PolarityRank: Finding an equilibrium between followers and contraries in a network

Fermín L. Cruz [*], Carlos G. Vallejo, Fernando Enríquez, José A. Troyano

Department of Computer Languages and Systems, University of Seville, Avda. Reina Mercedes, s/n. 41012 Seville, Spain

Keywords:
Ranking algorithms
Graphs
Relevance computing
Sentiment analysis
Data mining

A B S T R A C T

In this paper we present the relevance ranking algorithm named PolarityRank. This algorithm is inspired in PageRank, the webpage relevance calculus method used by Google, and generalizes it to deal with graphs having not only positive but also negative weighted arcs. Besides the definition of our algorithm, this paper includes the algebraic justification, the convergence demonstration and an empirical study in which PolarityRank is applied to two unrelated tasks where a graph with positive and negative weights can be built: the calculation of word semantic orientation and instance selection from a learning dataset.

1. Introduction

In the ancient world, an oracle was an answer obtained from a deity by the priests through the interpretation of physical signs of different nature, like the arrangement of stones or seeds tossed into the air, some weather phenomena or the outcome of an animal sacrifice. Somehow, the oracle brings to our world part of a deep knowledge available only to the gods, and materializes it in the form of answers for human questions.

The way we extract information from a network is very similar to the manner in which the oracles enabled our ancestors to communicate with the gods. In a network, the deep and unreachable knowledge is in the unfathomable search space made up by the overwhelming combinatorial of arcs and nodes. The graph analysis algorithms would be modern electronic priests that contact with the deity, and the results of the algorithms would be the desired oracles. In both cases humans play the same role: they want to know more.

As it occurs with an oracle, when working with a network to calculate metrics, indicators, clusters or any type of information, sometimes we have the feeling that there is more knowledge "in there" than what we have managed to extract. In those cases, we are forced to search for new algorithms that are able to access this knowledge still hidden.

In the area of network analysis, we can find a large variety of algorithms that compute local or global metrics for the different elements of a graph. One of the most widely applied measures is the relevance of the nodes, and one of the most used methods to calculate it is the ranking. The ranking algorithm par excellence is PageRank, used by Google for calculating the relevance of a web page in the Internet. Just as the authors establish "PageRank relies on the uniquely democratic nature of the web by using its vast link structure as an indicator of an individual page's value". The idea is to use links on pages as votes to other pages and make the votes of the most relevant pages have higher weight than those of the less relevant.

* Corresponding author. Tel.: +34 954556233; fax: +34 954557139.
E-mail address: fcruz@us.es (F.L. Cruz).

Despite the simplicity of the metric, PageRank has proven itself to be extremely powerful in various applications. But, what happens if we have votes of different signs? This occurs, for instance, in a social network when someone supports (positive vote) or refutes (negative vote) others arguments. Classical ranking algorithms such as PageRank cannot handle this information because they rely on a very basic voting concept in which only two options are available: support or lack of support. To benefit from this new kind of votes, we need to redefine the concept of relevance in terms of the new information given by the sign.

In this article we present a ranking algorithm that works with positive and negative votes. We have called it PolarityRank, and it has the ability to consider the sign of the ratings when determining the relevance of the nodes in a network. Our algorithm generalizes the idea of PageRank through the introduction of a special voting arithmetic. In this arithmetic, for example, a negative vote from a node that has bad reputation becomes positive in net terms.

The rest of the paper is organized as follows: in the second section, we present the foundations of the relevance ranking algorithms, on the third we present the PolarityRank algorithm along with its algebraic justification and the demonstration of its convergence. In section four we apply PolarityRank to two tasks: the calculation the semantic orientation of words and the instance selection in a learning dataset, showing the usefulness of PolarityRank in situations in which we have positive and negative evaluations in a network. Finally, the conclusions are drawn in section five.

2. Relevance ranking algorithms

The basic measures in social network analysis take into account the immediate environment of a node to determine its importance in the network. The main concept in these measures is the centrality that is calculated based on the number of contacts that a node has.

Unlike measures based on the centrality, the relevance-based rankings can get elements not belonging to the immediate neighborhood of a node to influence the calculation of its importance. This is achieved through the introduction of weighted voting, so a link received from an important node (which inherits, in turn, the importance of other nodes farther away) has more weight than one from a less important.

In this section we present three measures that rely on this idea of weighted voting according to the importance. Each one of them adds an interesting element, and in some way, there is a part of each one in our ranking algorithm.

2.1. PageRank

PageRank (Page, Brin, Motwani, & Winograd, 1998) is used to measure the importance of any Internet web page according to the links that page receives. Given a graph $G = (V,E)$ where V is a set of vertices and E a set of directed arcs between two vertices, two operations $In(V_i)$ and $Out(V_i)$ are defined first of all to calculate, respectively, the set of nodes with arcs entering or leaving the vertex V_i. From these two basic operations, we define the score (or PageRank) of a given vertex with the following formula:

$$PR(V_i) = (1-d) + d \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} PR(V_j),$$

where d is a damping factor that allows the convergence of the algorithm. In the context of navigation in the Internet, this factor represents the probability that a user accesses a page through a link at the current page, making $(1-d)$ the probability of that user jumping to a random page not linked to the current page. In the original definition of PageRank a value of 0.85 for factor d is recommended.

Starting from arbitrary values for the scores of the nodes of a graph, a convergence point is reached applying iteratively the formula until the largest difference in the scores obtained for each node, between two iterations, is under a certain threshold. Once the algorithm has finished, the score attained by each node represents the importance thereof, and may be used as a criterion for making decisions.

The previous formula is the most popular and also the most democratic version of PageRank. According to it, all nodes are in equality of conditions and the ranking assigned to each one of them depends exclusively on the topology of the network. However, the original formulation of PageRank is more general and includes a control parameter to influence externally in the calculation of the ranking. This more general formulation is:

$$PR(V_i) = (1-d)e_i + d \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} PR(V_j).$$

If the distribution of the values e_i is uniform, the importance of each node is the same and this formula behaves the same way as the previous one. However, if there are nodes with a bigger e_i, these nodes become sources of relevance, and contribute to their context with a constant amount of relevance at each iteration. We refer to this version as *biased PageRank*.

