Social Network Analysis and Mining (2019) 9:21
https://doi.org/10.1007/s13278-019-0559-9

REVIEW ARTICLE

Solving submodular text processing problems using influence graphs

Ali Vardasbi[1] · **Heshaam Faili**[1] · **Masoud Asadpour**[1]

Received: 2 June 2018 / Revised: 9 April 2019 / Accepted: 11 April 2019
© Springer-Verlag GmbH Austria, part of Springer Nature 2019

Abstract
Submodular functions appear in a considerable number of important natural language processing problems such as text summarization and dataset selection. Current graph-based approaches to solving such problems do not pay special attention to the submodularity and simplistically do not learn the graph model. Instead, they roughly set the edge weights in the graph proportional to the similarity of their two endpoints. We argue that such a shallow modeling needs to be replaced by a deeper approach which learns the graph edge weights. As such, we propose a new method for learning the graph model corresponding the submodular function that is going to be maximized. In a number of real-world networks, our method leads to a 50% error reduction compared to the previously used baseline methods. Furthermore, we apply our proposed method followed by an influence maximization algorithm to two NLP tasks: text summarization and *k*-means initialization for topic selection. Using these case studies, we experimentally show the significance of our learning method over the previous shallow methods.

Keywords Submodular function maximization · Influence spread · Extractive summarization · *K*-means clustering · Graph model

1 Introduction

Submodular functions are set functions with the submodular property. A set function $f : 2^V \rightarrow \mathbb{R}$ assigns to each subset S of the reference set V a real number $f(S)$. For example, V can be the set of possible positions for sensor installation. In this example, for every $S \subseteq V$ of possible positions, $f(S)$ gives the coverage area of the sensors placed at those positions. The submodular property is defined as follows:

$$\forall A, B \subseteq V \quad f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{1}$$

A considerable number of real-world problems have a submodular nature. Almost every problem in which the actions between members have a gain and a repeated action has no added value has submodular nature. Suppose two sets $A \subset B$ and a member $e \notin A$. If e is added to each of A or B, it introduces a marginal gain equal to her unrepeated actions relative to the initial set (A or B). Since $A \subset B$, the set of common actions between e and the members in A is included in B; hence, it cannot be greater than that of B. The marginal gain of e when added to each set equals her actions minus the aforementioned common actions. Consequently, the smaller common actions in A lead to a greater marginal gain; i.e.,

$$A \subset B \subseteq V \quad f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B) \tag{2}$$

The above inequality is known as the diminishing return property and is equivalent to the submodularity property stated at (1). This means that the added value (marginal gain) resulted from adding a new member e to a set, decreases when the set grows larger.

Most of the optimization problems concerning coverage are submodular. There are different types of coverage observed in several problems. For example, sensor placement concerns with the grid coverage: In which points of a grid should (a given number of sensors) be placed in order to cover as much grid points as possible? Or in the influence maximization problem we are interested in the graph coverage: which set of nodes (with a given cardinality) in a graph should be activated such that their cascade of influence through the graph covers the greatest possible number of nodes? Similarly, the summarization problem deals with the

✉ Ali Vardasbi
a.vardasbi@ut.ac.ir

Heshaam Faili
hfaili@ut.ac.ir

Masoud Asadpour
asadpour@ut.ac.ir

[1] School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran

Published online: 07 May 2019

content coverage: which members of a set should be picked so as to cover the largest possible fraction of its content?

In the field of natural language processing (NLP), submodular problems are quite popular. For example, in extractive text summarization, the objective is to find a set of sentences with the maximum concept coverage over the given text. In (Lin and Bilmes 2010), this problem is modeled by a penalized graph cut function. Assume the similarity between each pair (i, j) of textual units in a document is represented by w_{ij}. These similarities can be thought of the edge weights in the document's underlying graph $G(V, E)$. As such, a good summary is a collection of nodes $S \subset V$ with a high similarity to the rest of V (represented by graph cut function) and a low redundancy between themselves (represented by a penalty term). The objective function is expressed as $\sum_{i\in V\setminus S}\sum_{j\in S} w_{ij} - \lambda \sum_{i,j\in S: i\neq j} w_{ij}$. Since both the graph cut function and the penalty term are submodular, the resulting objective function is submodular as well.

Another example is the dataset selection. Assume that a big dataset is to be manually tagged for training purposes. Let S_V denote the system trained on the whole tagged dataset V and S_X be the system trained on a subset $X \subset V$. The size of X is determined by the available budget. Dataset selection is the task of finding a set X of a predefined size whose corresponding S_X has the best performance among all such subsets. Such a selection somehow tries to maximize the structural coverage of the dataset. In Mirzasoleiman et al. (2013), the dataset selection problem is solved via k-medoid clustering. Assume that each pair of elements (i, j) in V is associated with a dissimilarity measure d_{ij}. The loss function for k-medoid is defined by $L(S) = \frac{1}{|V|}\sum_{i\in V} \min_{j\in S} d_{ij}$. By introducing a suitable auxiliary element s_0, the objective function $f(S) = L(\{s_0\}) - L(S \cup \{s_0\})$ becomes submodular and maximizing f translates into minimizing the loss function L.

