sum_{q=1}^{q=n} \frac{1}{pos(a^q)} \tag{5}$$

$$\text{SR}(Q, N) = \frac{1}{n}\sum_{q=1}^{q=n} \Theta(N - pos(a^q))\,, \tag{6}$$

where Θ is Heaviside function defined by $\Theta(x) = 1$, if $x \geq 0$, and $\Theta(x) = 0$, otherwise.

The above performance measures can be used both for evaluating the weighting system and for implementing the

[2] In fact, $\mathbf{S^t} = (1-\lambda)\sum_{k=0}^{t-1} \lambda^{\mathbf{k}}\mathbf{C^k}\mathbf{D} + \lambda^{\mathbf{t}}\mathbf{C^t}\mathbf{S^0}$ holds by simple algebra. Thus, by power series convergence theorems, if $\|\lambda\mathbf{C}\| < 1$ is fulfilled, then $\lim_{t\to\infty} \mathbf{S^t} = (1 - \lambda)(\mathbf{I} - \lambda\mathbf{C})^{-1}\mathbf{D}$, which is the solution of Eq. (4), where $\mathbf{I}$ is the identity matrix.

learning procedure. A drawback of using positional measures is that they are discrete functions and thus non differentiable. In fact the values of MRR and SR change abruptly when the positions of two words are exchanged. A possible approach could be to adopt a learning scheme robust to discrete environments, e.g. simulated annealing or genetic algorithms. However, a major problem of these algorithms is their stochastic nature that makes the convergence very slow.

In order to adopt a differentiable error function, we defined a differentiable function $soft\_pos$ that replaces the concept of position pos

$$soft\_pos(a) = \sum_{w \in W,\, w \neq a} \sigma\left(\gamma(s(w) - s(a))\right) , \qquad (7)$$

where W is the set of words considered by the weighting algorithm, a is the word whose position value has to be calculated, and γ is a predefined parameter. Moreover, σ is a sigmoid function, i.e. a monotone increasing differentiable function such that $\lim_{x\to\infty} \sigma(x) = 1$ and $\lim_{x\to-\infty} \sigma(x) = 0$.

Each term contained in the sum of Eq. (7) compares the score of word a with the score of another word w. If $\gamma s(w)$ is very larger than $\gamma s(a)$, then the term assumes a value close to 1, and, in the converse case when $\gamma s(w)$ is very smaller than $\gamma s(a)$, the term approaches 0. Thus, for large γ, each terms approaches a Heaviside function and $soft\_pos(a)$ approximates $pos(a)$.

Based on Eq. (5) and (7) we defined a differentiable evaluation function which approximates MRR:

$$soft\text{MRR}(Q) = \frac{1}{|Q|} \sum_{q=0}^{q=Q} 1 \Big/ \sum_{i=0}^{i=|w|} \sigma\left(\gamma(s(w_i) - s(w_a))\right) \qquad (8)$$

Once the system is trained using this soft function, the evaluations can be done using the standard MRR (or SR).

3 Experimental Setup

3.1 The Single-Word QA problem

The problem on which the proposed approach was applied is a special case of Question Answering. In Single-Word QA each question has to be answered with a single correct word, given a number of documents related to the question. Since the target answer is a unique correct word the performance evaluations, unlike standard QA, do not require any human refereeing. To the best of our knowledge no standard dataset is available for the task of Single-Word QA, but a very popular and challenging example is provided by crosswords. Since crossword clues are intrinsically ambiguous and open-domain, they represent an very challenging reference point. It is worth mentioning that Single-Word QA was used as a testbed since it is particularly suited to evaluate term weighting. Actually, the answers of the QA system are directly dependent on the ranking produced by term weights.

The aim of our experiments is to show that the Context-based algorithm improves the ranking quality of the candidate answers provided by WebCrow [Ernandes *et al.*, 2005]. WebCrow is a Web-based QA system designed to tackle crossword clues for crossword cracking. At the core of this system there is a web search module, which, given a clue, retrieves useful documents using a search engine (currently Google) and extracts the candidate answers. These words are then ranked using statistical and morphological information.

