NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES
“COMPUTER SCIENCE”

MASTER THESIS

Topic-Aware Influence Maximization Framework

Xenofon V. Kitsios

SUPERVISOR: Alexis Delis, Professor

ATHENS

NOVEMBER 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
“ΠΛΗΡΟΦΟΡΙΚΗ”

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Πλαίσιο Μεγιστοποίησης Επιρροής με Επιγνωσή Θέματος

Ξενοφών Β. Κίτσιος

ΕΠΙΒΛΕΠΩΝ: Αλέξης Δελής, Καθηγητής

ΑΘΗΝΑ

ΝΟΕΜΒΡΙΟΣ 2021

MASTER THESIS

Topic-Aware Influence Maximization Framework

Xenofon V. Kitsios
R.N.: CS2.19.0022

November 2021

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Πλαίσιο Μεγιστοποίησης Επιρροής με Επιγνωσή Θέματος

Ξενοφών Β. Κίτσιος
Α.Μ.: CS2.19.0022

Νοέμβριος 2021

ABSTRACT

The focus of the thesis is given on the development of a novel framework to deal with the topic-aware influence maximization problem. The problem appears high computational complexity and belong in the area of social influence which involves both psychological and sociological aspects. In particular, the influence maximization problem aims to find a subset of nodes that maximize the expected spread of influence in a network.

Due to NP-hard computational complexity, the approximation algorithms are imperative to gain optimal or near-optimal solution with minor computational effort. In addition, real life networks should be used to provide valid results and significant information about how social influence is spread avoiding errors and problematic conclusions.

In the context of this thesis, the topic-aware influence maximization problem is studied and a novel framework based on the Hyperlink Induced Topic Search (HITS) algorithm is introduced. The proposed solution approach consists of two main components, i.e., the HITS algorithm and the Greedy or Cost Effective Lazy Forward (CELF) algorithms with modified Independent Cascade model in order to improve the results of the influence according to a topic. The HITS algorithm aims to analyze links and the Greedy and CELF algorithms to find the nodes that maximize the expected spread of influence in a network.

For the evaluation, the Neo4j graph database is used as a platform for the empirical experiments. Moreover, datasets of the Yelp social network alongside with generated graphs based on Barabási–Albert model provide the necessary data for testing. The computational results illustrate the pertinence of the developed algorithms and underline the role of the proposed framework's components. Finally, it is worth to mention that the developed Greedy and CELF algorithms of this thesis have been contributed to the Neo4j open-source community.

SUBJECT AREA: Social Network Analysis

KEYWORDS: Influence Maximization, Topic-Aware, Greedy, CELF, Independent Cascade model, HITS, Neo4j, Barabási–Albert model, Yelp dataset

ΠΕΡΙΛΗΨΗ

Η διπλωματική εργασία εστιάζει στην ανάπτυξη ενός νέου πλαισίου για την αντιμετώπιση του προβλήματος της μεγιστοποίησης με επίγνωση θέματος. Το πρόβλημα παρουσιάζει υψηλή υπολογιστική πολυπλοκότητα και ανήκει στην περιοχή της κοινωνικής επιρροής η οποία περιλαμβάνει τόσο ψυχολογικές όσο και κοινωνιολογικές πτυχές. Συγκεκριμένα, το πρόβλημα μεγιστοποίησης επιρροής στοχεύει στην εύρεση ενός υποσυνόλου από κόμβους που μεγιστοποιούν την αναμενόμενη διάδοση επιρροής σε ένα δίκτυο.

Λόγω της NP-hard υπολογιστικής πολυπλοκότητας, οι προσεγγιστικοί αλγόριθμοι είναι επιτακτικοί για την εύρεση της βέλτιστης ή σχεδόν βέλτιστης λύσης με μικρή υπολογιστική προσπάθεια. Επιπλέον, θα πρέπει να χρησιμοποιούνται πραγματικά δίκτυα για να παρέχουν έγκυρα αποτελέσματα και σημαντικές πληροφορίες σχετικά με το τρόπο διάδοσης της κοινωνικής επιρροής αποφεύγοντας έτσι λάθη και προβληματικά συμπεράσματα.

Στο πλαίσιο αυτής της διπλωματικής εργασίας, το πρόβλημα της μεγιστοποίησης επιρροής με επίγνωση θέματος μελετήθηκε και ένα νέο πλαίσιο βασισμένο στον αλγόριθμο Hyperlink Induced Topic Search (HITS) εισήχθη. Η προτεινόμενη προσεγγιστική λύση αποτελείται από δύο μέρη, δηλαδή τον αλγόριθμο HITS και τους αλγορίθμους Greedy ή Cost Effective Lazy Forward (CELF) με τροποποιημένο το μοντέλο Independent Cascade ώστε να βελτιωθούν τα αποτελέσματα της επιρροής με επίγνωση θέματος. Ο αλγόριθμος HITS στοχεύσει να αναλύει συνδέσεις και οι αλγόριθμοι Greedy και CELF να βρίσκουν τους κόμβους που μεγιστοποιούν την αναμενόμενη διάδοση επιρροής σε ένα δίκτυο.

Για την αξιολόγηση, η Neo4j βάση δεδομένων γράφου χρησιμοποιήθηκε σαν πλατφόρμα για τα εμπειρικά πειράματα. Επιπλέον, σύνολα δεδομένων από το κοινωνικό δίκτυο Yelp μαζί με δημιουργούμενους γράφους βασισμένους στο μοντέλο Barabási–Albert παρέχουν τα απαραίτητα δεδομένα για δοκιμή. Τα υπολογιστικά πειράματα απεικονίζουν την συνάφεια των αναπτυγμένων αλγορίθμων και υπογραμμίζουν το ρόλο των μερών του προτεινόμενου πλαισίου. Τέλος, αξίζει να σημειωθεί ότι οι αναπτυγμένοι αλγόριθμοι Greedy και CELF αυτής της διπλωματικής εργασίας έχουν γίνει συνεισφορά στη Neo4j κοινότητα ανοιχτού κώδικα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ανάλυση Κοινωνικών Δικτύων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μεγιστοποίηση Επιρροής, Επιγνωσή Θέματος, Greedy, CELF, Μοντέλο Independent Cascade, HITS, Neo4j, Μοντέλο Barabási-Albert, Σύνολο Δεδομένων Yelp

ACKNOWLEDGEMENTS

Firstly, I would like to express my appreciation to Prof. Alexis Delis. Furthermore, I would like to thank the committee examiner Prof. Mema Rousopoulou.

The completion of this study could not have been possible without the expertise and support of Dr. Panagiotis Liakos.

Finally, I would like to thank my family for their support. Especially my cousin Afroditi Anagnostopoulou.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

PREFACE

This thesis entitled as “Topic-Aware Influence Maximization Framework” has been prepared by Xenofon Kitsios.

The thesis is submitted as partial fulfilment of the requirement for the MSc degree “Computer Science” at Department of Informatics and Telecommunications at the National and Kapodistrian University of Athens.

The subject of this thesis has been inspired from the “Social Network Analysis” course taught by Dr. Panagiotis Liakos.

1. INTRODUCTION

Social influence involves the ways and efforts in order to change another person's beliefs, attitudes, or behavior. The main characteristic of the social influence process is the high level of complexity since it involves both psychological and sociological aspects. However, the target group may not understand the influence attempt and many times the outcomes are not consistent with the goals of the communicator. Research on social influence covers multidisciplinary fields and explores it on the basis of different questions such as "viral marketing", "disease modelling", etc.

In this context, social influence is modelled upon networks and is based on sociological assumptions about how people become influenced. Even though this is an efficient approach to present interactions among individuals, it is not capable to provide information about how influence is spread. Towards this direction, the influence maximization problem aims to find a subset of nodes such that the resulting propagation from the subset reaches the largest number of nodes in the network.

The influence maximization problem is an NP-hard and several approximation algorithms have been proposed to deal with it. However, they focus on a non-real representation without considering a dynamic environment which responds to the realistic conditions. Most of the times, this simplification result to errors and problematic conclusions that do not capture the effect of social influence in practice. For this reason, real life networks should be used in an attempt to provide valid results and significant information about how social influence is spread.

Limited studies conducted to study the topic-aware modification, in which the network is modelled as a graph where the edge of any two users is associated with a topic distribution. The focus of this thesis is given on the development of a novel framework to deal with the topic-aware influence maximization problem.

The framework succeeds to turn classical topic-blind influence maximization algorithms into topic-aware by using an established link analysis algorithm. In addition, algorithms developed in Neo4j (i.e., a modern graph database) following the proposed novel framework.

The remainder of this thesis is arranged as follows:

Chapter 2 describes the research background of the influence maximization problem and a literature review of the most significant studies.

Chapter 3 introduces the Hyperlink Induced Topic Search (HITS) algorithm, which used for analyzing links and rating nodes by comparing edges pointing in and out of them in a graph.

Chapter 4 addresses the influence maximization problem and the proposed novel framework is presented and its components are thoroughly explained.

Chapter 5 presents in detail the conducted experiments and analyses the derived results.

Chapter 6 provides an overall synopsis of conclusions and directions of future research.

2. INFLUENCE MAXIMIZATION PROBLEM

2.1 Definition

The influence maximization problem aims to find a subset of nodes in a network such that the resulting influence propagating from that subset reaches the largest number of nodes in the network. The influence could represent information, behavior, disease, product adoption, traffic etc., that can pass across edges of connected peers within a network.

Definition 2.1.1 (Influence maximization problem)**.** Given a network $G(V, E)$ with V vertices (also called nodes), E edges, and a number k, where $k < V$, find a set S of k nodes that the expected spread of influence is maximized.

The influence maximization problem is a **NP**-hard problem and for this reason significant computational effort is required. For example, a network of 1.000 nodes has $\begin{pmatrix} n \\ k \end{pmatrix} \approx 8 \times 10^{12}$ different possible candidates of size $k = 5$ seed set, which is impossible to be solved exactly even using state-of-the-art computing resources of high performance.

2.2 Literature Review

The first algorithmic treatment of the problem is provided by Domingos and Richardson [1] [2], who modelled the diffusion process in term of Markov random fields and proposed heuristic solutions to the problem.

Kempe et al. [3] study the influence maximization problem as a discrete optimization problem focusing on two propagation models, the “Independent Cascade”, and the “Linear Threshold”. In both models, at a given timestamp, each node is either active or inactive. Each node’s tendency to become active increases monotonically as more of its neighbor becomes active. An active node never becomes inactive again.

In the Independent Cascade model when a node v first becomes active, say at time t, it is considered contagious. It has one chance of influence each inactive neighbor u with probability $p_{(v,u)}$, independently of the history so far. If the tentative succeeds, u becomes active at time $t + 1$.

Independent Cascade model is presented in Algorithm 2.1.

Algorithm 2.1 Independent Cascade model

```
1:  procedure $IndependentCascade(G(V, E), \Phi(t_0), p)$
2:      $i \leftarrow 0$                                        ▷ $i$ represents the current round
3:      while $\Phi(t_i) \neq \emptyset$ do                     ▷ As long as a new node activates
4:          $i \leftarrow i + 1$
5:          $\Phi(t_i) \leftarrow \emptyset$
6:          for all $v \in \Phi(t_{i-1})$ do                    ▷ Nodes activated in previous round
7:              for all $u \in N_v^V$ do                        ▷ Neighbors of activated node
8:                  if $u \notin \cup_{j=0}^{i} \Phi(t_j)$ then     ▷ Neighbor must be inactive
9:                      if $rand(0, 1) < p_{(v,u)}$ then        ▷ A chance to get active
10:                         $\Phi(t_i) \leftarrow \Phi(t_i) \cup \{u\}$   ▷ Neighbor becomes active
11:                     end if
12:                 end if
13:             end for
14:         end for
15:     end while
16:     return $\cup_{j=0}^{i} \Phi(t_j)$                       ▷ Return the activated nodes of all rounds
17: end procedure
```

In the Linear Threshold model, each node u is influenced by each neighbor v according to a weight $b_{(v,u)}$, so that the sum of incoming weights to u is less than or equal to 1. Each node u chooses a threshold θ_u uniformly at random in the $[0, 1]$ range. At any timestamp t, if the total weight from the active neighbors of an inactive node u is at least θ_u, then u becomes active at timestamp $t + 1$.

Linear Threshold model is presented in Algorithm 2.2.

Algorithm 2.2 Linear Threshold model

```
 1: procedure $LinearThreshold(G(V, E), \Phi(t_0), b)$
 2:     for all $v \in V$ do
 3:         $\theta_v \leftarrow rand(0, 1)$                              ▷ Set random thresholds
 4:     end for
 5:     $i \leftarrow 0$                                                    ▷ $i$ represents the current round
 6:     while $i = 0$ or $\Phi(t_{i-1}) \neq \Phi(t_i)$ do                ▷ As long as a new node activates
 7:         $\Phi(t_{i+1}) \leftarrow \Phi(t_i)$
 8:         for all $v \in V \backslash \Phi(t_i)$ do                      ▷ Inactive nodes
 9:             if $\sum_{u \in N_v^{\Phi(t_i)}} b_{(v,u)} \geq \theta_v$ then     ▷ A chance to get active
10:                 $\Phi(t_{i+1}) \leftarrow \Phi(t_{i+1}) \cup \{u\}$         ▷ Node becomes active
11:             end if
12:         end for
13:         $i \leftarrow i + 1$
14:     end while
15:     return $\Phi(t_i)$                                                 ▷ Return the activated nodes
16: end procedure
```

Both Independent Cascade and Linear Threshold models are repeated until there is not any new node to become active. In the Independent Cascade model, the influencer becomes more important, since s/he holds a probability. This is one of the major differences between the two models since that in Linear Threshold model the influence parameter is assigned to the nodes randomly.

The algorithm introduced by Kempe et al. [3] is known as “Greedy”. Greedy algorithm successively selects the node that maximize the marginal gain and provides an approximation of the optimum solution within a factor of $(1-1/e)$. This is due to the nice properties of monotonicity and submodularity that the spread function exhibits under these models.

Given a propagation model m and a seed set S, where $S \subseteq V$, the expected number of active nodes at the end of process is denoted by $\sigma_m(S)$. The influence maximization problem requires to find the set $S \subseteq V$, $|S| = k$, such that $\sigma_m(S)$ is maximum.