2.2. Hyperlink-Induced Topic Search (HITS)

HITS (Kleinberg, 1999) is an algorithm designed to value websites based on their links. Unlike PageRank, HITS calculates two values for each node: its *Authority* and its *Hub*. Both values are defined by mutual recursion and are calculated through an iterative process. The meaning of the *Authority* and *Hub* values for a given node is as follows:

- *Authority:* assesses how relevant is the node itself and it is calculated based on the *Hub* values of the links pointing to the node.
- *Hub:* assesses how relevant is the node as a recommender and it is calculated based on the *Authority* values of the nodes to which it points to.

The definitions of both values, which support the iterative algorithm for calculating the relevance, are the following:

$$Authority(V_i) = \sum_{j \in In(V_i)} Hub(V_j),$$

$$Hub(V_i) = \sum_{j \in Out(V_i)} Authority(V_j).$$

After each iteration, the values are normalized so that the following invariants are ensured:

$$\sum_{i=1}^{n} (Authority(i))^2 = 1, \quad \sum_{i=1}^{n} (Hub(i))^2 = 1.$$

As it occurs with PageRank, high values of *Authority* and *Hub* reflect a higher relevance, differing in this case the relevance from the point of view of the content of the node, and the relevance from the capacity of a node as a recommender. The intuitive justification of these definitions is the following: if a node points to many nodes with a high *Authority*, it must have a high value of *Hub*; in the same way, if a node is pointed to by many nodes with a high *Hub*, it should have a high value of *Authority*.

2.3. PageRank-based algorithms

There are many algorithms based on PageRank adapted to solve specific problems. TextRank (Mihalcea & Tarau, 2004, Mihalcea, Tarau, & Figa, 2004) adapts PageRank to problems related to Natural Language Processing. Independently of its applications, an interesting aspect is that TextRank generalizes the PageRank algorithm so that it can be applied to graphs with weighted arcs. In this case, the score of each node would be calculated as follows:

$$TR(V_i) = (1 - d) + d \sum_{j \in In(V_i)} \frac{p_{ji}}{\sum_{k \in Out(V_j)} p_{jk}} TR(V_j),$$

where p_{ji} is the weight of the arc that goes from the vertex V_j to V_i.

To apply this idea to the analysis of texts in natural language, it is only necessary to obtain a graph from a text, calculate the TextRank score and use that score to answer questions about the textual units to which the corresponding nodes are referred to. This algorithm has been applied to tasks such as keyword extraction and summary generation (Mihalcea & Tarau, 2004) or word sense disambiguation (Mihalcea et al., 2004) with very good results. In each case, the way how the graph is constructed is different. For example, in the keyword extraction the vertices are words, and arcs are established between vertices if the corresponding words appear close together in a sentence.

TrustRank (Gyöngyi, Garcia-Molina, & Pedersen, 2004) uses the biased PageRank algorithm to propagate a trust function through a web graph in order to detect possible web spam. The e_i values are used to manually introduce information about if some seeds are web spam or not. This approach, together with a seed selection method also based on PageRank, outperforms this last one in demoting spam web pages in the relevance ranking. Moreover, PopRank (Nie, Zhang, Wen, & Ma, 2005) adapts PageRank to rank *web objects* rather than web pages. *Web objects* are various kinds of objects embedded in the web pages: products, people, papers, organizations, etc. As there are different types of relationships between web objects, and each one of these types is not equally important in order to compute the relevance of the objects, PopRank defines a *popularity propagation factor* for each type of relationships. This factor is equivalent to a weight assigned to each arc of a certain type, being the PopRank formulation very similar to the TextRank formulation.

As we shall see later on, these ideas (allowing weighted arcs and using a priori infomation) will be very important in the formulation of PolarityRank.

3. PolarityRank

In some problems, there are positive and negative links between entities (e.g. in on-line social networks or in recommender systems). The ranking algorithms explained in Section 2 cannot be directly applied to this type of graphs. One could apply the algorithm in two passes, removing in each case the positive or negative edges, and combining in some way the relevance values obtained. However, some authors suggest that "it is often important to view positive and negative links in an on-line system as inter-related, rather than as distinct noninteracting features of the system" (Leskovec, Huttenlocher, & Kleinberg, 2010). This is exactly what our algorithm aims to accomplish.

Our ranking algorithm takes from PageRank the idea that the relevance of a node depends on the relevance of the nodes that point to it, as with TextRank and PopRank it allows us to associate weights to the arcs, and resembles HITS on the fact that two mutually dependent measures are calculated simultaneously. The main idea behind PolarityRank is to calculate two measures of relevance, one positive and one negative for each node in the graph. Regardless of the concrete meaning of "positive" and "negative", which depends on the problem we are modeling, the relevances depend on one another following a special *influences arithmetic*. According to this arithmetic, and similarly to what happened in PageRank, the positive relevance of a node n is increased proportionally to the positive relevance of the nodes connected to n by edges of positive weights. But also, the positive relevance of n is increased in proportion to the negative relevance of those nodes connected to n with edges of negative weights. The same principles would apply to the negative relevance values of the nodes.

In the following sections the formal definition of PolarityRank, its algebraic justification and the demonstration of the convergence of its calculation are presented.