Name	*Feature Set*
FS-A	$idf(w)$, $idf(u)$, $dist(\hat{w}, \hat{u})$
FS-B	$idf(w)$, $idf(u)$, $dist(\hat{w}, \hat{u})$, $dist(\hat{u}, Q)$
FS-B*	$idf(w)$, $idf(u)$, $dist(\hat{w}, \hat{u})$, $dist(\hat{u}, Q)$, *sw-list*

Table 1: The set of features inserted in the Context Functions used for the experiments. $dist(\hat{w}, \hat{u})$ is the number of separating words between occurrence $\hat{w}$ and $\hat{u}$. $dist(\hat{u}, Q)$ is the number of separating words between $\hat{u}$ and the closest occurrences of the words of query Q. *sw-list* indicates that a stopword list is used in order to avoid the presence of stopwords within the context window.

In the experiments we extended WebCrow with the addition of the the Context-based algorithm. The results show that the ranking of the correct answers is improved.

Our dataset consisted of 525 Italian crossword clues[3] randomly extracted from the archive in [Ernandes *et al.*, 2005]. The dataset has been divided into two subsets. On one side, *NE*-questions (165 examples), whose answers are exclusively factoid Named Entities, e.g. $\prec$*Real-life comic played in film by Dustin Hoffman:* **lennybruce**$\succ$. On the other side, *nonNE*-questions (360 examples), whose answers are non-Named Entities (common nouns, adjectives, verbs, and so on, e.g. $\prec$*Twenty four hours ago:* **yesterday**$\succ$).

3.2 Weighting techniques under comparison

In order to assess the effectiveness of the Context-based algorithm we compared its ranking performances with three different weighting techniques. The first is *TFIDF*, which gives a statistical evaluation of the importance of a word to a document. The other two techniques, *WebCrow-S* and *WebCrow-SM*, represent two ranking methods adopted by WebCrow. WebCrow-S exploits only statistical information of the words, the documents and the queries. This weighting scheme is a modified version of the one presented in [Kwok *et al.*, 2001]. WebCrow-SM additionally uses a morphological analysis of the documents joined with a morphological Answer Type classification of the clues. For all the details about WebCrow, see [Ernandes *et al.*, 2005].

4 Experimental Results

Train Set and Test Set. The two subsets (*NE* and *nonNE*-questions) were randomly divided into a train set and a test set, containing 40% and 60% of the examples respectively. The experiments were repeated 3 times for each configuration. Each example of the data set contained: i) a question/answer couple; ii) a set of H documents retrieved by Google (the documents are selected in Google's order); iii) the vector of ranked terms $\mathbf{W}$ (the candidate answer list) extracted from the H documents. In our experiments $\mathbf{W}$ contains words of any length.

Features. Several Context Functions were trained, each one with a different set of features. Table 1 presents the list of feature combinations adopted in the experiments. To fairly

[3]The dataset is available on http://webcrow.dii.unisi.it.

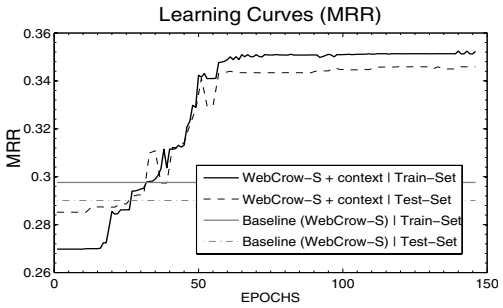

Figure 1: Learning curve of the Context Function on the *NE*-question problem with $softMRR$ and the FS-B feature configuration. WebCrow-S is outperformed after 34 epochs. After 60/70 epochs no sensible improvement can be observed.

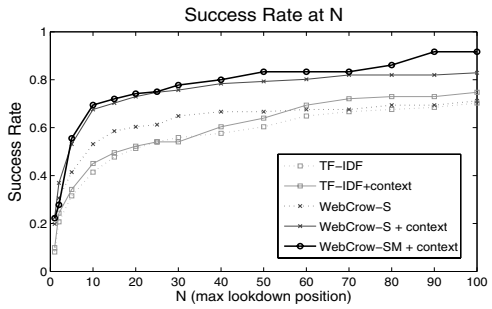

Figure 2: Success Rate at N for the *NE*-questions test set. The curves depict the probability (X-axis) of finding the correct answer to a question within the first N candidate answers (Y-axis). For clarity, the curve of WebCrow-SM is not reported.