- Monotonicity indicates that as more neighbors of some arbitrary node gets active, the probability of u to become active increases (i.e. $\sigma_m(S) \leq \sigma_m(T)$ whenever $S \subseteq T$).
- Submodularity indicates that the marginal gain of a new node shrinks as the set grows (i.e. $\sigma_m(S \cup \{u\}) - \sigma_m(S) \geq \sigma_m(T \cup \{u\}) - \sigma_m(T)$ whenever $S \subseteq T$).

In their paper [3], Kempe et al. run Monte-Carlo simulations to obtain an accurate estimation of the expected spread. In particularly, they show that for any $\varepsilon > 0$, there is a $\gamma > 0$ such that by using $(1 + \gamma)$-approximate values of expected spread a $(1 - 1/e - \varepsilon)$-approximation for the influence maximization problem is obtained.

There are two major limitations in Greedy algorithm:

1. The algorithm requires repeated computations of the spread function for various set of seed. The problem of computing the spread under Independent Cascade and Linear Threshold models is **#P**-hard. As a result, Monte-Carlo simulations are run for sufficiently times to obtain an accurate estimation which result in long computation time.
2. In each iteration, the algorithm searches all the nodes in the graph as a potential candidate for the next seed node. As a result, this algorithm entails a quadratic number of steps in terms of the number of nodes.

Greedy algorithm is presented in Algorithm 2.3.

Algorithm 2.3 Greedy algorithm

```
1: procedure $Greedy(G(V,E), k, w, simulations)$
2:     $S \leftarrow \emptyset$
3:     for $i = 1$ to $k$ do
4:         for all $u \in V \backslash S$ do          ▷ Calculate the spread for all not selected nodes
5:             $s_u \leftarrow 0$
6:             for $j = 1$ to $simulations$ do
7:                 $s_u \leftarrow s_u + |Inf(S \cup \{u\})|$
8:             end for
9:             $s_u \leftarrow s_u / simulations$
10:        end for
11:        $S \leftarrow S \cup \{arg\,max_{u \in V \backslash S}\{s_u\}\}$          ▷ Select the node that maximize the spread
12:    end for
13:    return $S$
14: end procedure
```

Leskovec et al. [4] introduced an efficient algorithm called Cost-Effective Lazy Forward (CELF). In CELF algorithm the submodularity is exploited based on a “Lazy Forward” optimization approach by selecting new nodes.

The idea is that the marginal gain of a node in the current iteration cannot be better than its marginal gain in previous iterations. CELF algorithm maintains a table $\langle u, \Delta_u(S)\rangle$ sorted on $\Delta_u(S)$ decreasing order, where S is the current seed and $\Delta_u(S)$ is re-evaluated only for the top node at a time and if it is needed, the table is resorted. If a node remains at the top, it is picked in the next seed.

Empirical experiments show that the CELF algorithm dramatically improves the efficiency, being 700 times faster than the Greedy algorithm. It is worth to mention that the Greedy algorithm proposes $(1 - 1/e - \varepsilon)$ optimal solutions and CELF algorithm proposes $1/2(1 - 1/e)$ optimal solutions.

CELF algorithm is presented in Algorithm 2.4.

Algorithm 2.4 CELF algorithm

```
1: procedure CELF($G(V, E)$, $k$, $w$, $simulations$)
2:     $S \leftarrow \emptyset$
3:     for $i = 1$ to $k$ do
4:         for all $u \in V \backslash S$ do          ▷ Calculate the spread for all not selected nodes
5:             if $s_u^{r-1} < s_{u'}^r$ then          ▷ If top node spread is higher than others, continue
6:                 $continue$
7:             end if
8:             $s_u \leftarrow 0$
9:             for $j = 1$ to $simulations$ do
10:                $s_u \leftarrow s_u + |Inf(S \cup \{u\})|$
11:            end for
12:            $s_u \leftarrow s_u / simulations$
13:        end for
14:        $S \leftarrow S \cup \{arg\,max_{u \in V \backslash S}\{s_u\}\}$          ▷ Select the node that maximize the spread
15:    end for
16:    return $S$
17: end procedure
```

Goyal et al. [5] propose the “CELF++” algorithm which constitutes a highly optimized approach based on the CELF algorithm to further improve the naive Greedy algorithm. CELF++ algorithm exploits the property of submodularity of the spread function for influence propagation models to avoid unnecessary recomputations of gains incurred by CELF algorithm. Empirical experiments show that CELF++ algorithm works effectively and efficiently, resulting in significant improvements in terms of both running time and the average number of node look-ups.

Several techniques [4, 5] are proposed to optimize the Greedy algorithm [3]. Moreover, researchers have started to search new ways of maximizing the spread of influence [6, 7, 8, 9]. Some of these new ways focus on topic-aware influence maximization problem [7, 9].

Saito et al. [6] study how to learn the probabilities for the Independent Cascade model from a set of past propagations. They formally define the likelihood maximization problem and then apply the Expectation Maximization algorithm to solve it. Even though the proposed formulation is elegant, it is not scalable to voluminous datasets for the reason that in each iteration the Expectation Maximization algorithm must update the influence probability associated to each edge.

Tang et al. [7] are among the first that studied the problem of influence maximization based on a topic. Given a network and a topic distribution for each node, the aim to find topic specific subnetworks, and topic specific influence weights among subnetworks. They introduce the “Topical Affinity Propagation” approach using a graphical probabilistic model. Moreover, Tang et al. deal with the efficiency by devising a distributed learning algorithm under the MapReduce model. The focus of their work is the expert finding problem without proposing any propagation model or studying the influence maximization problem.

Most of the works assume a weighted graph as input and do not address how the edge probabilities (or influence weights) can be obtained. Goyal et al. [8] study this problem an instance of General Threshold model. They extend this model by introducing temporal decay, as well as factors such as the influenceability of a specific user and influence-proneness of a certain action. The interesting fact about their work is that they manage to predict whether a user will perform an action and when.

Barbieri et al. [9] extent the Independent Cascade model to be topic aware. The model of this work named “Topic-aware Independent Cascade”. The relationship strength between two nodes is computed by their topic preference learned from history activities on a social network. In their experiments show that topic-aware influence propagation models are more accurate in describing real world influence driven propagations than the state-of-the-art topic-blind models and considering the characteristics of the item, many adoptions can be obtained in the influence maximization problem.

This thesis utilizes the Greedy and CELF algorithms as well as the Independent Cascade model in order to develop a novel framework that deals with the topic-aware influence maximization problem.

3. HYPERLINK-INDUCED TOPIC SEARCH (HITS) ALGORITHM

3.1 Definition

Kleinberg [10] developed the Hyperlink-Induced Topic Search (HITS) algorithm for analyzing the links in a network. The HITS algorithm has been successfully applied in rating websites and scientific journals.

HITS algorithm defines a $hubScore$ and an $authorityScore$ for each node. The $authorityScore$ estimates the value of the node and the $hubScore$ estimates the value of its links to other nodes.

The idea stemmed from a particular insight into the creation of web pages when the internet was originally forming. Certain web pages, known as hubs, served as large directories that were not actually authoritative in the information that they held, but were used as compilations of a broad catalog of information that led users direct to other authoritative pages.

The steps of the algorithm are:

1. Set for each node the $hubScore = 1$ and the $authorityScore = 1$.
2. Choose a number of k steps.
3. In each step, perform a sequence of updates.
   Each update works as follows:
   - First apply the Authority Update Rule.
   - Then apply the Hub Update Rule.
4. Normalize $hubScore$ and $authorityScore$ to their relative sizes; divide down each $authorityScore$ by the sum of all $authorityScore$ and divide down each $hubScore$ by the sum of all $hubScore$.

Definition 3.1.1 (Authority update rule)**.** For each node, update $authorityScore$ to be the sum of the $hubScore$ of all nodes that points to the node.

Definition 3.1.2 (Hub update rule)**.** For each node, update $hubScore$ to be the sum of the $authorityScore$ of all nodes that the node points to.

3.2 Example

In the following example [11], the results of the query “newspapers” that are retrieved from the web are examined. Figure 3.1 shows the results. Query returns websites relevant to newspapers, like USA Today, New York Time, Wall St. Journal, SJ Merc News, and popular websites, like Amazon, Yahoo!, Facebook. The unnamed nodes represent the sample of websites relevant to query.

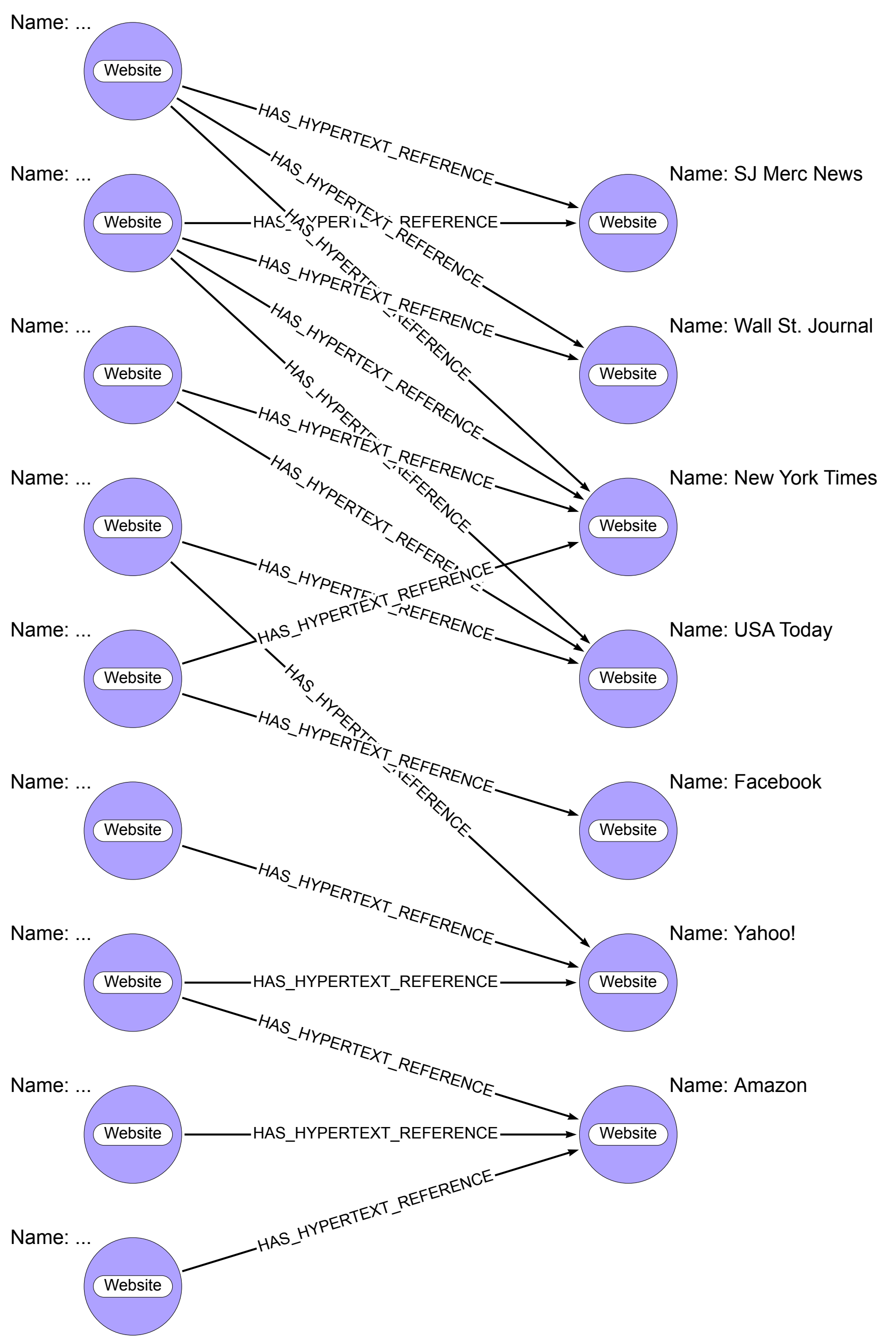

Figure 3.1: HITS algorithm example - Results of the query “newspapers”

In Figure 3.2 all $hubScore$ and $authorityScore$ are set equal to 1. Moreover, in this example two sequence of updates are going to be performed; so the number of steps k is set 2.