3.1. Definition

Let $G = (V,E)$ be a directed graph where V is a set of nodes and E a set of directed edges between two nodes. Each edge of E contains an associated real value or weight, distinct from zero, being p_{ji} the weight associated with the edge going from node v_j to v_i. The operation $Out(v_i)$ is defined, returning the set of indices j of the nodes for which there exists an outgoing edge from v_i. Operations $In^+(v_i)$ *y* $In^-(v_i)$ are defined, returning the sets of indices j of the nodes for which there exists an incoming edge to v_i whose weight is positive or negative, respectively. We define the positive and negative PolarityRank of a node v_i (formula (1)), where the values of e_i^+ are greater than zero for certain nodes that act as positive seeds and zero for the other nodes, and the values e_i^- are greater than zero for certain nodes that act as negative seeds and zero for the rest of nodes.

$$\begin{aligned}
PR^+(v_i) &= (1-d)e_i^+ + d\left(\sum_{j\in In^+(v_i)} \frac{p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|} PR^+(v_j) + \sum_{j\in In^-(v_i)} \frac{-p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|} PR^-(v_j)\right) \\
PR^-(v_i) &= (1-d)e_i^- + d\left(\sum_{j\in In^+(v_i)} \frac{p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|} PR^-(v_j) + \sum_{j\in In^-(v_i)} \frac{-p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|} PR^+(v_j)\right)
\end{aligned} \tag{1}$$

We propose that the sum of the values of e_i^+ on the one hand and e_i^- on the other equals the number of nodes in the graph; this way we obtain scores with magnitudes similar to those of the original PageRank algorithm.

3.2. Algebraic justification

We will study now the algebraic foundation of PolarityRank. Let $n = |V|$ be the number of nodes in the graph. We denote $\boldsymbol{P}$ as the matrix of the (p_{ij}).

Let us define $p_j = \sum_{k\in Out(v_j)}|p_{jk}|$, i.e. the sum of the weights of the edges starting at v_j. In the matrix $\boldsymbol{P}$, p_j is the sum of the absolute values of the row j; p_j can be written as $p_j = \sum_{k=1}^{n}|p_{jk}|$.

The PolarityRank can now be expressed as

$$\begin{aligned}
PR^+(v_i) &= (1-d)e_i^+ + d\left(\sum_{j\in In^+(v_i)} \frac{p_{ji}}{p_j} PR^+(v_j) + \sum_{j\in In^-(v_i)} \frac{-p_{ji}}{p_j} PR^-(v_j)\right) \\
PR^-(v_i) &= (1-d)e_i^- + d\left(\sum_{j\in In^+(v_i)} \frac{p_{ji}}{p_j} PR^-(v_j) + \sum_{j\in In^-(v_i)} \frac{-p_{ji}}{p_j} PR^+(v_j)\right).
\end{aligned}$$

Let us define now the matrix $\boldsymbol{Q} = \boldsymbol{P}^t$, i.e. the transpose of P (if the matrix $\boldsymbol{P}$ describes an undirected graph then $\boldsymbol{Q} = \boldsymbol{P}$). Call $q_j = \sum_{k=1}^{n}|q_{kj}|$, i.e. the sum of the elements of column j from $\boldsymbol{Q}$. Obviously, $p_j = q_j$.

Define now two matrices $\boldsymbol{Q}^+ = (q_{ij}^+)$ and $\boldsymbol{Q}^- = (q_{ij}^-)$ as

$$q_{ij}^+ = \begin{cases} q_{ij} & \text{if } q_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$q_{ij}^- = \begin{cases} -q_{ij} & \text{if } q_{ij} < 0 \\ 0 & \text{otherwise,} \end{cases}$$

q_j can be seen as the sum of the elements of column j from matrix $\boldsymbol{Q}^+$ plus the sum of the elements of the same column from the matrix $\boldsymbol{Q}^-$.

We can now express PolarityRank as follows:

$$PR^+(v_i) = (1-d)e_i^+ + d\left(\sum_{j=1}^{n}\frac{q_{ij}^+}{q_j}PR^+(v_j) + \sum_{j=1}^{n}\frac{q_{ij}^-}{q_j}PR^-(v_j)\right)$$
$$PR^-(v_i) = (1-d)e_i^- + d\left(\sum_{j=1}^{n}\frac{q_{ij}^-}{q_j}PR^+(v_j) + \sum_{j=1}^{n}\frac{q_{ij}^+}{q_j}PR^-(v_j)\right).$$

We define the matrix $\boldsymbol{A}^+ = (a_{ij}^+)$ as $a_{ij}^+ = q_{ij}^+/q_j$ and the matrix $\boldsymbol{A}^- = (a_{ij}^-)$ as $a_{ij}^- = q_{ij}^-/q_j$, then

$$PR^+(v_i) = (1-d)e_i^+ + d\left(\sum_{j=1}^{n}a_{ij}^+PR^+(v_j) + \sum_{j=1}^{n}a_{ij}^-PR^-(v_j)\right)$$
$$PR^-(v_i) = (1-d)e_i^- + d\left(\sum_{j=1}^{n}a_{ij}^-PR^+(v_j) + \sum_{j=1}^{n}a_{ij}^+PR^-(v_j)\right).$$

Next we are going to make some definitions to simplify the notation. Let $m = 2n$. We define the vector $\boldsymbol{x}$ of $m \times 1$ elements as

$$\boldsymbol{x} = \begin{bmatrix} PR^+(v_1) \\ \vdots \\ PR^+(v_n) \\ PR^-(v_1) \\ \vdots \\ PR^-(v_n) \end{bmatrix}$$

and the vector $\boldsymbol{e}$ of $m \times 1$ elements as

$$\boldsymbol{e} = \begin{bmatrix} e_1^+ \\ \vdots \\ e_n^+ \\ e_1^- \\ \vdots \\ e_n^- \end{bmatrix}.$$

Finally, we define the matrix $\boldsymbol{A}$ of $m \times m$ elements

$$\boldsymbol{A} = \left[\begin{array}{c|c} \boldsymbol{A}^+ & \boldsymbol{A}^- \\ \hline \boldsymbol{A}^- & \boldsymbol{A}^+ \end{array}\right].$$

With the previous auxiliary vectors and matrices, we can write the PolarityRank Eq. (1) much more easily as

$$\boldsymbol{x} = (1-d)\boldsymbol{e} + d\boldsymbol{A}\boldsymbol{x}. \tag{2}$$

$\boldsymbol{A}$ is a stochastic matrix: All the elements of $\boldsymbol{A}$ are between 0 and 1 (both inclusive):

$$a_{ij}^+ = \frac{q_{ij}^+}{\sum_{k=1}^{n}q_{kj}^+ + \sum_{k=1}^{n}q_{kj}^-} \quad \text{and} \quad a_{ij}^- = \frac{q_{ij}^-}{\sum_{k=1}^{n}q_{kj}^+ + \sum_{k=1}^{n}q_{kj}^-},$$

and obviously, the sum of the elements of each column is 1.

e_i^+ and e_i^- have been chosen so that $\sum_{i=1}^{n}e_i^+ = \sum_{i=1}^{n}e_i^- = n$ and positive. Thus $\|\boldsymbol{e}\|_1 = m$.