Default TW	*Feature Set*	*NE*	*nonNE*
TFIDF		**0.216**	**0.071**
TFIDF + *context*	FS-A	0.233	-
WebCrow-S		**0.290**	-
WebCrow-S + *context*	FS-B	0.346	-
WebCrow-S + *context*	FS-B*	0.355	-
WebCrow-SM		**0.293**	**0.104**
WebCrow-SM + *context*	FS-B*	0.345	0.121

Table 2: MRR performance.The results of the three baseline algorithms are given in bold. Column *Default TW* denotes the algorithm used for the default term weights d_w, *Feature Set* specifies the Feature Set of the Context Function. The last two columns contain the MRR values obtained by the system on the two subsets of the testset. "*context*" indicates that the context weighting algorithm is used.

compare the Context-based weighting with the other techniques the features were restricted to those effectively used by each algorithm under comparison. For example, to reweight TFIDF scores we adopted a feature set, called FS-A (Tab. 1, first row), that exploits no information about the question. In this work no morphological information was integrated in the Context Functions. Nevertheless, these were able to improve also WebCrow-SM's performances, which adopts morphological analysis. For simplicity we chose 20 as a fix dimension for the context window, with $k_L = 10$ and $k_R = 10$.

Initialization and stop policy. The parameters of the Context Functions were initialized with small random weights, with the only exception of λ, that was initialized to 0.5. Each training session was stopped after 75 epochs. This stop policy was suggested by a prior observation of the variation curves over both the learning set and the test set (see Fig. 1). The resilient learning scheme guaranteed a fast convergence, requiring less than 60/70 epochs to reach good local minima.

EXP 1: Answering *NE*-questions. For the *NE*-questions problem the Context Functions were trained and tested using $H = 5$ as the number of documents used for extracting the word candidates. The results are presented in Table 2, which shows the performances in terms of MRR. The ranking performances of the compared methods are given in bold at row 1 (TFIDF), 3 (WebCrow-S) and 6 (WebCrow-SM).

The Context-based algorithm outperformed the compared weighting methods. In particular, the reweighting of the default scores provided by WebCrow-S, produced a 22.3% increment of the MRR (from 0.29 to 0.355, row 5).

An insightful resume of the performance improvements is provided by the Success Rate curve (Fig. 2), that shows the probability of finding the correct answer within the first N words of $\mathbf{W}$. Each weighting scheme under comparison appears consistently below the curve of its Context-based reweighting. In particular, with $N = 50$, the observed success rate was: TFIDF, 60.4%; TFIDF+context, 63.4%; WebCrow-S, 66.7%; WebCrow-S+context, 79.3%; WebCrow-SM, 75%; WebCrow-SM+context, 83.3%.

EXP 2: Answering *nonNE*-questions. Since the *nonNE*-questions represent a more difficult task, all the Context Functions were trained and tested with a higher number of documents, $H = 10$. The improvement (Tab. 2) of the ranking quality provided by the Context-based algorithm was less evident than on the *NE* subset, but still clear. This phenomenon depends on the different nature of these questions, which make improbable a strong increment of the MRR using exclusively statistical features, as revealed by the poor performance of the TFIDF measure (first row). The reweighting of the term scores provided by WebCrow-SM lead to a 16.3% increment of the MRR (from 0.104 to 0.121, row 6 and 7).

EXP 3: Varying the number of the Web documents (H). An interesting feature to evaluate is the ability of the Context Function to work with a number of documents that differs from that used in the learning phase. For this reason we tested the ranking quality of the Context Functions trained over the *NE* subset, with d_w given by WebCrow-SM (last two rows of Table 2), varying H. The Context-based algorithm proved to be robust to the changes of H, constantly outperforming WebCrow-SM (Fig. 3). With $H = 20$, the addition of the context increased the MRR by 15% (from 0.355 to 0.408).

An example. An example can help to better illustrate these promising results. WebCrow-SM answered the question: ≺*La paura nella folla:* **panico**≻ (≺*Fear among the crowd:* **panic**≻), with the correct answer in 16-th position. With the addition of the Context-based weighting, `panic` jumped to 2-nd position, and other related words appeared among the first 20 candidates, as: `collective`, `multitude`, `emergency` and `irrational`. A passage extracted from one of the documents may explain why the context information can result ef-

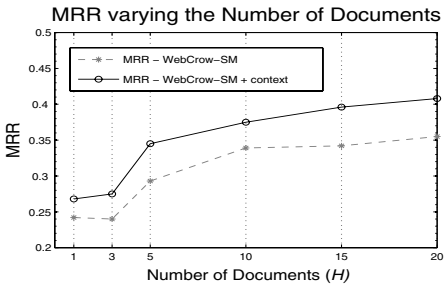

Figure 3: The MRR performance of the Context-based algorithm over *NE*-questions, in comparison with WebCrow-SM, varying the number of docs used in the test set.