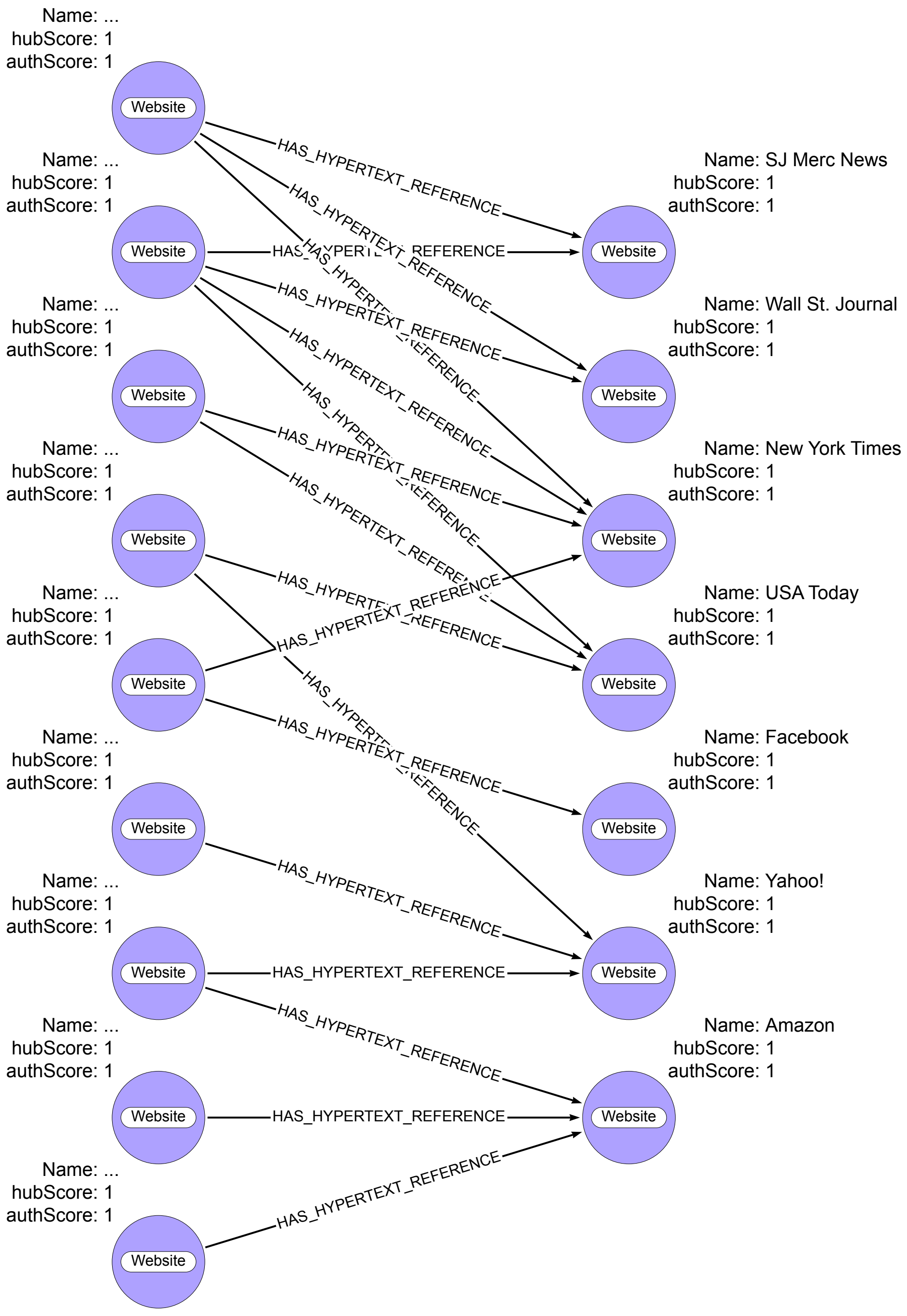

Figure 3.2: HITS algorithm example - Setting all $hubScore = 1$ and $authorityScore = 1$

In Figure 3.3, the authority update rule for step 1 is shown. For each node, $authorityScore$ is updated to be the sum of the $hubScore$ of all nodes that point to the node. Nodes on the left have $authorityScore = 0$ due to no incoming edges. Moreover, the $authorityScore$ of the websites relevant to newspaper is relative increased.

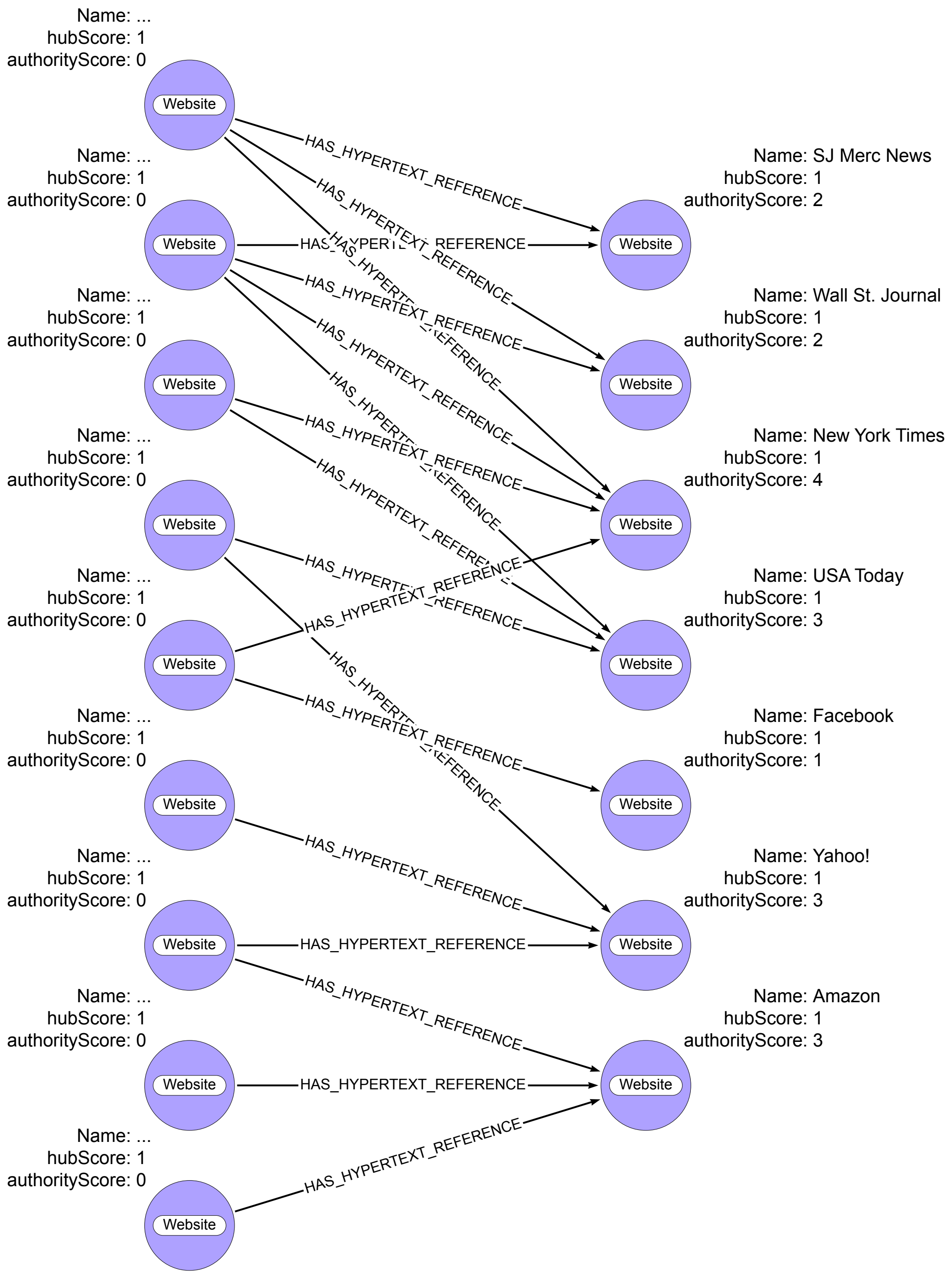

Figure 3.3: HITS algorithm example - Applying authority update rule for step 1

In Figure 3.4, the hub update rule for step 1 is shown. For each node, $hubScore$ is updated to be the sum of the $authorityScore$ for all nodes that the node points to. The set of nodes on the right have $hubScore = 0$ due to no outgoing edges. Moreover, it is important to mention that websites that point to websites relevant to newspaper have higher $hubScore$.

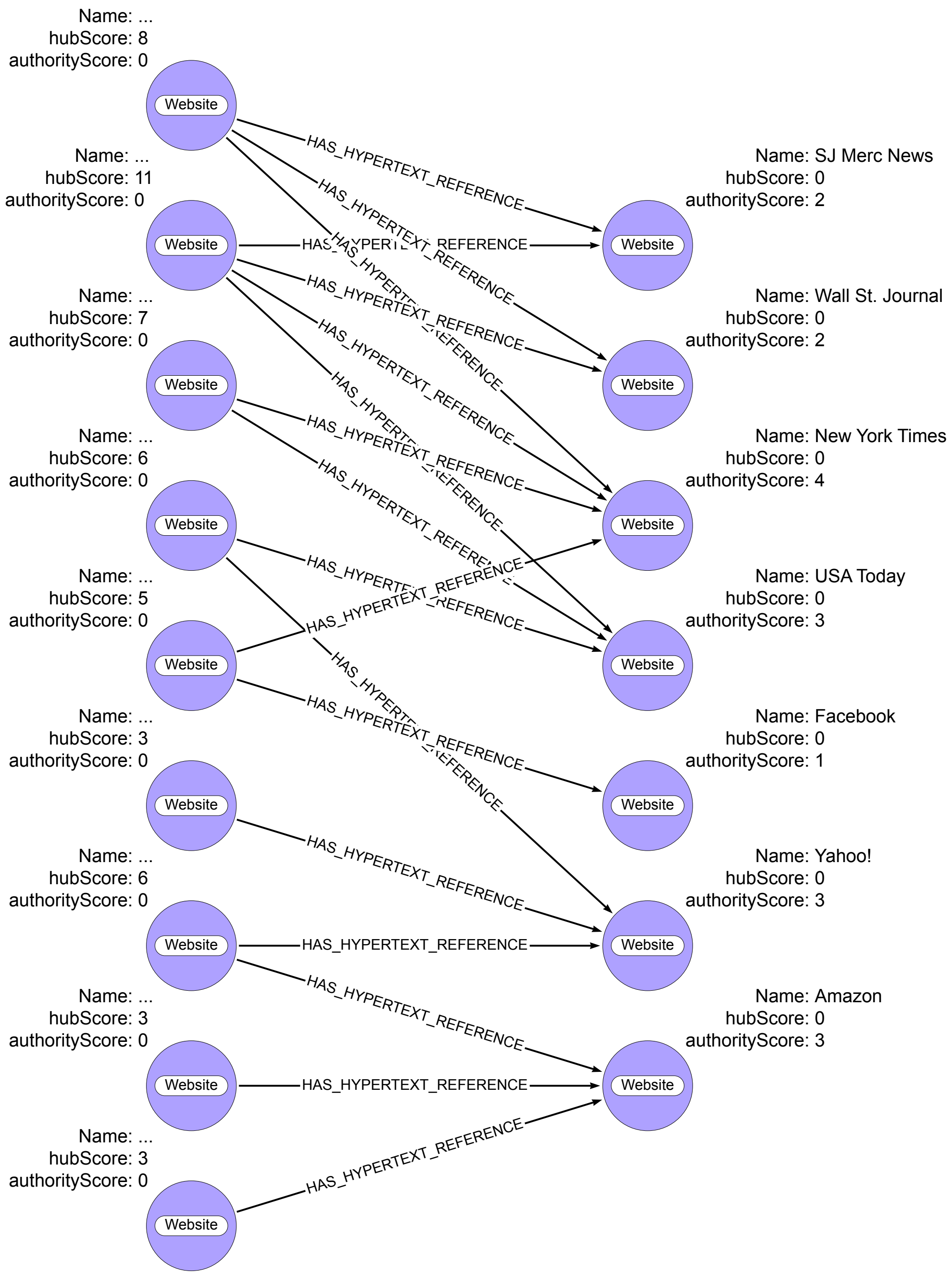

Figure 3.4: HITS algorithm example - Applying hub update rule for step 1

In Figure 3.5, the authority update rule for step 2 is shown. Websites relevant to newspaper have the highest $authorityScore$. That is, the $authorityScore$ will tend to be increased as more hub and authority updates occurs in the following steps.

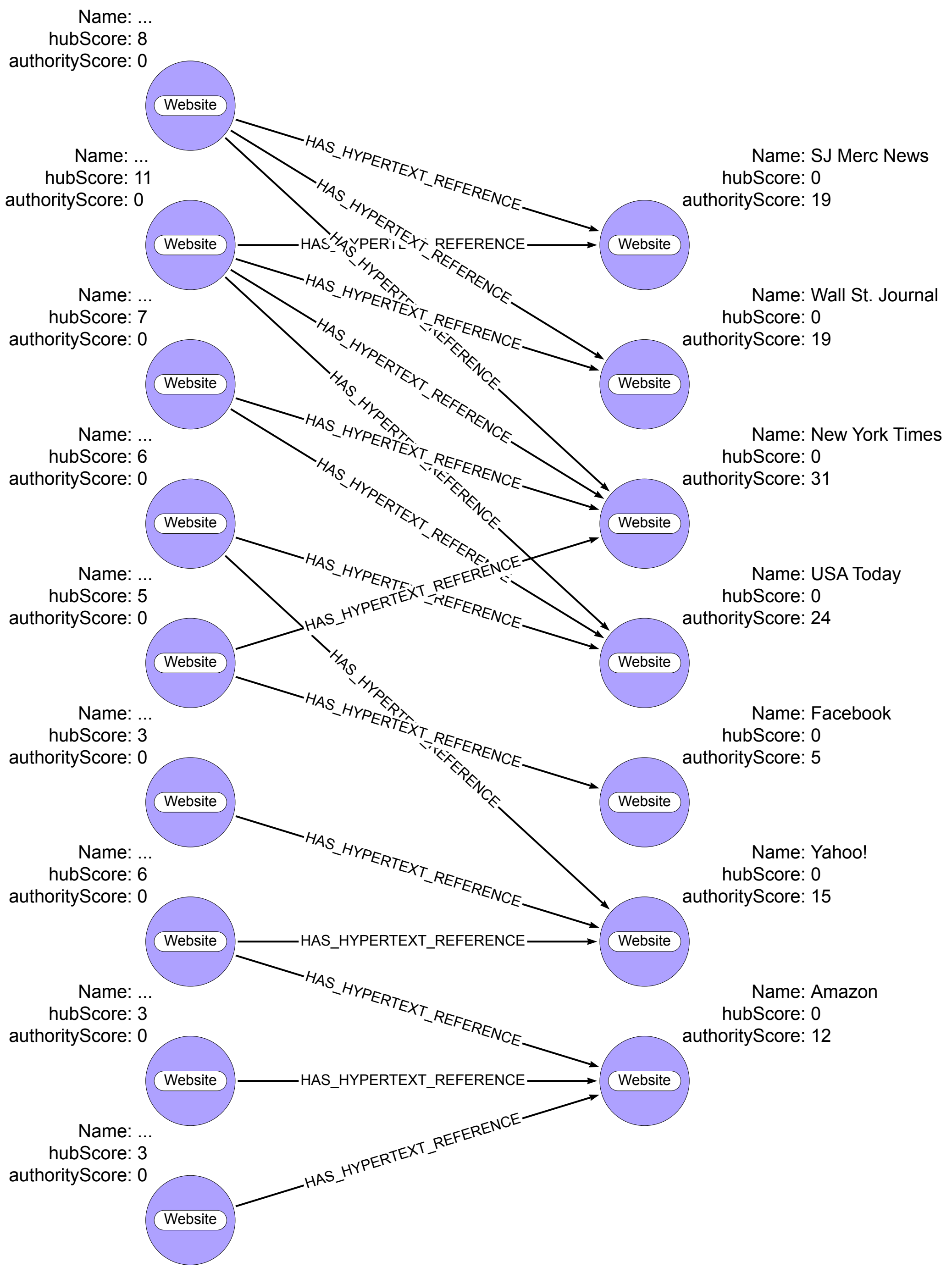

Figure 3.5: HITS algorithm example - Applying authority update rule for step 2

In Figure 3.6, the hub update rule for step 2 is shown. Websites that point only to websites relevant to newspaper tend to have increased $hubScore$. That is, the $hubScore$ will tend to be increased as more hub and authority updates occurs in the following steps.