We now define the vector with $m \times 1$ elements $\boldsymbol{f} = \boldsymbol{e}/m$ (i.e., the elements of e_i^+ and e_i^- divided by m), and the vector $\boldsymbol{u}$ as the vector with $m \times 1$ ones.

Let us analyze the matrix $\boldsymbol{f}\boldsymbol{u}^t$:

$$\boldsymbol{f}\boldsymbol{u}^t = \begin{bmatrix} e_1^+/m \\ \vdots \\ e_n^+/m \\ e_1^-/m \\ \vdots \\ e_n^-/m \end{bmatrix} [1 \quad \ldots \quad 1] = \begin{bmatrix} e_1^+/m & \ldots & e_1^+/m \\ \vdots & & \vdots \\ e_n^+/m & \ldots & e_n^+/m \\ e_1^-/m & \ldots & e_1^-/m \\ \vdots & & \vdots \\ e_n^-/m & \ldots & e_n^-/m \end{bmatrix}.$$

Its elements are between 0 and 1, and the sum of the elements in each column is 1, so $\boldsymbol{fu}^t$ is a stochastic matrix.

If $\boldsymbol{x}$ is normalized so that $\|\boldsymbol{x}\|_1 = m$ then $\boldsymbol{u}^t\boldsymbol{x} = m$, provided that the elements of $\boldsymbol{x}$ are positive; but if we start giving non-negative values, they will always remain non-negative since the elements of $\boldsymbol{A}$ are also non-negative.

The Eq. (2) can then be written as:

$$\boldsymbol{x} = (1-d)\boldsymbol{e} + d\boldsymbol{Ax} = (1-d)\boldsymbol{e}\frac{1}{m}\boldsymbol{u}^t\boldsymbol{x} + d\boldsymbol{Ax} = (1-d)\boldsymbol{fu}^t\boldsymbol{x} + d\boldsymbol{Ax} = ((1-d)\boldsymbol{fu}^t + d\boldsymbol{A})\boldsymbol{x}.$$

If we name $\boldsymbol{B} = ((1-d)\boldsymbol{fu}^t + d\boldsymbol{A})$, the expression defining the PolarityRank (1) can then be written as

$$\boldsymbol{x} = \boldsymbol{Bx}. \tag{3}$$

$\boldsymbol{B}$ is the convex combination (linear combination where the two coefficients are positive and sum to 1) of two stochastic matrices, then $\boldsymbol{B}$ is also stochastic. Hence, as a result of the Perron–Frobenius theorem, the Eq. (3) has the eigenvalue 1, and the rest of its eigenvalues have modulus less than 1 (may be complex). The previous system has a solution that is just the eigenvector corresponding to eigenvalue 1.

3.3. Convergence

The calculation of PolarityRank (the $\boldsymbol{x}$ vector) can be done from the expression (1) by an iterated calculation that we will demonstrate that converges.

As we have seen, the expression (1) equals the expression (2): $\boldsymbol{x} = (1-d)\boldsymbol{e} + d\boldsymbol{Ax}$. Let us express $\boldsymbol{x}_k$ as the k-th term of the PolarityRank iterative calculation:

$$\boldsymbol{x}_{k+1} = (1-d)\boldsymbol{e} + d\boldsymbol{Ax}_k$$

and

$$\boldsymbol{x}_k = (1-d)\boldsymbol{e} + d\boldsymbol{Ax}_{k-1},$$

therefore

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| = d\|\boldsymbol{A}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})\| \leqslant d\|\boldsymbol{A}\|\|(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})\|$$

for any norm compatible with the vector and matrix. For example, using norm 1,

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{m}|x_i|$$

and

$$\|\boldsymbol{A}\|_1 = \max_{j=1\ldots m}\sum_{i=1}^{m}|a_{ij}|.$$

In this case, $\|\boldsymbol{A}\|_1 = 1$ (recall that it is stochastic) and $d < 1$, then $\lim_{k\to\infty}\|x_{k+1} - x_k\| \to 0$, thus convergence is guaranteed.

4. Experimental applications

To study the behavior of the PolarityRank algorithm, we experienced with its application to two significantly distinct problems: the calculation of the semantic orientation of adjectives and the selection of instances in learning datasets. These problems have been chosen due to the fact that they can be modeled by means of graphs, constructed with different intentions and meanings, but involving edges with positive and negative weights that encode positive and negative influences between the nodes. These problems are therefore affordable by applying ranking algorithms on graphs with the added difficulty of the existence of edges with negative weights.

Our interest is to assess the quality of the relevance calculated by PolarityRank and compare it with the relevance obtained with PageRank. To do this, we start in both cases from a graph in which there are positive and negative edges, and perform two types of experiments:

- PageRank (*PR*): we calculate the relevance applying PageRank (with weights) to the corresponding graph once discarded the edges that have negative weights.
- PolarityRank ($PR^{\pm}$): we calculate the relevance applying PolarityRank to a graph with positive and negative edges.

In the following sections we include for each task its definition, a brief description of the technique used to resolve it and some empirical results showing the positive effect of including the information given by the PolarityRank in the solution. We also compare our results with other well-known techniques for each problem.

Direct relations:

The camera has a **bright** *and* **accurate** len.
It is a **marvellous**, really **entertaining** movie.
. . . **clear** *and* **easy to use** interface.
. . . **easy to get information**, **user-friendly** interface.