The interesting thing about the submodular functions is that their maximization admits a $\left(1 - \frac{1}{e}\right)$—approximation polynomial algorithm, namely the greedy solution (Nemhauser et al. 1978). However, the greedy solution has practical complications due to its execution time on big data. A considerable amount of research is devoted to the submodular maximization speedups (Badanidiyuru et al. 2014; Chekuri et al. 2015; Zhou et al. 2016).

Graph-based algorithms are another class of approaches for solving submodular text processing problems (Mei et al. 2010; Mihalcea and Tarau 2004; Tixier et al. 2017). Most of these graph-based algorithms sort the nodes based on a centrality measure and output the top nodes as the solution. Usually, centrality measures such as PageRank cannot model the redundancy incurred by the selection of similar nodes. By redundancy, we mean repeated actions of members which have no gain; i.e., $f(B \cup \{e\}) \leq f(B) + f(\{e\})$. Formally, the centrality measures can properly model the gain of members $f(\{e\})$ but are incapable to model the marginal gains $f(B \cup \{e\}) - f(B) f(B \cup \{e\}) - f(B)$. More concretely, they successfully compare members and announce that $f(\{e_1\}) > f(\{e_2\})$, but have nothing to say about $f(B \cup \{e_1\}) - f(B)$ and $f(B \cup \{e_2\}) - f(B)$. Therefore, the use of the centrality measures in submodular applications should be accompanied by a redundancy minimization measure. On the other hand, in influence maximization algorithms, the coverage is maximized by simultaneously aiming to increase the importance and decrease the redundancy. As a result, influence maximization algorithms are suitable solutions for submodular problems such as extractive summarization (Wang et al. 2013).

In this study, we propose a graph-based approach for maximizing submodular functions. Each solution for real-world submodular maximization problems contains two major phases: modeling the problem to a submodular function and then maximizing the derived function. It has to be noted that this work mainly deals with the first phase. For the second phase, the available influence maximization algorithms are used in this work and our contribution in the second phase is very narrow, i.e., suggesting that the influence maximization is suitable for solving the more general submodular maximization.

Unlike similar graph-based approaches in which the graph is deduced directly from the members' relations, we argue that *learning* the graph from such relations is necessary to obtain more accurate models with higher performances. We introduce a gradient descent method to learn the graph model from the member relations and design an experiment to intrinsically evaluate our method. Since our model learning method is to be used in the context of submodular maximization, we will show its goodness in two case studies as well: text summarization and k-means initialization for topic detection. In both of these cases, we first learn the graph representing the members using our method. Then, we perform a state-of-the-art influence maximization algorithm to select the nodes who, hopefully, maximize the corresponding submodular function. These case studies extrinsically verify that our method for learning the graph model positively impacts the performance of the current system.

2 Previous works

In this section, we briefly review some of the highly regarded graph-based research on NLP, mostly text summarization.

TextRank (Mihalcea 2004) uses graph-based ranking algorithms such as HITS (Kleinberg 1999) and positional power function (Herings et al. 2001) to rank and extract high ranked sentences. The graph representation of text is built by setting the edge weights equal to the sentences pair similarity. Likewise, LexRank (Erkan and Radev 2004) uses

a number of graph centrality measures such as degree centrality and PageRank (Brin and Page 1998) to detect central sentences. The edge weights of their graph model are equal to the cosine similarity between sentence pairs.

Leskovec et al. (2004) approached the extractive summarization task by generating an unweighted semantic graph of the document as well as their gold summaries and train a support vector machine on them. Working on unweighted graphs, (Matsuo et al. 2006) proposes a graph clustering algorithm for word clustering based on a word similarity measure by web counts. They perform the Newman clustering on a graph that is built as follows. Each word is represented by a node and if the pointwise mutual information or the Chi-square value between a pair of words is above a certain threshold, an edge is drawn between the two nodes.

Agirre et al. (2006) proposed a graph-based unsupervised induction and tagging of nominal word senses. They select the senses using two different hub ranking strategies inspired by HyperLex (Véronis 2004) and PageRank on the co-occurrence graph. In their graph model, each edge is assigned a weight which measures the relative frequency of the two words co-occurring: $w_{ij} = 1 - \max\left[\frac{\text{freq}_{ij}}{\text{freq}_i}, \frac{\text{freq}_{ij}}{\text{freq}_j}\right]$.