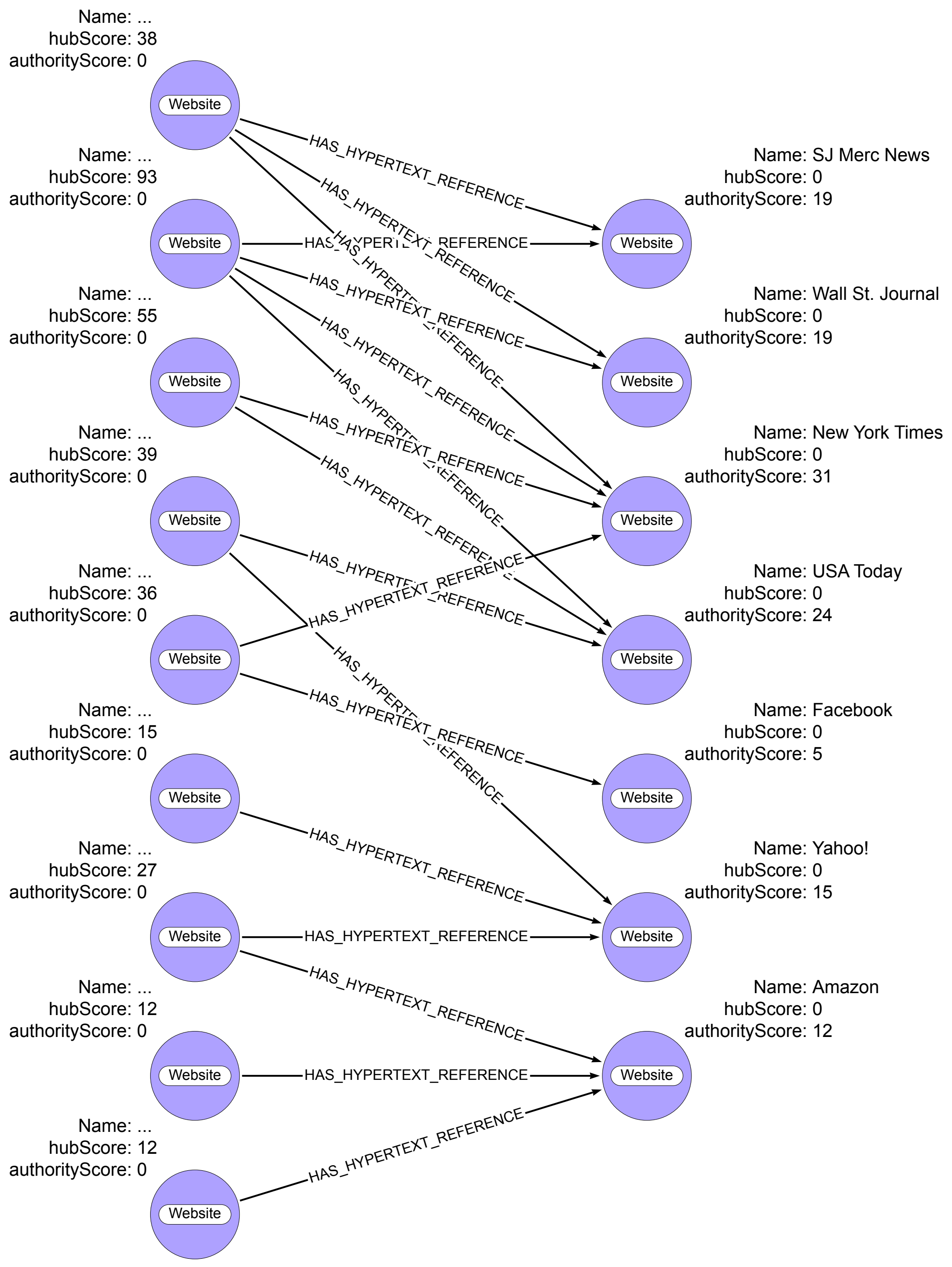

Figure 3.6: HITS algorithm example - Applying hub update rule for step 2

In Figure 3.7, the $authorityScore$ and the $hubScore$ are normalized by dividing down each $authorityScore$ by the sum of all $authorityScore$ and divide down each $hubScore$ by the sum of all $hubScore$.

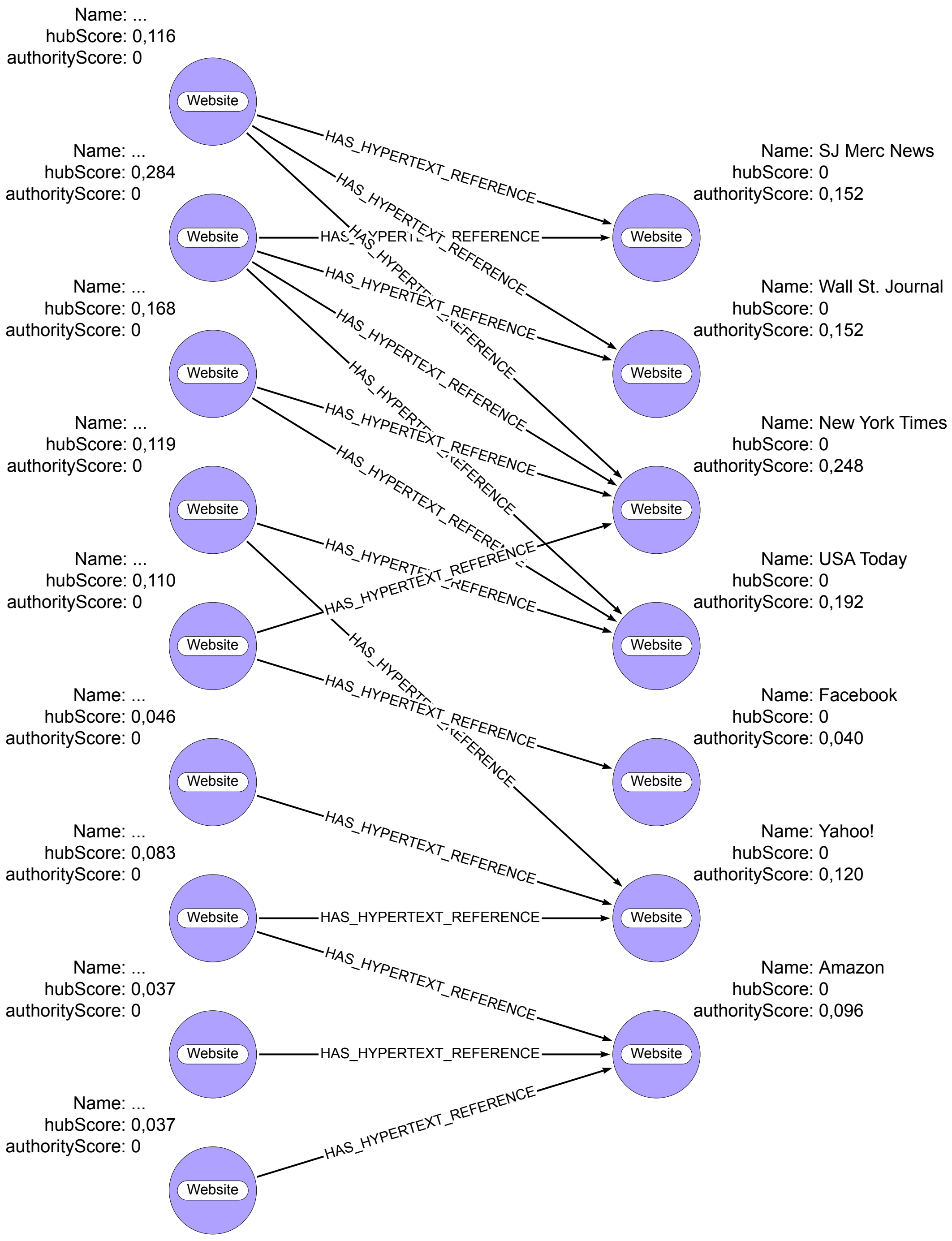

Figure 3.7: HITS algorithm example - Normalizing $authorityScore$ and $hubScore$

4. TOPIC-AWARE INFLUENCE MAXIMIZATION FRAMEWORK

4.1 Definition

The proposed novel framework consists of two main components in order to improve the results of the social influence according to a topic. This is achieved by using a HITS algorithm, which aims to analyze links, and a Greedy or a CELF algorithm with modified Independent Cascade model in order to find k nodes that maximize the expected spread of influence in a network. Using the link analysis results of HITS algorithm, the main goal is to highlight the appropriate nodes for a topic that is examined.

Aiming to calculate the influence maximization spread that takes into account the HITS scores a modified Independent Cascade model is used. The modification also considers the chance to influence a node by the $hubScore$ which is given as input.

The Algorithm 2.1 presents the Independent Cascade model, where a node considers only one chance of influencing each inactive neighbour with probability $p_{(v)}$. Thus, the probability is going to be multiplied by the $hubScore_{(v,u)}$ in order to take advantage of the HITS algorithm. Since the HITS algorithm scores are normalized $p * hubScore < 1$. Below is the change made to the Independent Cascade model.

```
 9: if $rand(0,1) < p_{(v,u)} * hubScore_{(v)}$ then        ▷ Previously: $rand(0,1) < p_{(v,u)}$
10:     $\Phi(t_i) \leftarrow \Phi(t_i) \cup \{u\}$
11: end if
```

The $hubScore$ expresses the value of links of the node and the $authorityScore$ expresses the value of the node. The $hubScore$ is ideal to be used since the goal is to highlight the nodes. In case there is no $hubScore$ for the node an assumption is made that the $hubScore$ is equal to zero.

In Figures 4.1 and 4.2 red nodes are inactive, green are active and all $p_{(v,u)} = 1$. In Figure 4.1 the chance Alice to influence Bob is $0,40 * 1$. Alice has high $hubScore$ after the link analysis on the topic and she is benefited.

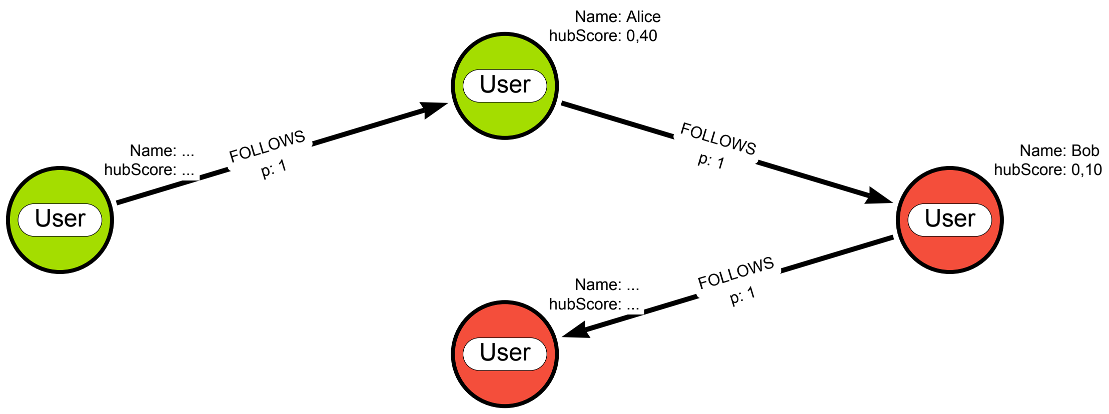

Figure 4.1: Example 1/2 of using $hubScore$ into influence maximization process

On the other hand, in Figure 4.2 the chance Bob to influence other nodes is $0,10 * 1$. Bob has low $hubScore$ after the link analysis on the topic and he is not benefited. In practice, Alice is master on choosing restaurants because she reviews restaurants with high $authorityScore$ and her friends trust her more than Bob.

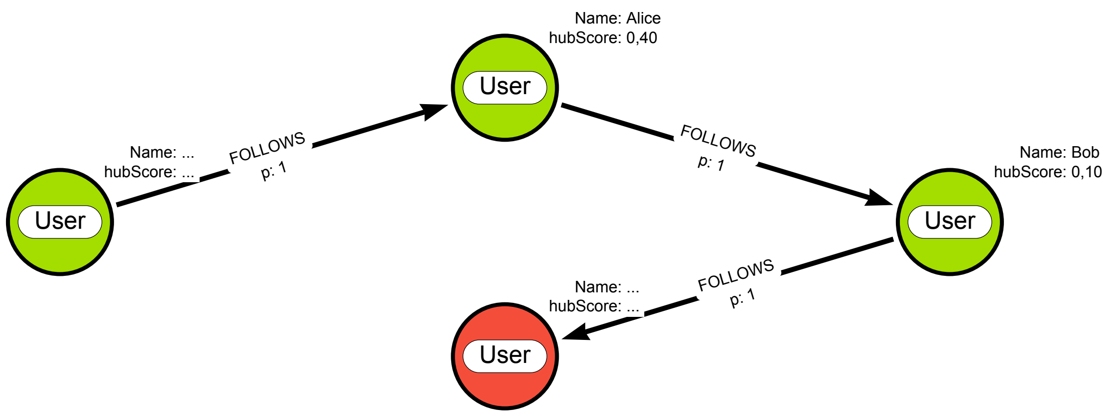

Figure 4.2: Example 2/2 of using $hubScore$ into influence maximization process

Since Bob may or may not likes restaurants the variable $p_{(v,u)}$ should remain. In addition, a use of universal probability based on widely accepted facts is recommended. Facts like, most of people like restaurants so all $p_{(v,u)}$ should be increased, a specific cuisine is less likely to be liked so all $p_{(v,u)}$ should be lowered, etc.