Inverse relations:

The camera has a **bright** *but* **inaccurate** len.
It is a **entertaining** *but* **typical** film.
The driving is **soft** *and not* **aggresive**.

Fig. 1. Examples of direct and inverse relationships extracted from the corpus.

4.1. Semantic orientation

Within the discipline of Natural Language Processing (NLP), the works related to texts with subjective content, in which opinions or emotions are flushed, are forced to deal with some additional difficulties, besides the problems common to the rest of the tasks of NLP. In recent years, the interest of the community in these tasks, collectively known as *Opinion Mining* or *Sentiment Analysis* (Pang & Lee, 2008), has increased. One of the most interesting applications is to build systems that are able to extract opinions from product reviews (e.g. a camera), detecting the concrete object of each opinion (e.g., the quality of the image), and the polarity and intensity of it. This process, which can be understood as a specialization of the information extraction systems, provides a very valuable structured information about the opinions of the users of a particular product. This information can be used in the construction of summaries or various kinds of reports, useful both for consumers and producers.

A key problem to solve in this type of system is the calculation of the semantic orientation of the words that participate in the extracted opinions. The semantic orientation of a word is a measure of the emotional implications, positive or negative, of that word when being used in a subjective context. The semantic orientation is usually represented by a real number whose sign indicates the polarity of the opinion involved, and whose magnitude indicates its intensity. For example, the adjective *excellent* has a positive semantic orientation of great intensity. If we chose to express it through a real number between -1 and 1, we could use a value close or equal to 1. Similarly, we would probably assign to the adjective *awful* a value close or equal to -1, while *acceptable* would have a positive value, but of lesser magnitude than *excellent*.

Different authors have conducted experiments to build opinion lexicons, resources where adjectives or other type of words are collected along with semantic orientation values calculated automatically. Some of them (Hu & Liu, 2004; Kamps, Marx, Mokken, & De Rijke, 2004; Esuli & Sebastiani, 2007) are based on lexical resources such as WordNet (Fellbaum, 1998). (Hatzivassiloglou & McKeown, 1997) instead is based on cooccurrences of adjectives in a corpus: the observation of two adjectives coordinated by the conjunction *and* suggests that both share the same semantic orientation; by analogy, the coordination of adjectives using *but* suggests contrary semantic orientations.

Using this same principle, we have experienced with the application of the PolarityRank algorithm to the construction of a lexicon with semantic orientations of adjectives.

4.1.1. Experimental setup

As a basis for the automatic construction of the lexicon, we used product reviews of various kinds taken from the website *Epinions*.[1] The corpus contains nearly 234,000 reviews totaling more than 134 million words.

Using simple extraction patterns, we detected the conjunctive relations between adjectives that appear in the reviews. We say that the conjunctive relation is *direct* when both adjectives share the same semantic orientation, and *inverse* when they have opposite semantic orientations (the relationship is direct or inverse depending on the conjunction used and the occurrence or not of negations before the adjectives). In Fig. 1, some examples of direct and inverse conjunctive relations observed in the corpus are shown.

Starting from the extracted conjunctive constructions we generate a graph as follows. Adjectives that participate in these constructions will be the nodes. Nodes corresponding to a pair of adjectives that participate in the same conjunctive construction will be connected by two edges, one in each direction. The weights of the edges will indicate the number of occurrences of the relationship, and the sign if the relationship is direct or inverse. Thus, if we have two nodes connected by a pair of positive edges, it is understood that each adjective is voting to assign the other adjective a semantic orientation similar to their own. Similarly, if two nodes are connected with negative edges, both adjectives are voting to assign the opposite semantic orientation to his teammates.

Now we just have to select some nodes that will work as positive and negative seeds.[2] For each node i corresponding to a positive seed, we assign a nonzero value to e_i^+. Similarly we proceed with the e^- of the nodes corresponding to negative seeds. The e^+ and e^- for the other nodes are equal to zero.

1 www.epinions.com.

2 Positive seeds: good, nice, excellent, positive, fortunate, correct, superior. Negative seeds: bad, nasty, poor, negative, unfortunate, wrong, below.

From the values PR^+ and PR^- obtained, we calculate the semantic orientation of each node as the normalized difference of these values:

$$SO(n) = \frac{PR^+(n) - PR^-(n)}{PR^+(n) + PR^-(n)}. \tag{4}$$

To measure the goodness of the values we obtain, we use the Micro-WNOp resource (Cerini, Compagnoni, Demontis, Formentelli, & Gandini, 2007) as *gold standard*. It consists of a sample of about 1100 synsets of WordNet, each of which were assigned manually three values between 0 and 1 indicating positivity, negativity, and neutrality. We use as semantic orientation the difference between the values of positivity and negativity. Only those Micro-WNOp adjectives found in our lexicon are used as *gold standard*. Ordering the list of adjectives according to the values of the semantic orientation of the *gold standard* and those calculated by our method, we get two rankings. We compare both rankings by the *Kendall distance* τ_p (Fagin, Kumar, Mahdian, Sivakumar, & Vee, 2004), which is calculated by the following formula:

$$\tau_p = \frac{n_d + p * n_u}{Z}, \tag{5}$$

being n_d the number of discordant pairs (ordered pairs in the *gold standard* and not ordered in the candidate *ranking*), n_u the number of ordered pairs in the *gold standard* and the same in the other ranking, and p a penalty factor assigned to each of these pairs. Z is the total number of ordered pairs in the *gold standard*. We use a value of $\frac{1}{2}$ for the penalty factor. The closer the result of this calculation is to zero, the more similar to the *ranking* provided by Micro-WNOp will be the *ranking* obtained from our lexicon.