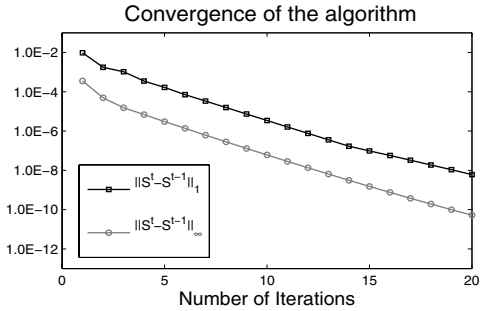

Figure 4: The convergence of the algorithm, a straight line on a log scale, is exponential. We report both 1-norm and ∞-norm of the score difference, between two subsequent iterations.

fective: "`...the panic is an irrational behavior of the crowd ...`". It is clear that the words `panic` and `irrational` can be strongly reinforced by the contextual presence of other relevant words, like `behavior` and `crowd`.

4.1 Computational Cost

The main drawback of the Context-based weighting method is due to the additional time cost introduced by the learning step, that is linear in the number of examples and in the number of iterations required for learning convergence. Nevertheless the training of the Context Functions can be successfully obtained with a small set of examples, as proved in the previous paragraphs, noticing that the dimension of the training set is independent with respect to the test set. The latter could be several order of magnitudes larger without requiring an increment of the learning set.

Leaving aside the learning phase and the initialization of the data used by the Context Function (i.e. word distances, idf, etc.), the theoretical computational requirement of the Context-based weighting algorithm is $O(|\hat{\mathbf{W}}|\times(k_l+k_r)\times T)$, where $|\hat{\mathbf{W}}|$ is the total number of word occurrences in the document set, (k_l+k_r) provides the dimension of the context window and T is the number of iterations that are required by the system to converge. In theory, the convergence of the Jacobian algorithm (Sec. 2.2) is exponential. We experimentally observed (Fig. 4) that the system converges in a small number of iterations, often requiring less than four steps to reach a maximum weight modification $\|\mathbf{S^t} - \mathbf{S^{t-1}}\|_\infty$ inferior to 10^{-5}. Thus, T does not introduce a great overhead.

5 Conclusions and Further work

In this paper we have proposed a novel term weighting method that exploits the information provided by the contexts in which the terms are found. We have defined a parametric function, called *Context Function*, that models the influence exercised by a word on another one that participates in the same context. Furthermore, we have presented a learning environment that allows us to adapt the parameters of the Context Function to the training data. The approach has been proved to be effective on the problem of Single-Word Question Answering, improving the answer-ranking performances of three different term weighting schemes. Future matter of research includes the application of the Context-based term weighting to other fields, as Document Classification and Information Extraction and the use of more complex implementation of the context functions.

References

[Debole and Sebastiani, 2003] F. Debole and F. Sebastiani. Supervised term weighting for automated text categorization. In *Proc. of SAC '03*, Melbourne, USA, 2003.

[Deerwester *et al.*, 1990] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[Ernandes *et al.*, 2005] M. Ernandes, G. Angelini, and M. Gori. Webcrow: A web-based system for crossword solving. In *Proc. of AAAI '05*, Pittsburgh, USA, 2005.

[Golub and Loan, 1996] G. H. Golub and C. F. Van Loan. *Matrix computations (3nd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[Gori *et al.*, 2005] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proc. of IJCNN '05*, Montreal, Canada, 2005.

[Kwok *et al.*, 2001] C. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. *ACM Trans. Inf. Syst.*, 19(3):242–262, 2001.

[Manning and Schtze, 1999] C. D. Manning and H. Schtze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, USA, 1999.

[Mihalcea and Tarau, 2004] R. Mihalcea, and P. Tarau. TextRank: Bringing Order into Texts. In *Proc. of EMNLP '04*, Barcelona, Spain, July 2004.

[Page *et al.*, 1998] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Proc. of ASIS '98*, Pittsburgh, USA, 1998.

[Riedmiller and Braun, 1993] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of ICNN '93*, San Francisco, USA, 1993.

[Salton and McGill, 1986] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[Seeley, 1949] J. R. Seeley. The net of reciprocal influence. *Canadian Journal of Psychology*, 3(4):234–240, 1949.

[Voorhees, 1999] E. Voorhees. Natural language processing and information retrieval. In *Proc. of SCIE '99*, Frascati, Italy, 1999.