The framework involves the following steps:

1. Create a projected graph G' for a topic, from the initial graph G, using the necessary nodes and edges needed to apply HITS algorithm.
2. Apply HITS algorithm on G' and store HITS algorithm results on G.
3. Create a projected graph G'', from the initial graph G, using the necessary nodes and edges needed to apply either Greedy or CELF algorithm.
4. Apply either Greedy or CELF algorithm with the modified Independent Cascade model on G'' and store influence maximization results on G.

4.2 Example

Figures 4.3, 4.4, 4.5 and 4.6 represents the initial graph G of the example. There are three types of nodes, “User”, “Business” and “Category”, and there are three types of edges, “FOLLOWS”, “REVIEWS” and “IN_CATEGORY”.

In Figure 4.3, the initial network is presented.

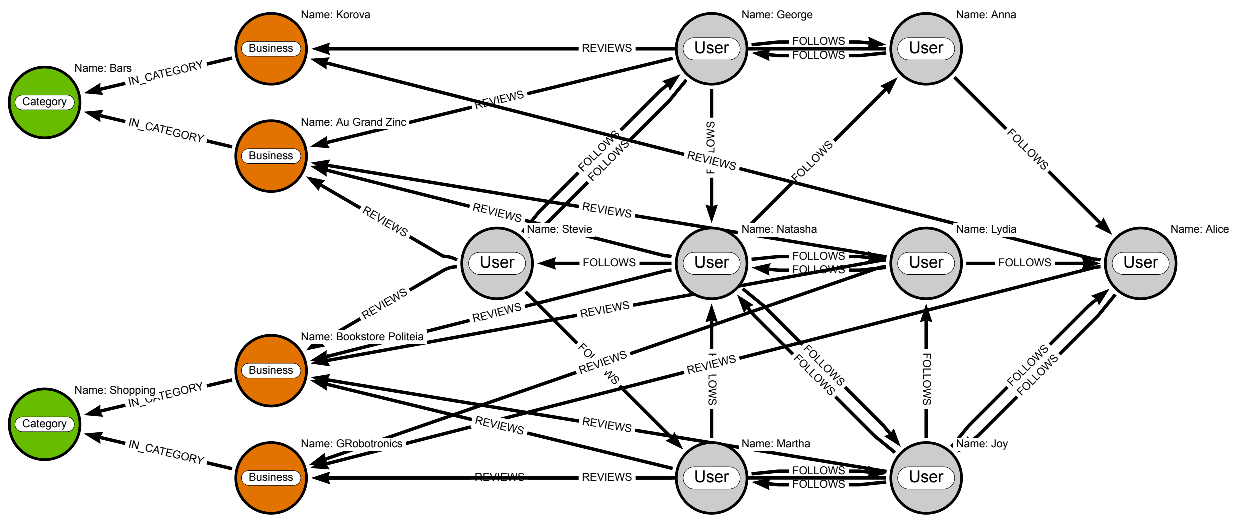

Figure 4.3: Topic-aware influence maximization framework example - The initial network

In Figure 4.4, “FOLLOWS” edges of the initial network are presented.

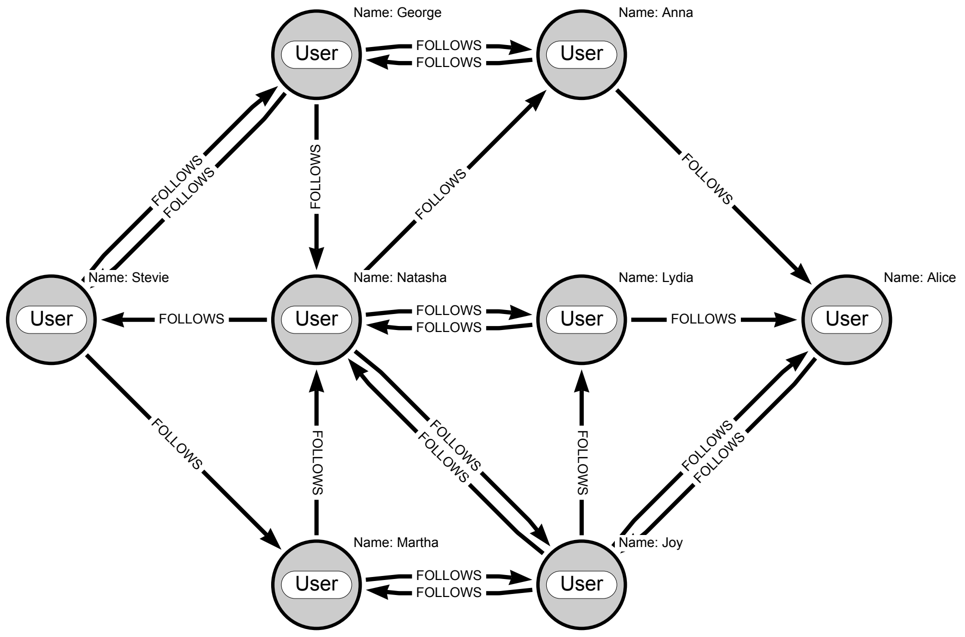

Figure 4.4: Topic-aware influence maximization framework example - “FOLLOWS” edges of the initial network

In Figure 4.5, “REVIEWS” edges of the initial network are presented.

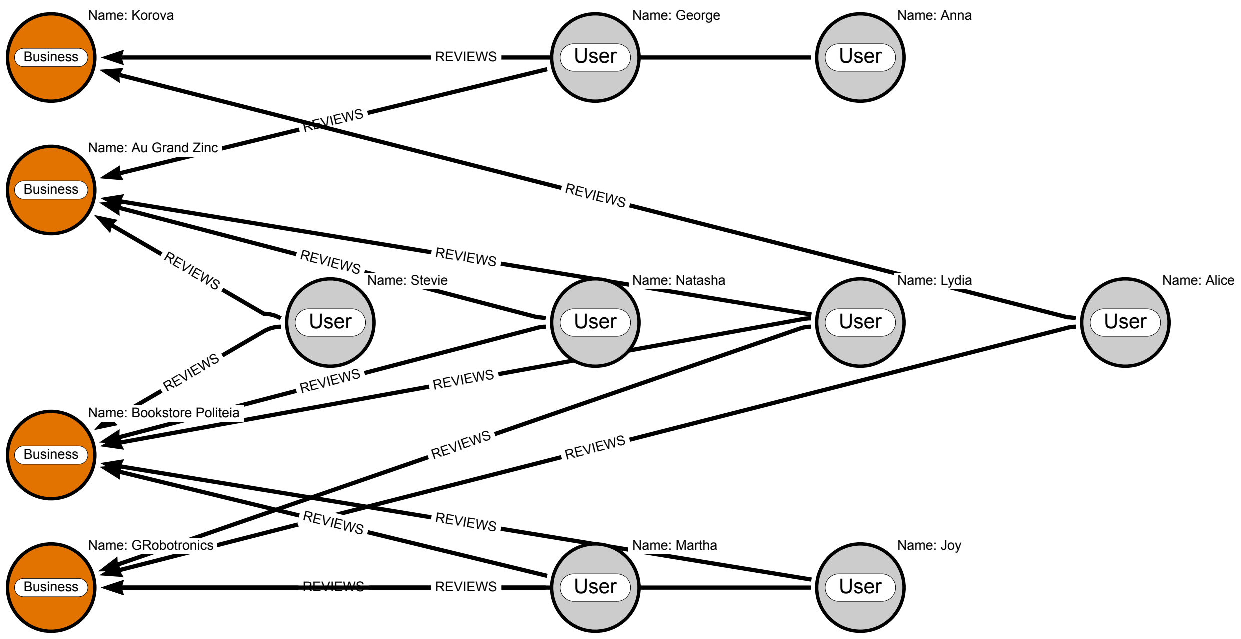

Figure 4.5: Topic-aware influence maximization framework example - “REVIEWS” edges of the initial network

In Figure 4.6, “IN_CATEGORY” edges of the initial network are presented.

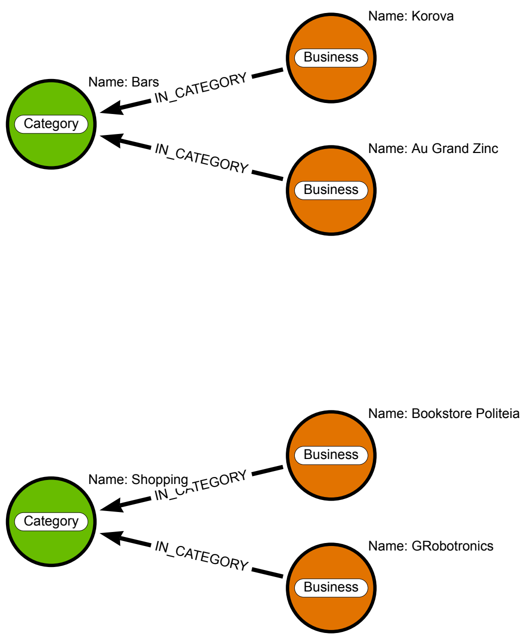

Figure 4.6: Topic-aware influence maximization framework example - “IN_CATEGORY” edges of the initial network

The topics for this network are expressed by “Category” nodes, i.e., Bars or Shopping.

The goal is to find the users that can spread the most for each topic. Based on the graph and what represents, there will be a feature that expresses the topic.

4.2.1 Create a Projected Graph to Apply HITS Algorithm

To apply HITS algorithm, the “User” and “Business” nodes are required. The “REVIEWS” edges connect these two types of nodes. Since the goal is to maximize the spread based on a topic, it is needed to make a projected graph with “User” and “Business” nodes, and “REVIEWS” edges by filtering on each “Category” node.

Figure 4.7 shows the projection by filtering on category Bars. Using this projection, the HITS algorithm is executed to calculate the $hubScore$ and $authorityScore$.

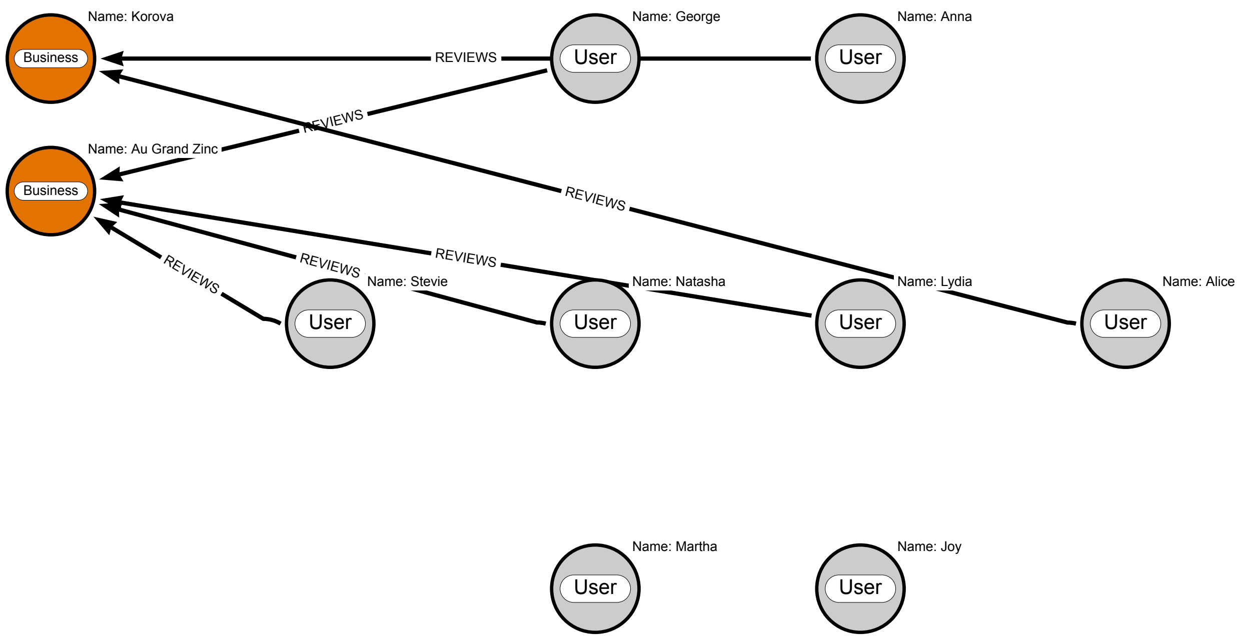

Figure 4.7: Topic-aware influence maximization framework example - Projected graph for topic Bars

Figure 4.8 shows the projection by filtering on category Shopping. Using this projection, the HITS algorithm is executed to calculate the $hubScore$ and $authorityScore$.

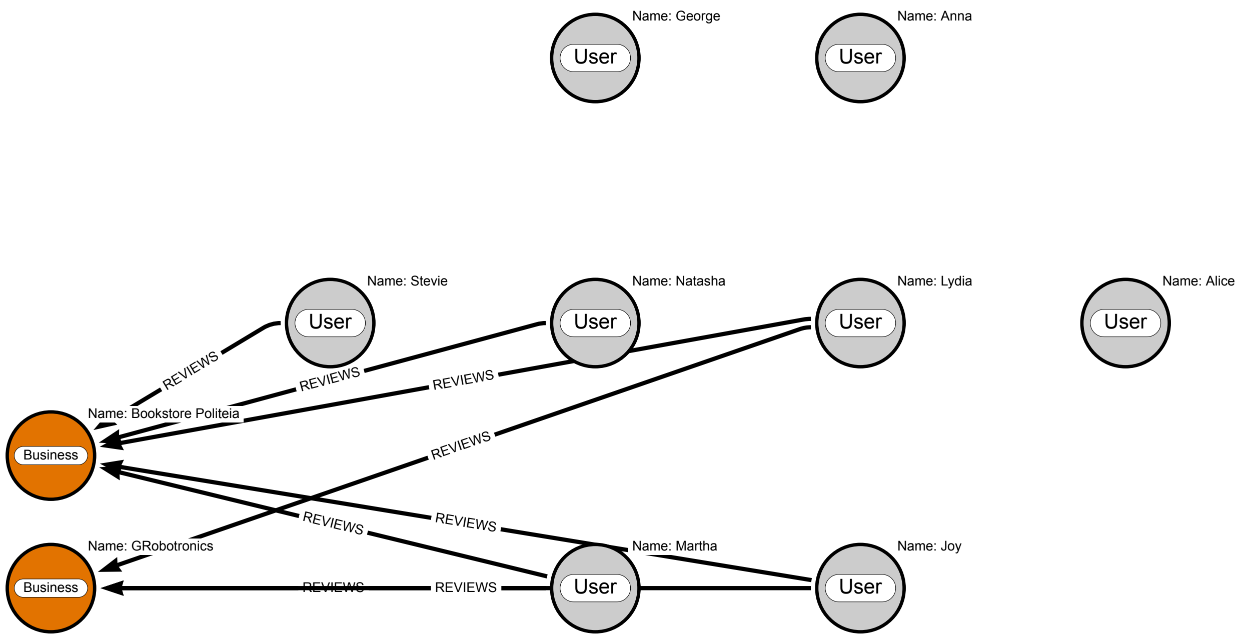

Figure 4.8: Topic-aware influence maximization framework example - Projected graph for topic Shopping

4.2.2 Apply HITS Algorithm

The 2nd step includes the execution of the HITS algorithm. Tables 4.1 and 4.2 show the results of $hubScore$ and $authorityScore$.