To assess whether the use of the PolarityRank algorithm is an improvement respect to the original PageRank algorithm, we have conducted experiments in which we apply PageRank to the graph in two rounds. In a first round, we assign nonzero values of e to the nodes corresponding to positive seeds, and thus calculate the positive PageRank values. Then, we assign values of e distinct from zero to the nodes corresponding to negative seeds, and calculate negative PageRank values. With these two values we calculate the semantic orientation using the same formula above (formula (4)). Of course, the negative edges were not taken into account in these experiments, since the PageRank algorithm does not consider them.

We have also computed the semantic orientation of the nodes in the graph using the well-known *Pointwise Mutual Information-Information Retrieval* (*PMI-IR*) algorithm (Turney, 2002; Turney & Littman, 2003). This algorithm estimates the semantic orientation of a term based on the number of webpages (*hits*) that contain that term in the near context of a positive seed (*excellent*) and a negative seed (*poor*):

$$SO_{PMI-IR}(t) = log_2\left(\frac{hits(t\ NEAR\ excellent)hits(poor)}{hits(t\ NEAR\ poor)hits(excellent)}\right) \tag{6}$$

The method rely in queries to a web search engine (AltaVista). The operator *NEAR* constrains the search to documents that contain the words within 10 words of one another, in either order.

4.1.2. Results

The results are shown in Table 1. The Kendall distance obtained between the gold-standard ranking and the one obtained by the application of PolarityRank to the graph is 0.236. This is a better result than the one obtained by applying PageRank to the graph with no negative edges (0.242) and than the one obtained by applying the *Pointwise Mutual Information-Information Retrieval* algorithm (0.255).

These results support the better adaptation of PolarityRank to the problem, as it uses a portion of the information appeared in the corpus (related to the inverse conjunctive relations) that you cannot directly deal with when using PageRank. The number of negative edges in the graph is only 5% of all. Still, PolarityRank achieves a reduction over PageRank of 2.5% in $\tau_{\frac{1}{2}}$ (see first row of Table 2). To measure the benefit obtained by PolarityRank in a situation where positive and negative edges are more balanced, we conducted a second experiment whose results are shown in the second row of Table 2. In this experiment we sampled randomly the direct relations 100 times, keeping 15,467 in each case so that the resulting graph would have the same number of positive and negative arcs. The values shown in the table are the means of the results obtained for each ranking in each one of the samples. In this case, PolarityRank obtained a much more significant improvement with respect to PageRank (a reduction of 9.54% in $\tau_{\frac{1}{2}}$). This result suggests that using PolarityRank will imply further improvements in those graphs with a more balanced number of positive and negative edges.

Table 1
Results for the estimation of semantic orientation of words, using three methods: *Pointwise Mutual Information-Information Retrieval* (*PMI-IR*), *PageRank* and *PolarityRank*.

	Kendall distance $\left(\tau_{\frac{1}{2}}\right)$
PMI-IR	0.255
PageRank	0.242
PolarityRank	**0.236**

Table 2
Results for the experiments on the estimation of semantic orientation of words: reductions in Kendall distances $\delta(\tau_{\frac{1}{2}})$ achieved by PolarityRank over PageRank in unbalanced and balanced situations.

	Positive edges	*Negative edges*	*Kendall distance* $(\tau_{\frac{1}{2}})$		
			PageRank	PolarityRank	$\delta\left(\tau_{\frac{1}{2}}\right)$ (%)
Complete graph	277,148	15,467	0.242	**0.236**	−2.50
Balanced graph	15,467	15,467	0.409	**0.370**	−9.54

```
func WITS($T$: instance set; $PO$: processing order)
    dev $S$: set of weighted instances
algorithm
    create an empty queue of instances $q_c$ for each class of the instance set $T$
    for each instance $ins$ in $T$:
        asign $ins$ to its class queue, keeping the $PO$ order
    end for
    $S := \emptyset$
    for each queue $q_c$:
        $numErrorsQueue_c$ := number of elements in $q_c$
    end for
    $numErrors := |T|$
    while there exist instances in any queue $q_c$ and $numErrors > 0$:
        $cand$ := next instance in the queue $q_c$ with highest $numErrorsQueue_c$
        remove $cand$ from $q_c$
        $weightCand$ := weight that minimizes the number of errors
        $numErrorsPost$ := error obtained by $cand \cup S$ evaluated over $T$
        if $numErrors - numErrorsPost \geq G$:
            assign weight $weightCand$ to $cand$
            $S := cand \cup S$
            $numErrors := numErrorsPost$
            for each queue $q_c$:
                recalculate $numErrorsQueue_c$
            end for
        end if
    end while
end algorithm
```

Fig. 2. WITS algorithm Pseudocode.

4.2. Instance selection

One of the most interesting and useful tasks in data mining is the classification of data using the information available in a data set previously tagged with a certain class. One of the most popular methods is the nearest neighbor technique (Wilson, 1972): the new instance is compared by a particular metric with all other instances, searching the one whose distance to the new one is the smallest possible. Finally, the class of the nearest instance is assigned to the new instance assuming it has more in common with it than with any other. The nearest neighbor technique (or its variant, the k-nearest neighbors) has proven to be one of the methods that have shown better results for the task of classification.

A drawback of this technique is that every time a new instance has to be classified, it has to be compared with all the instances in the set, with the consequent cost in terms of processing time and storage requirements. One way to mitigate this cost is using instance selection (or reduction) techniques: instead of using all instances of the original set, only some of them are selected, either because they are representatives of all the elements of the set or because they properly delineate the boundaries of each class (Aha, Kibler, & Albert, 1991; Wilson & Martinez, 2000).

A very interesting approach that obtains good results is WITS (Morring & Martinez, 2004) which is an acronym for *Weighted Instance Typicality Search Algorithm*. The algorithm takes a collection of instances and an a priori processing order as input, and produces as output a subset of representative instances and a weight assigned to each of them. The algorithm is sensitive to the order in which the instances are processed, and it is in fact at that point where we evaluate the contribution of the ranking produced by PolarityRank. In Fig. 2 the pseudocode of the WITS algorithm is included.

Table 3
Description of the UCI data sets used. For each data set the abbreviation used in the next table, the number of instances, the number of attributes and the percentage of instances belonging to the mayority class are shown.