In Alexandrescu and Kirchhoff (2007), a graph construction method has been proposed for the label propagation (LP) algorithm. They introduce a data-driven method that uses a supervised classifier to optimize the representation of the initial feature space. As such, their contribution lies on recomputing the distance between pairs. After that, the weights are simply set to $\exp\left(-\frac{\text{distance}_{ij}^2}{\alpha^2}\right)$, similar to the LP algorithm.

DivRank is a vertex reinforced random walk method that, unlike PageRank, has time-varying probabilities for random steps (Mei et al. 2010). Changing the step probabilities enables DivRank to highlight the diversity among the hubs. The edges are unit-weighted and added for each sentence pair with a cosine similarity higher than 0.1.

In Lin and Bilmes (2010), a graph-based extractive summarization is proposed. The edge weights in their graph are equal to the cosine similarities between sentence pairs, and the objective function is a modified graph cut function. Roughly, their submodular objective function is the sum of all edge weights for pairs with one end on the summary set, and the other on the rest of the graph subtracted by the sum of all edge weights for pairs with both ends within the summary set.

In Galluccio et al. (2012), the minimal spanning tree (MST) as grown by Prim's algorithm is used to detect initial centroids for k-means clustering algorithm. They also propose a data-driven hierarchical classification algorithm to avoid the time-consuming computation of huge similarity matrices. Needless to say that the edge weights in their model are set to the distance between pairs of data points.

In (Baralis et al. 2013), the graph nodes represent sets of words and the edge weights indicate the correlation between node pairs. The correlations among word sets are extracted using association rule mining. They adopt a variant of PageRank to select central word sets and extract the sentences that best cover the central words.

Berton et al. (2015) proposed to use link prediction measures for graph construction. They set the edge weights to similarity between node pairs and focus on the problem of which edges should be retained and which should not. Consequently, their approach for graph construction goes one step deeper than the ones that use k nearest neighbors or retain the edges stronger than a predefined threshold.

In Yasunaga et al. (2017), a graph-based summarization is proposed that applies graph convolutional networks on sentence relation graphs. The relation graphs are built by setting the edges equal to the sentence pair similarities. They propose three different methods to construct such a graph. The first model uses the cosine similarity between sentence pairs. The second model is the approximate discourse graph where edges are built by counting discourse relation indicators between sentences to mimic the characterization of sentence relationships. Their third and most advanced model diversifies the edges of the second model by applying a sentence personalized multiplicative effect to the edges.

There are plenty of works on submodular maximization in the literature (Mirzasoleiman et al. 2016, 2018; Kazemi et al. 2018). These works should be used after a problem is modeled into a submodular function (graph or non-graph). Comparing the results obtained by replacing our suggestion for submodular maximization (i.e., influence maximization) with these studies is out of the scope of this work.

As was mentioned earlier, and will be discussed further in the next sections, the previous graph-based methods directly use similarities (distances or other relations) as the edge weights. We, to the contrary, learn the edge weights from the relations.

3 Graph model

The first step in every graph-based algorithm is to build a justifiable model for the graph using the inputs of the problem. In addition to the input and output of the problem, the graph model depends also on the algorithm which is going to be applied to the graph. In this study, an influence maximization approach is proposed to solve submodular text processing problems. Therefore, in this section, we are interested in building an influence spread network from a submodular function. Such a network is shortly referred to as *Influence Graph*. In this context, *influence* models

coverage. To state it more precisely, the probability of one node u to influence another node v in the influence graph is desired to suitably model the fraction of v's actions covered by u (directional graph), or in some cases the similarity between v and u (unidirectional graph).

Reducing a submodular maximization problem into an influence maximization one has the following benefits:

- State-of-the-art algorithms for the influence maximization problem are scalable. This means that they can find the solution in a practical time even if the size of the problem is huge and their high performance will not be flawed. As such, our approach gives a scalable framework for maximizing the submodular functions in the context of text processing problems.
- A considerable number of text processing problems are based on the similarity measurement between textual units such as sentences and documents. The length of the two compared units has a non-negligible impact on the measurement performance and feeding two fairly equal length units to the similarity measurement system is usually desirable. However, the comparison between one unit and a set of previously selected units is required in the submodular maximizers. This means that the inputs of the similarity function have two intensely different lengths. Using the influence graph instead, the similarity between a unit and a set of units is approximated from the graph structure. The graph structure itself is obtained from the similarity between pairs of units and is reliable due to the fairly equal lengths of the units. This approach gives a general framework for such approximations and replaces the problem-specific heuristic methods.

Definition 1 formally defines this problem.