Table 4.1: Topic-aware influence maximization framework example - HITS algorithm results for topic Bars

Name	Type	$hubScore$	$authorityScore$
Alice	User	0,00	0,00
Anna	User	0,00	0,00
George	User	0,50	0,00
Joy	User	0,00	0,00
Lydia	User	0,50	0,00
Martha	User	0,00	0,00
Natasha	User	0,50	0,00
Stevie	User	0,50	0,00
Au Grand Zinc	Business	0,00	0.99
Korova	Business	0,00	0.01

Table 4.2: Topic-aware influence maximization framework example - HITS algorithm results for topic Shopping

Name	Type	$hubScore$	$authorityScore$
Alice	User	0.24	0,00
Anna	User	0,00	0,00
George	User	0,00	0,00
Joy	User	0.51	0,00
Lydia	User	0,51	0,00
Martha	User	0.51	0,00
Natasha	User	0,50	0,00
Stevie	User	0,27	0,00
Bookstore Politeia	Business	0,00	0.76
GRobotronics	Business	0,00	0.64

4.2.3 Create a Projected Graph to Apply Greedy or CELF Algorithm

Figure 4.4 shows “FOLLOWS” edges among “User” nodes. Using this projection an influence maximization algorithm, either Greedy or CELF, is executed to calculate the influence spread. Either Greedy or CELF algorithm should run, in this case, between users and their friendship relation.

4.2.4 Apply Greedy or CELF Algorithm

In the final step of topic-aware influence maximization framework, the $hubScore$ from a previous step is considered as input parameter of Greedy or CELF algorithms. Tables 4.3 and 4.4 show the results of the spread for the topics.

Table 4.3: Topic-aware influence maximization framework example - Influence maximization results for topic Bars

Name	Spread
Natasha	1,150
George	2,290
Lydia	3,330
Stevie	4,330
Anna	5,330
Joy	6,330
Martha	7,330
Alice	8,330

Table 4.4: Topic-aware influence maximization framework example - Influence maximization results for topic Shopping

Name	Spread
Joy	1,200
Natasha	2,300
Lydia	3,340
Stevie	4,336
George	5,360
Anna	6,360
Martha	7,360
Alice	8,360

In addition, a topic-blind influence maximization is executed in order to compare the topic results with the topic-blind results. Table 4.5 shows the results of the spread for a topic-blind.

Table 4.5: Topic-aware influence maximization framework example - Influence maximization results on topic-blind

Name	Spread
George	1,747
Natasha	2,937
Anna	4,036
Joy	5,116
Lydia	6,196
Stevie	7,196
Martha	8,196
Alice	9,196

Based on the derived results, is observed the top nodes differs for each category and topic-blind case.

5. EXPERIMENTS

5.1 Software

A graph database, specifically the Neo4j graph database (Subsection 5.1.1), was used as a platform for the empirical experiments. For the needs of thesis, Greedy and CELF algorithms were implemented for Neo4j. Those implementations have been contributed to the Neo4j open-source community and released in version 1.6 of the GDS library (Subsection 5.1.2).

Graph databases belong to NoSQL database based on graph theory that addresses the limitations of relational databases. Those database uses graph structures for semantic queries with nodes, edges, and properties to represent and store data.

The graph, as the key concept of the system, relates the data items to a collection of nodes and edges. Graph databases portray the data conceptually by transferring the data into nodes and its relationships into edges.

Graph databases have become especially popular nowadays with the growth of the needs of social networks and the needs of analyzing complex networks. Some of the well-known such databases are Neo4j, ArangoDB, Dgraph, OrientDB, etc.

5.1.1 Neo4j

Neo4j is the most popular graph database management system. The community edition is licensed under the free GNU General Public License v3 while the enterprise edition is licensed under a Neo4j commercial license.

The Neo4j is an ACID compliant transactional database with native graph storage and processing. In Neo4j everything is stored in the form of node, edge, and property. Each node and edge can have any number of properties. Both nodes and edges can be labelled. Labels can be used to narrow searches. Neo4j supports indexes.

Cypher declarative query language, used in Neo4j, allows expressive and efficient data querying. The language is designed with the power and capability of SQL in mind, but Cypher is based on the components and needs of a database built upon the concepts of graph theory.

5.1.2 Graph Data Science (GDS) Library

Graph Data Science (GDS) library uses the graph database and provides many algorithms. Graph algorithms are used to compute metrics for graphs, nodes, or relationships. They can provide insights on relevant entities in a graph (centralities, ranking), or inherent structures like communities (community detection, graph partitioning, clustering). Their algorithms are efficiently implemented and parallel.

To run the algorithms as efficiently as possible, the GDS library uses a specialized in memory graph format to represent the graph data. It is therefore necessary to load the graph data from the Neo4j database into an in-memory graph catalogue. The amount of data loaded can be controlled by so called graph projections, which also allow, for example, filtering on node labels and relationship types, among other options. Figure 5.1 shows the standard operations of the GDS library.

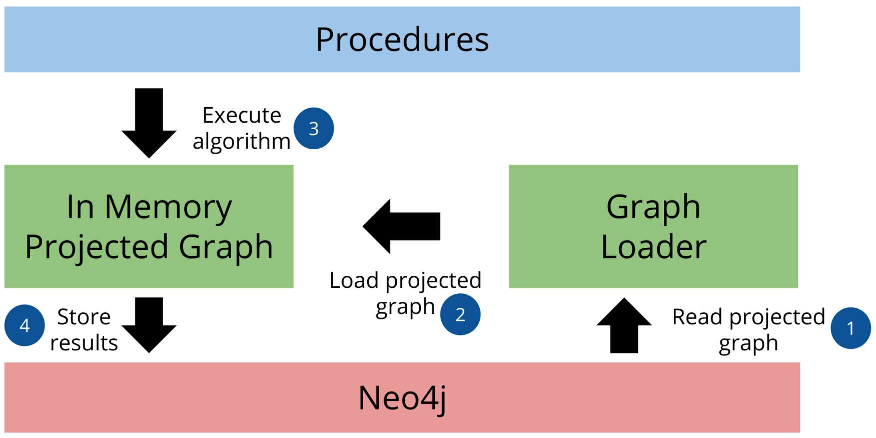

Figure 5.1: GDS library - Standard operations

5.2 Data Sources

For the experiments, two type of data source are used. Generated graphs, based on the Barabási–Albert model (Subsection 5.2.1), were used to evaluate the performance of the algorithms and the Yelp dataset (Subsection 5.2.2) was used to evaluate the proposed framework.

5.2.1 Barabási–Albert Model

The Barabási–Albert model [12] is an algorithm for generating random scale free networks using a preferential attachment mechanism. The algorithm is introduced by Albert-László Barabási and Réka Albert.

Many observed networks fall into the class of scale free networks, meaning that they have power law degree distributions, while random graph models do not exhibit power laws. The Barabási–Albert model is one of the several proposed models that generate scale free networks.

It incorporates two important general concepts:

Growth means that the number of nodes in the network increases over time.

Preferential attachment means that the more connected a node is, the more likely is to receive new links.

Both growth and preferential attachment exist widely in real networks.

Awesome Procedures on Cypher (APOC) library is a Neo4j library that includes a procedure to generate a random graph according to the Barabási–Albert model. Figure 5.2 represents the schema that is used for the experiments.

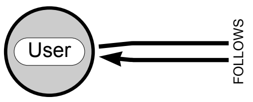

Figure 5.2: Barabási–Albert model - Schema used in topic-blind experiments

5.2.2 Yelp Dataset

Yelp's website, yelp.com, is a crowd-sourced local business review and social networking website.

The website includes pages devoted to individual locations, such as restaurants, schools, etc., where Yelp users can submit a review of their products or services using a one-to-five-star rating system. In addition to writing reviews, users can react to reviews, etc.

The Yelp dataset is a subset of Yelp businesses, reviews, and user data for use in personal, educational, and academic purposes. Figure 5.3 represents the schema of Yelp dataset that is used for the experiments. Table 5.1 shows the number of each label of nodes and edges.

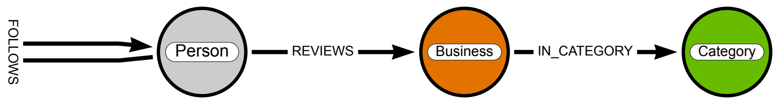

Figure 5.3: Yelp dataset - Schema used in topic-aware experiments

Table 5.1: Yelp dataset - Number of nodes/edges per type

Type	Number of nodes/edges
User	22.081.413
Business	160.585
Category	1.330
FOLLOWS	116.030.305
REVIEW	8.345.514
IN_CATEGORY	708.884

5.3 Performance Experiments on Greedy and CELF Algorithms

In this section, the performance of the implementations that contributed in Neo4j community is examined. Besides testing, behavior of each algorithm is also observed.

All experiments were conducted in generated graphs, via Neo4j APOC library, based on Barabási–Albert model. The experiments were conducted in a personal computer (Windows 10, Intel i7 4-cores, 16GB ram) using Neo4j enterprise v4.3.1, APOC v4.3.0.0 and GDS v1.6.1.

In Appendix A, tables with the settings of parameters and the running times of all experiments are listed.

5.3.1 Greedy/CELF Algorithms Experiment

In this experiment the advantage of CELF algorithm that uses the Lazy Forward feature is observed. Figure 5.4 shows that the running time of Greedy algorithm escalates, as expected, much faster than CELF algorithm running time.

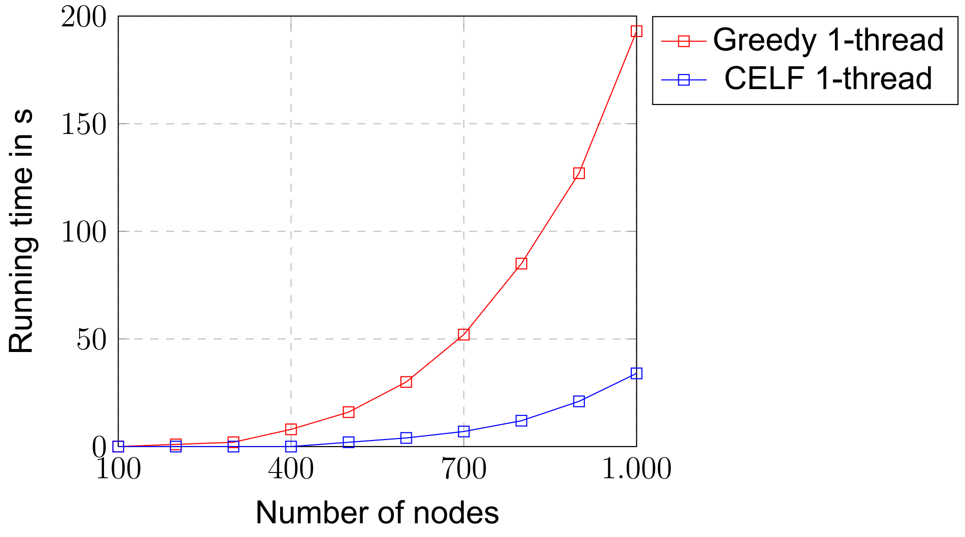

Figure 5.4: Performance experiment - Greedy/CELF algorithms

5.3.1.1 Multithreads

In the same experiment it is worth to compare, in Figures 5.5, 5.6, and 5.7, the multithreading implementations. Threading the function of the Independent Cascade model offers advantage to the Greedy algorithm. On the other hand, due to Lazy Forward feature, the CELF algorithm uses the Independent Cascade model much lower times, and threads do not offer any significant advantage.