Data set	# Instances	# Attributes	% Majority class instances
Australian Credit Approval (ACA)	690	15	55.51
Breast Cancer Wisconsin (BCW)	699	10	65.52
Echocardiogram (ECH)	132	12	67.57
Hepatitis (HEP)	155	19	79.35
Horse Colic (HOR)	368	27	63.04
Ionosphere (ION)	351	34	64.10
Liver Disorders (LIV)	345	6	57.97
Pima Indians Diabetes (PIM)	768	8	65.10
Molecular Biology (Promoters) (PRO)	106	58	50.00
Voting Records (VOT)	435	16	61.40

In each iteration, WITS determines the quality of the set of instances selected so far, it evaluates the number of hits achieved by classifying the complete set of instances using that subset. In order to do this, it applies the nearest neighbor technique using a *weighted* distance based on the HVDM metric (Wilson & Martinez, 1997). HVDM means *Heterogeneous Value Difference Metric* and combines Euclidean distance for continuous attributes and *Value Difference Metric* (VDM) for the discrete.

4.2.1. Experimental setup

The experiments were carried out with 10 data sets taken from the well known UCI repository (Asuncion & Newman, 2007) and whose characteristics are given in Table 3. Only data sets with two classes have been taken, so that one of them has been considered the positive type and the other the negative. Ten-fold cross-validation has been used for the evaluation.

In our experiments we have calculated instance rankings that have subsequently been used to determine the processing order required by the WITS algorithm. We first need to calculate a graph from the original set of instances. We consider the instances of the data set are the vertices of the graph and the edges are labeled with a measure of similarity between the vertices connected. In order to do this we need to have a measure of distance between any pair of vertices of the graph, we use the metric HOEM (Wilson & Martinez, 1997): the Euclidean distance for continuous attributes and the overlap distance for the discrete ones (0 if they have the same value, 1 if they have different values). This distance has been normalized so that

$$\forall x, y \in T \quad 0 \leqslant \text{dist}(x, y) \leqslant 1.$$

To evaluate the similarity between two instances the function

$$\text{sim}(x, y) = \frac{1}{1 + k\ \text{dist}(x, y)}$$

has been used, which given the distance between two instances returns the similarity between them. k is a parameter that needs to be adjusted adequately.

The calculation of PolarityRank takes as input a graph with positive and negative edges. The edges between instances of the same class have been considered as positive votes and those that connect instances of different classes as negative votes. The weights p_{ij} of the edges are calculated in the following manner:

- Let us $In^{+}(v_i)$ be the set of indexes of the instances that have the same class as v_i. If $j \in In^{+}(v_i)$, p_{ji} will be the similarity value between v_j and v_i, sim (j,i).
- Let us $In^{-}(v_i)$ be the set of indexes of the instances that have a different class than v_i. If $j \in In^{-}(v_i)$, p_{ji} will be the similarity value between v_j and v_i with the opposite sign, $-$sim (j,i).

As positive and negative seeds there have been taken the higher ranked instances returned by the PageRank ranking algorithm applied to two independent graphs, one for the positive class and another for the negative one. The number of instances used as seeds has been a percentage of the total set of instances of the dataset; experimentally it has been concluded that the optimum percentage was 0.89%. The value assigned to each seed has been m/ns, where m is the total number of items and ns the number of seeds.

Once PolarityRank is calculated, we use the value of the normalized difference of PR^{+} and PR^{-} (formula (4)) to obtain the processing order that WITS algorithm needs. In the case of the processing order produced by PageRank, we also apply the same procedure that we used in the semantic orientation experiments.

4.2.2. Results

The results obtained are shown in Table 4. It reflects both the precision and the size of the selected instances set in each experiment. The size was measured as the ratio, in percentage, between the size of the selected instances set and the size of

Table 4
Comparison between the precision and size of the sets of instances selected by 1NN, WITS using as ranking the one calculated by PageRank (*PR*) and WITS using the ranking returned by PolarityRank ($PR^{\pm}$).

Data set	1NN		DROP3		*PR*		$PR^{\pm}$	
	% Prec.	% Size	% Prec.	% Size	% Prec.	% Size	% Prec.	% Size
ACA	82.03	100.00	83.91	5.96	84.35	**0.98**	**84.49**	1.14
BCW	95.28	100.00	96.14	3.58	96.14	**0.48**	**96.28**	0.49
ECH	89.19	100.00	**93.39**	10.66	90.54	***3.60***	90.54	***3.60***
HEP	80.65	100.00	81.87	7.81	81.94	8.46	**86.45**	**6.67**
HOR	67.44	100.00	70.13	10.30	**80.16**	**3.80**	79.35	4.11
ION	86.32	100.00	87.75	7.06	91.17	**3.99**	**92.88**	4.81
LIV	***62.90***	100.00	60.84	24.99	***62.90***	**2.96**	60.29	3.93
PIM	70.18	100.00	75.01	16.90	**75.78**	**1.75**	75.13	1.90
PRO	83.96	100.00	**86.82**	16.67	70.75	**13.63**	84.91	15.83
VOT	92.41	100.00	**95.87**	5.11	94.25	3.35	94.48	**3.01**
Aver.	81.04	100.00	83.17	10.90	82.80	**4.30**	**84.48**	4.55

the training set, so a smaller value indicates a greater reduction capacity. Precision was measured as the number of hits on the total of classified instances, so that a higher value indicates a better precision.

We have compared the results obtained by the WITS algorithm using as processing order both PageRank (*PR*) and PolarityRank ($PR^{\pm}$) with those obtained by the nearest neighbor algorithm (1NN) and with the DROP3 algorithm (Wilson & Martinez, 2000), which is the most popular and cited among the instance selection algorithms. In the first two columns the results of the nearest neighbor technique are shown, for which the size is obviously a 100% as all the instances in the data set are used in classification phase. The rest of the columns present the results obtained when instances are selected by DROP3 and by the WITS algorithm using as processing order, respectively, the rankings produced by PageRank (*PR*) and PolarityRank ($PR^{\pm}$).