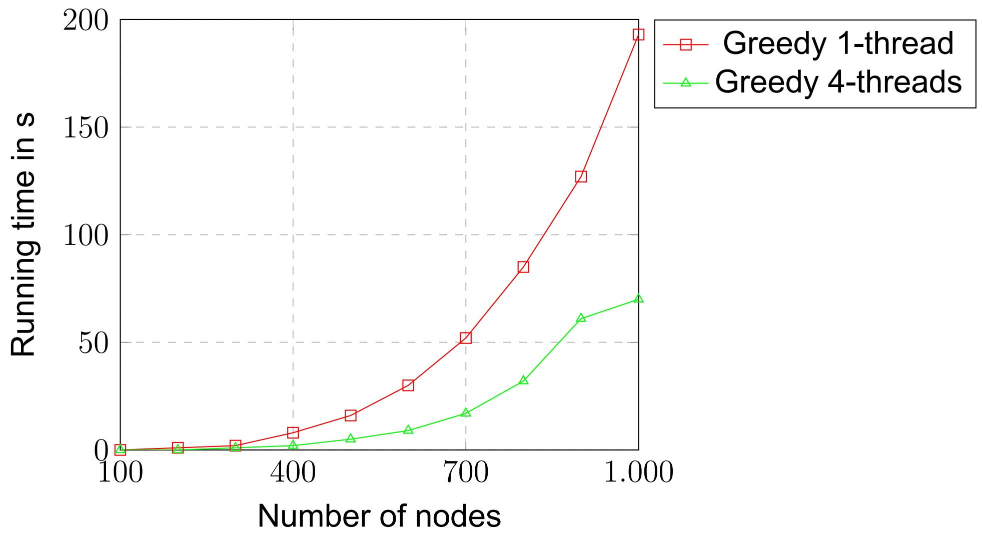

Figure 5.5: Performance experiment - Greedy algorithm multithreads

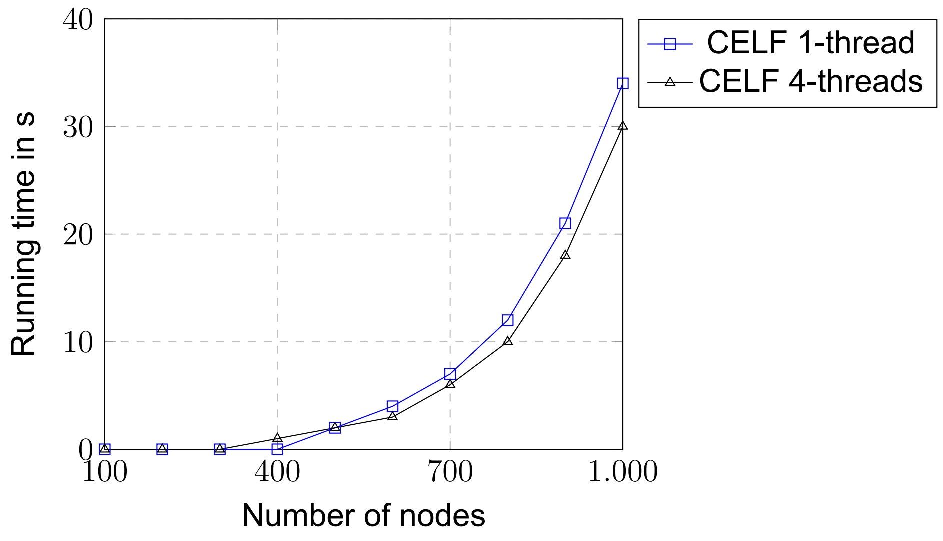

Figure 5.6: Performance experiment - CELF algorithm multithreads

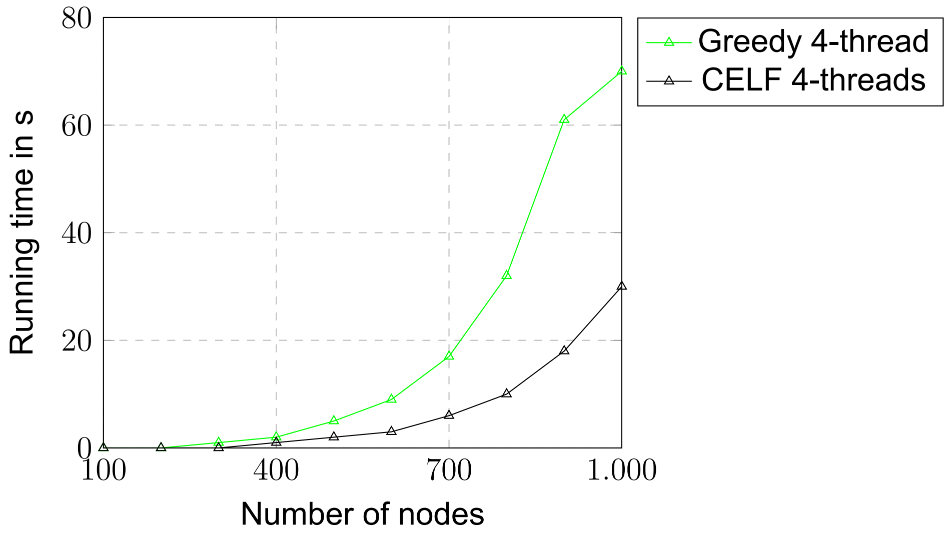

Figure 5.7: Performance experiment - Greedy/CELF algorithms multithreads

5.3.2 Large-Scale Networks Experiment

In this experiment, in Figure 5.8, the behavior in large scale networks is tested. For instance, in a large-scale network with 10.000 nodes and 747.150 edges, Greedy algorithm needs almost 4 hours to be executed and CELF algorithm about 1 hour and 30 minutes. Therefore, CELF algorithm should be preferred in large scale networks.

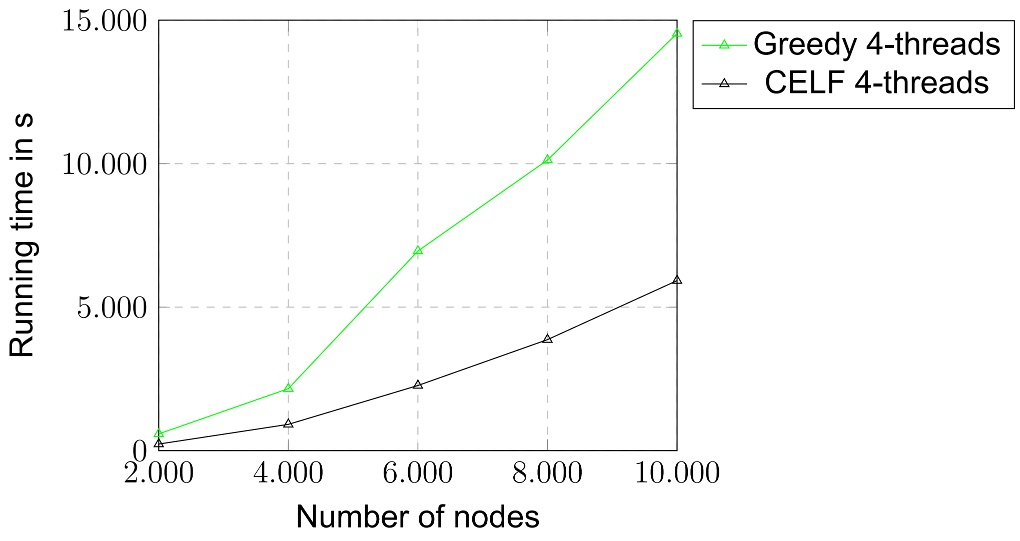

Figure 5.8: Performance experiment - Number of nodes

5.3.3 Seed Set Size Experiment

In this experiment the impact of requesting a large seed set is observed. Once again, CELF algorithm is not affected because in Lazy Forward implementation the use of the Independent Cascade mode is minimal. Figure 5.9 shows that Greedy algorithm escalate as the seed set increases.

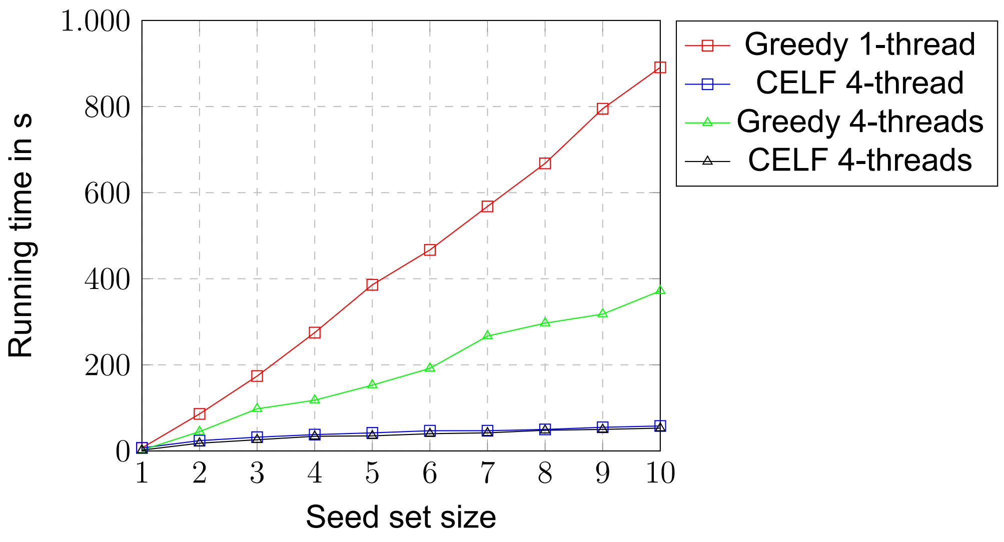

Figure 5.9: Performance experiment - Seed set size

5.3.4 Monte-Carlo Simulations Experiment

In this experiment, in Figures 5.10 and 5.11, the impact of running large number of Monte-Carlo simulations is tested. As the number of simulations increased it is logical the running time to increases. Moreover, it is observed the need of multithreads which reduces significantly the running time.

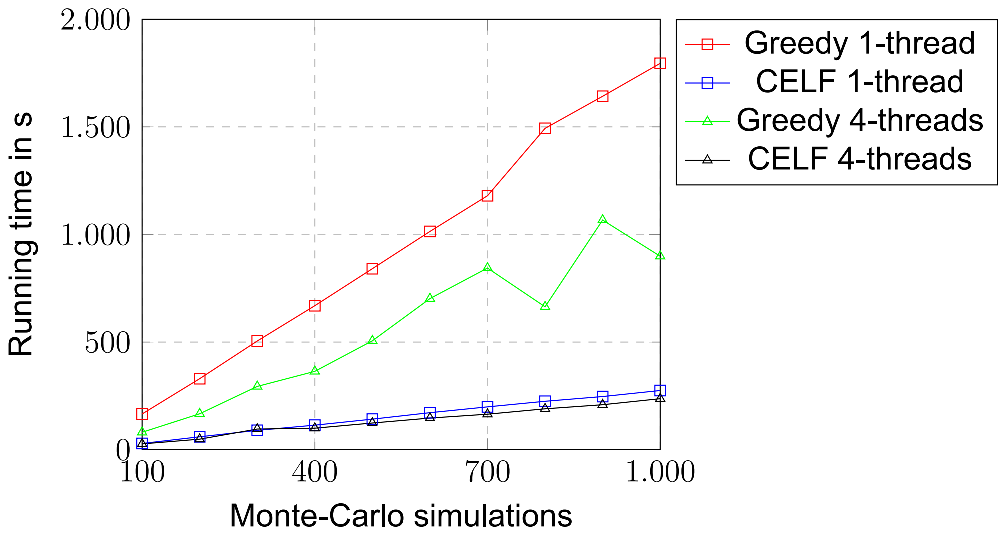

Figure 5.10: Performance experiment - Monte-Carlo simulations (part 1/2)

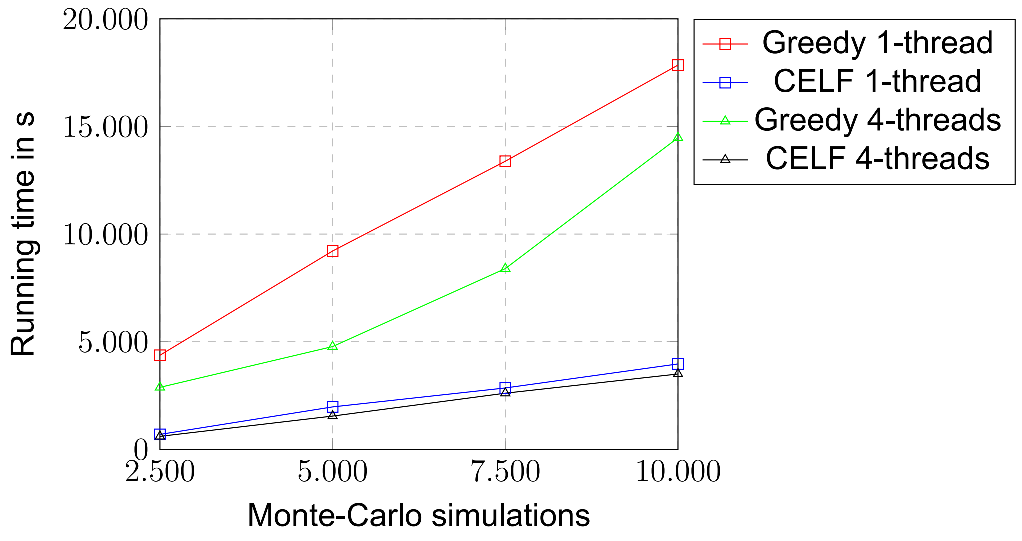

Figure 5.11: Performance experiment - Monte-Carlo simulations (part 2/2)

Table 5.2 shows how the spread differences among various Monte-Carlo simulations. As the number of simulations increases, the spread also tents to increases. Since this is not a significant gap the overall spread could be considered as stable.

Table 5.2: Performance experiment - Spread with various number of Monte-Carlo simulations

Monte-Carlo simulations	1st Node spread	2nd Node spread	3nd Node spread
100	3.557,91	3.893,50	4.071,59
200	3.586,83	3.921,85	4.098,34
300	3.727,16	4.057,15	4.225,91
400	3.671,89	4.003,20	4.176,61
500	3.679,53	4.016,54	4.193,38
600	3.643,50	3.985,19	4.164,75
700	3.660,69	3.999,88	4.179,34
800	3.675,49	4.011,57	4.190,66
900	3.646,51	3.982,05	4.162,39
1.000	3.649,77	3.985,42	4.165,47
2.500	3.681,92	4.017,86	4.197,55
5.000	3.890,60	4.194,24	4.355,33
7.500	3.973,63	4.261,81	4.415,36
10.000	3.970,03	4.258,12	4.411,54

5.3.5 Propagation Probability Experiment

In this experiment, in Figure 5.12, the impact of propagation probability is observed. As the propagation probability increases, more nodes tend to become active and therefore the algorithm needs more time to examine the increased active nodes.

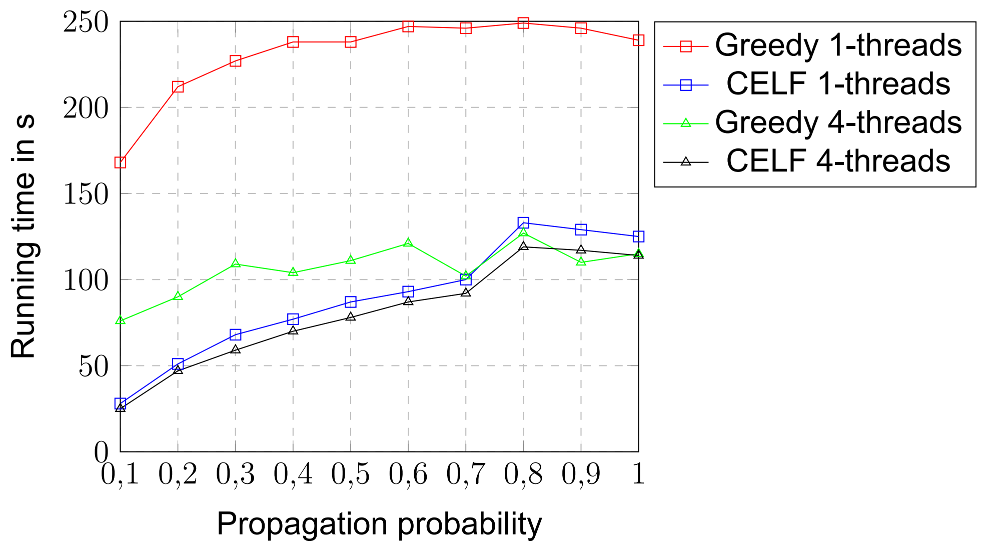

Figure 5.12: Performance experiment - Propagation probability

5.4 Evaluating Topic-Aware Results

In this section, the proposed topic-aware influence maximization framework is tested in the Yelp dataset.

The following executions were conducted in a dedicated virtual machine (Ubuntu 20.04 server, 4 vCPUs, 32GB ram) with Neo4j enterprise v4.2.7, APOC v4.2.0.5 and GDS v1.6.2. The Neo4j installation was configured in order to handle the Yelp graph as optimally as possible.

Table 5.3 displays the top 10 categories based on the number of businesses. The total number of categories is 1.330.

Table 5.3: Yelp dataset - Top 10 categories based on the number of businesses

Category	Number of businesses
Restaurants	50.763
Food	29.469
Shopping	26.205
Beauty & Spas	16.574
Home Services	16.465
Health & Medical	15.102
Local Services	12.192
Nightlife	11.990
Bars	10.741
Automotive	10.119

Firstly, a topic-blind influence maximization was executed to retrieve the nodes that spread the most regardless topic. The results are shown in Table 5.4.

Table 5.4: Evaluating experiment - 10 users with the highest topic-blind influence in Yelp dataset

ID	Name	Spread
1053362	Walker	1.501,63
475465	Ruggy	2.733,34
683280	Randy	3.814,21
900364	Scott	4.805,63
645273	Steven	5.760,70
900460	Danny	6.683,02
314538	Katie	7.576,80
303635	Abby	8.455,31
2208928	Rodney	9.331,71
916387	Vince	10.156,42

Then, a topic-aware influence maximization was executed to retrieve the nodes that spread the most for top 3 categories. The results are shown in Tables 5.5, 5.6, and 5.7.

Table 5.5: Evaluating experiment - 10 users with the highest influence for Restaurant category in Yelp dataset

ID	Name	Spread
167347	Damien	29,82
176183	Kelly	52,19
165009	Andrew	70,29
429006	Don	87,39
181090	Daniel	104,25
185819	Chris	120,07
176435	Leighann	135,20
223111	Michael	150,18
180803	Ligaya	163,50
166192	Michael	176,05

Table 5.6: Evaluating experiment - 10 users with the highest influence for Food category in Yelp dataset

ID	Name	Spread
429006	Don	24,33
165009	Andrew	48,63
223111	Michael	71,40
224369	Matt	91,51
172160	Laura	105,44
166192	Michael	117,19
683280	Randy	128,19
475465	Ruggy	138,79
207225	Lorrie	148,72
176183	Kelly	158,49

Table 5.7: Evaluating experiment - Users with the highest influence for Shopping category in Yelp dataset

ID	Name	Spread
683280	Damien	15,73
240628	Kelly	28,85
224369	Matt	41,44
172160	Laura	53,27
429006	Don	65,04
381836	Brittany	76,32
207225	Lorrie	87,33
223111	Michael	97,93
164736	Ed	108,49
168615	Alden	118,72

The users that maximize the influence differ between the results on each category and on a topic-blind.

Table 5.8 shows the parameters of CELF and HITS algorithms that used in topic-aware experiment and Table 5.9 shows the running time.

Table 5.8: Evaluating experiment - Parameters of topic-aware influence maximization with HITS algorithm

Parameter	Value
Seed set size	10
Monte-Carlo simulations	100
Propagation probability	0.1
Threads	4
HITS algorithm steps	10

Table 5.9: Evaluating experiment - Running time per topic of topic-aware influence maximization with HITS algorithm

Topic	#Businesses	#REVIEWS	HITS running time	CELF running time	HITS + CELF running time
NONE	-	-	-	344s	344s
Restaurants	50.763	5.395.602	39s	142s	181s
Food	29.469	2.245.004	13s	139s	152s
Shopping	26.205	674.098	5s	136s	141s

6. CONCLUSIONS

In this thesis, the problem of Topic-Aware Influence Maximization was studied and a novel framework proposed for solving it. The studied problem is a realistic model of real-life applications, and it has started to be used for promoting products, innovations and opinions exploiting the social influence among communities. The novel proposed framework aims to maximize the social influence on the basis of a topic utilizing the link analysis algorithm HITS and an approximation influence maximization algorithm (i.e., Greedy or CELF) by creating a pipeline between those algorithms. In addition, the Greedy and the CELF algorithms are further examined evaluating their performances under various conditions in an attempt to identify their advantages and disadvantages.

It should be highlighted that the proposed framework is extremely efficient for analysts that are using networks for modelling and the information about the costs of nodes are missing. The proposed framework could benefit to optimize decision making or determining proper choices. The developed Greedy and CELF algorithms submitted to the Neo4j graph database open-source community (and accepted by the Neo4j team) contributing to the version 1.6 of the Neo4j Graph Data Science (GDS) library.

The conclusions extracted during this thesis can constitute the basis for future research. Inspired by the proposed solution framework, a worth pursing research direction is towards the analysis considering various operational constraints such as the budget, the time and their combination as well as other issues.

In particular,

- The budget in the spread of influence problem in social networks is considered as an amount of the resource that can be spent on influencing nodes. This resource is more often expressed as a budget k of different nature such as money, gifts, conversations. However, each successful influence of a node in the network reduces the budget. Typically, it is assumed that the amount of a budget taken for influencing a node is equal for all of the nodes in network. In some cases, it may be true and may happen. For instance, in a marketing campaign the same product is being sent to different customers and the cost of the product distribution among them is estimated to be equal. On the other hand, as the influence process is subjective, spending some amount of the budget on a user does not mean that s/he will be influenced due to their susceptibility, which differs from user to user.

- The time constraint means that nodes should be influenced within a given time and the evaluation of the results is considered as a termination condition. Typically, the models work until no more nodes could be influenced. In case of implementing time constraint, the process may be evaluated sooner. Considering the temporal social networks, the use of this strict termination condition could be more complicated as in case the network changes, the influence process may be infinite. For this reason, the time constraint is crucial for temporal social networks by introducing a moment which allows comparisons. From the perspective of marketers, they invest time and expect to have the return on this investment within a predefined time frame (sales period, offer seasons etc.). In case of adopting good manners, studies revealed that the sooner the better.

ABBREVIATIONS - ACRONYMS

ACID	Atomicity Consistency Isolation Durability
APOC	Awesome Procedures on Cypher
CELF	Cost-Effective Lazy Forward
HITS	Hyperlink-Induced Topic Search
GDS	Graph Data Science

APPENDIX A. EXPERIMENTS RUNNING TIMES AND PARAMETERS

Table A.1: Performance experiment - Greedy/CELF algorithms running times and parameters

#Nodes	#Edges	#Edges per Node	Threads	Seed Set Size	Monte-Carlo Simulations	Propagation Probability	Greedy running time	CELF running time
100	485	5	1	3	100	0,1	0s	0s
			4				0s	0s
200	1.945	10	1				1s	0s
			4				0s	0s
300	4.380	15	1				2s	0s
			4				0s	0s
400	7.790	20	1				8s	0s
			4				2s	0s
500	12.175	25	1				16s	2s
			4				5s	2s
600	17.535	30	1				30s	4s
			4				9s	3s
700	23.870	35	1				52s	7s
			4				17s	6s
800	31.180	40	1				85s	12s
			4				32s	10s
900	39.465	45	1				127s	21s
			4				61s	18s
1.000	48.725	50	1				193s	34s
			4				70s	30s

Table A.2: Performance experiment - Large-scale networks running times and parameters

#Nodes	#Edges	#Edges per Node	Threads	Seed Set Size	Monte-Carlo Simulations	Propagation Probability	Greedy running time	CELF running time
2.000	147.150	75	4	3	100	0,1	584s	230s
4.000	297.150						2.363s	916s
6.000	447.150						6.956s	2.271s
8.000	597.150						10.127s	3.868s
10.000	747.150						14.536s	5.925s

Table A.3: Performance experiment - Seed set size running times and parameters

#Nodes	#Edges	#Edges per Node	Threads	Seed Set Size	Monte-Carlo Simulations	Propagation Probability	Greedy running time	CELF running time
1.000	48.725	50	1	1	100	0,1	7s	7s
			4				2s	2s
			1	2			86s	24s
			4				45s	18s
			1	3			174s	32s
			4				98s	26s
			1	4			275s	38s
			4				118s	34s
			1	5			386s	42s
			4				153s	35s
			1	6			467s	47s
			4				192s	40s
			1	7			568s	47s
			4				267s	42s
			1	8			668s	50s
			4				297s	48s
			1	9			795s	55s
			4				318s	50s
			1	10			891s	58s
			4				372s	53s

Table A.4: Performance experiment - Monte-Carlo simulations running times and parameters

#Nodes	#Edges	#Edges per Node	Threads	Seed Set Size	Monte-Carlo Simulations	Propagation Probability	Greedy running time	CELF running time
1.000	48.725	50	1	3	100	0,1	166s	29s
			4				81s	27s
			1		200		330s	60s
			4				167s	49s
			1		300		505s	90s
			4				294s	96s
			1		400		669s	114s
			4				364s	100s
			1		500		841s	142s
			4				506s	124s
			1		600		1.014s	172s
			4				702s	147s
			1		700		1.180s	199s
			4				844s	165s
			1		800		1.493s	225s
			4				664s	190s
			1		900		1.642s	247s
			4				1.067s	209s
			1		1.000		1.795s	275s
			4				899s	237s
			1		2.500		4.370s	693s
			4				3.879s	603s
			1		5.000		9.214s	1.971s
			4				4.774s	1.545s
			1		7.500		13.384s	2.851s
			4				8.396s	2.608s
			1		10.000		17.852s	3.967s
			4				14.476s	3.500s

Table A.5: Performance experiment - Propagation probability running times and parameters

#Nodes	#Edges	#Edges per Node	Threads	Seed Set Size	Monte-Carlo Simulations	Propagation Probability	Greedy running time	CELF running time
1.000	48.725	50	1	1	100	0,1	168s	28s
			4				76s	25s
			1			0,2	212s	51s
			4				90s	47s
			1			0,3	227s	68s
			4				109s	59s
			1			0,4	238s	77s
			4				104s	70s
			1			0,5	238s	87s
			4				111s	78s
			1			0,6	247s	93s
			4				121s	87s
			1			0,7	246s	100s
			4				102s	92s
			1			0,8	249s	133s
			4				127s	119s
			1			0,9	246s	129s
			4				110s	117s
			1			1	239s	125s
			4				115s	114s

REFERENCES

[1] P. Domingos and M. Richardson, “Mining the Network Value of Customers,” in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’01. New York, NY, USA: Association for Computing Machinery, 2001, pp. 57–66. [Online]. Available: https://doi.org/10.1145/502512.502525

[2] M. Richardson and P. Domingos, “Mining Knowledge-Sharing Sites for Viral Marketing,” in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’02. New York, NY, USA: Association for Computing Machinery, 2002, pp. 61–70. [Online]. Available: https://doi.org/10.1145/775047.775057

[3] D. Kempe, J. Kleinberg, and É. Tardos, “Maximizing the Spread of Influence through a Social Network,” in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’03. New York, NY, USA: Association for Computing Machinery, 2003, pp. 137–146. [Online]. Available: https://doi.org/10.1145/956750.956769

[4] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, “Cost-Effective Outbreak Detection in Networks,” in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’07. New York, NY, USA: Association for Computing Machinery, 2007, pp. 420–429. [Online]. Available: https://doi.org/10.1145/1281192.1281239

[5] A. Goyal, W. Lu, and L. V. Lakshmanan, “CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks,” in *Proceedings of the 20th International Conference Companion on World Wide Web*, ser. WWW ’11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 47–48. [Online]. Available: https://doi.org/10.1145/1963192.1963217

[6] K. Saito, R. Nakano, and M. Kimura, “Prediction of Information Diffusion Probabilities for Independent Cascade Model,” in *Knowledge-Based Intelligent Information and Engineering Systems*, I. Lovrek, R. J. Howlett, and L. C. Jain, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 67–75. [Online]. Available: https://doi.org/10.1007/978-3-540-85567-5_9

[7] J. Tang, J. Sun, C. Wang, and Z. Yang, “Social Influence Analysis in Large-Scale Networks,” in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 807–816. [Online]. Available: https://doi.org/10.1145/1557019.1557108

[8] A. Goyal, F. Bonchi, and L. V. Lakshmanan, “Learning Influence Probabilities in Social Networks,” in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, ser. WSDM ’10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 241–250. [Online]. Available: https://doi.org/10.1145/1718487.1718518

[9] N. Barbieri, F. Bonchi, and G. Manco, “Topic-Aware Social Influence Propagation Models,” in *2012 IEEE 12th International Conference on Data Mining*, 2012, pp. 81–90. [Online]. Available: https://doi.org/10.1109/ICDM.2012.122

[10] J. M. Kleinberg, “Authoritative Sources in a Hyperlinked Environment,” *J. ACM*, vol. 46, no. 5, pp. 604–632, Sept. 1999. [Online]. Available: https://doi.org/10.1145/324133.324140

[11] D. Easley and J. Kleinberg, *Link Analysis and Web Search*. Cambridge University Press, 2010, pp. 399–406. [Online]. Available: https://doi.org/10.1017/CBO9780511761942.015

[12] R. Albert and A.-L. Barabási, “Topology of Evolving Networks: Local Events and Universality,” *Phys. Rev. Lett.*, vol. 85, pp. 5234–5237, Dec. 2000. [Online]. Available: https://www.doi.org/10.1103/PhysRevLett.85.5234