In the table, the technique that obtains the better precision and the best reduction is highlighted in bold. When the best precision is obtained by the nearest neighbor technique, the one that, among the rest, gets the best precision is written in italics. In case of a tie between several winning techniques, a combination of bold with italics has been used.

We can see that the best average precision is obtained by PolarityRank ($PR^{\pm}$), improving the nearest neighbor technique in nine data sets and losing just in one occasion. This means that PolarityRank brings a wealth of editing capabilities: among the instances that it selects, it does not include those that misclassify.

$PR^{\pm}$ improves the precision of DROP3 in 6 datasets, losing in four and obtaining the best average. Regarding reduction $PR^{\pm}$ improves DROP3 in all the datasets. When PolarityRank is used as processing order in WITS it also improves PageRank in six occasions, tying in one and losing in the remaining three. Regarding reduction, *PR* exceeds $PR^{\pm}$ seven times, tying one and losing two. However, with a very small loss in reduction (only a difference of 0.25%) $PR^{\pm}$ achieves a substantially better precision with regard to *PR* (an improvement of 1.68%), being this measure also much more important than the reduction.

We can conclude that the use of PolarityRank allows to select very good elements from the set of source data; remember that in order to do this the algorithm considers first the elements with higher PolarityRank, which means that this measure is able to identify very well the most characteristic elements of each data set.

5. Conclusions

In this paper we have presented a ranking algorithm, on graphs with weights, which can handle both positive and negative influences. Ranking or relevance calculation algorithms make possible to extract very valuable information from networks and graphs. Those that take into account the weights of the edges usually are limited to the analysis of supportive (positive) relationships between the nodes of the network. On the basis of PageRank, the best known relevance algorithm without any doubt, we defined a new algorithm called PolarityRank that allows taking into account the penalties (or negative votes) between the nodes of a network.

Our work includes the formal definition of PolarityRank, its algebraic justification and a demonstration of its convergence. We have applied this idea in the resolution of two unrelated tasks in which one can define positive and negative relationships between nodes of a graph: the calculation of the semantic orientation of words and the selection of instances in a learning database. In both cases we have conducted an experimental study and results show that the use of this new ranking provides an information that cannot be captured by the algorithms of the PageRank style.

PolarityRank can be applied to any knowledge propagation problem: whenever a small amount of information on some of the entities involved (and about the similarities and differences between the entities) is available, a graph can be built with positive and negative edges representing the similarities and differences between them. For example, we are planning to apply PolarityRank to web spam detection and to computation of trust and reputation in social networks.

References

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning, 6*, 37–66.

Asuncion, A. & Newman, D. (2007). UCI machine learning repository. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Cerini, S., Compagnoni, V., Demontis, A., Formentelli, M., & Gandini, G. (2007). Language resources and linguistic theory: Typology, second language acquisition, English linguistics. Franco Angeli Editore, Milano, IT, Ch. Micro-WNOp: A gold standard for the evaluation of automatically compiled lexical resources for opinion mining. <http://www.unipv.it/wnop/#CITECerini07>.

Esuli, A. & Sebastiani, F. (2007). Pageranking wordnet synsets: An application to opinion mining. In *Proceedings of ACL-07, the 45th annual meeting of the association of computational linguistics* (pp. 424–431). Association for Computational Linguistics.

Fagin, R., Kumar, R., Mahdian, M., Sivakumar, D., & Vee, E. (2004). Comparing and aggregating rankings with ties. In *PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems* (pp. 47–58). New York, NY, USA: ACM.

Fellbaum, C. (Ed.). (1998). *WordNet: An electronic lexical database*. MIT Press.

Gyöngyi, Z., Garcia-Molina, H. & Pedersen, J. (2004). Combating web spam with trustrank. In *VLDB '04: Proceedings of the thirtieth international conference on very large data bases* (pp. 576–587). VLDB Endowment.

Hatzivassiloglou, V. & McKeown, K. R. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the association for computational linguistics* (pp. 174–181). Association for Computational Linguistics, Morristown, NJ, USA.

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 168–177). New York, NY, USA: ACM.

Kamps, J., Marx, M., Mokken, R.J., De Rijke, M. (2004). Using wordnet to measure semantic orientation of adjectives. In *National Institute for* (Vol. 26, pp. 1115–1118).

Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM, 46*, 604–632.

Leskovec, J., Huttenlocher, D., & Kleinberg, J. (2010). Predicting positive and negative links in online social networks. In *WWW '10: Proceedings of the 19th international conference on World wide web* (pp. 641–650). New York, NY, USA: ACM.

Mihalcea, R. & Tarau, P. (2004). Textrank. bringing order into texts. In *Proceedings of the conference on empirical methods in natural language processing*. Barcelona, Spain.

Mihalcea, R., Tarau, P., & Figa, E. (2004). Pagerank on semantic networks with application to word sense disambiguation. In *Proceedings of the 20th international conference on computational linguistics*. Geneva, Switzerland.

Morring, B. D., & Martinez, T. R. (2004). Weighted instance typicality search (WITS): A nearest neighbor data reduction algorithm. *Intelligent Data Analysis, 8*(1), 61–78.

Nie, Z., Zhang, Y., Wen, J.-R., & Ma, W.-Y. (2005). Object-level ranking: Bringing order to web objects. In *WWW '05: Proceedings of the 14th international conference on World Wide Web* (pp. 567–574). New York, NY, USA: ACM.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). *The pagerank citation ranking: Bringing order to the web*. Tech. rep., Stanford Digital Library Technologies Project.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval, 2*(1-2), 1–135.

Turney, P. D. (2002). Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting of the association for computational linguistics* (pp. 417–424).

Turney, P. D., & Littman, M. L. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems, 21*, 315–346.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics, 2*(3), 408–421.

Wilson, D. R., & Martinez, T. R. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research, 6*, 1–34.

Wilson, D. R., & Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning, 38*, 257–286.