Definition 1 Given a set $S = \{x_1, \dots, x_n\}$ and a similarity function $\delta : S^2 \to [0, 1]$ (as input) build a weighted graph $G(S, E)$ with weight function $\omega : E \to [0, 1]$ (as the output) such that the expected probability of a node x_i influencing x_j be equal to $\delta(x_i, x_j)$.

Most of the present algorithms treat this step with the simplest approach: they set the edge weights proportional to the similarity. This simple approach, while having nearly zero execution time, has shown acceptable results in the literature (Pilehvar and Navigli 2015; Erkan and Radev 2004; Beliga et al. 2015). Throughout the rest of this paper, this approach is called *Baseline-0* (the numbers at the end indicate the level of depth). We show in this study that using deeper methods helps to improve the performance of the optimization problem.

3.1 Proposed methods

We propose three methods for learning the influence graph model. The first method (*Proportional Weights*, or shortly *PW-1*) heuristically tries to learn the weight function through a simplified local viewpoint: one edge at a time. The aim in PW-1 is to propose a simple and fast algorithm which is one level deeper than Baseline-0. In the second method (*Minimized Mean Squared Error*, or shortly *MMSE-2*), the local viewpoint of PW-1 goes one level deeper and instead of correcting one edge at a time, all the edges sharing one node as their endpoint are corrected simultaneously. *PW-1* and *MMSE-2* both have practical running times and are implemented in this study. The third method (*Global Matrix*, or shortly *GM-3*), on the other hand, has a global viewpoint and considers the full network at each iteration. This globalization comes at the cost of a $O(n^3)$ running time that makes GM-3 impractical for moderate input sizes. GM-3 is only interesting theoretically and we state it in this study to complete the route from shallow (Baseline-0) to deep (GM-3) approaches.

Before stating the methods, required assumptions and definitions are presented. Based on Definition 1, the similarity between the ith and jth elements is computed by $\delta(x_i, x_j)$. For simplicity, this quantity is shown by δ_{ij}. As such, function $\delta : S^2 \to [0, 1]$ and matrix $[\delta_{ij}]_{n\times n}$ are used interchangeably in the coming sections. Similarly, the weight function $\omega : E \to [0, 1]$ in Definition 1 is represented by $[\omega_{ij}]_{n\times n}$ in which ω_{ij} is the weight of the link connecting node x_i to x_j. Ideally, the problem seeks a suitable $[\omega_{ij}]_{n\times n}$ to achieve

$$\forall 0 \le i, j \le n : p(x_i \to x_j) = \delta_{ij} \tag{3}$$

The influence of x_i is accumulated through influence paths of different length to activate x_j. If the expected influence probability is only approximated with paths of length at most two, the following closed form is easily obtained

$$p_2(x_i \to x_j) = 1 - \prod_{\substack{x_k \in S \\ k \neq i}} (1 - \omega_{ik}\omega_{kj}) \tag{4}$$

where the subscript in $p_2(x_i \to x_j)$ represents the length two approximation. Furthermore, it is reasonably assumed that $\omega_{jj} = 1$. Equation (4) is obtained using the independence of influence probabilities. The probability of x_j being influenced by x_i through x_k is simply $\omega_{ik}\omega_{kj}$; i.e., the probability that the $x_i x_k x_j$ path is connected. Therefore, by the independent influence paths assumption, the probability that none of the paths from x_i to x_j are connected equals $\prod_{\substack{x_k \in S \\ k \neq i}} (1 - \omega_{ik}\omega_{kj})$. To compute Eq. (4), it is required to iterate only through the common neighbors of x_i and x_j. Utilizing V and E as the size of nodes and edges of the underlying

graph, the average degree order can be expressed by $O\left(\frac{E}{V}\right)$. Therefore, the average complexity of Eq. (4) for all edges becomes $O\left(\frac{E^2}{V}\right)$. Since real-world networks (as well as the ones constructed in NLP tasks) are scale free in which the average degree is of order $O(\log V)$, the average complexity of Eq. (4) is practically $O\left(V \log^2 V\right)$.

In the consequent methods, we use an approximation $p_2\left(x_i \to x_j\right)$ instead of the exact $p\left(x_i \to x_j\right)$ in order to make the methods running time practical.

3.1.1 PW-1: proportional weights method

As was mentioned in Sect. 3 the simplest way for modeling the influence graph is to set the edge weights proportional to the similarities (we called it the Baseline-0); i.e.,

$$\omega_{ij} = \alpha \delta_{ij} \tag{5}$$

PW-1 is based on finding the best fitting coefficients $\left[\alpha_{ij}\right]_{n\times n}$ for the following relations:

$$\omega_{ij} = \alpha_{ij} \delta_{ij} \tag{6}$$

and iteratively adjusts the coefficients $\left[\alpha_{ij}\right]_{n\times n}$. The intuition of this method is to improve the Baseline-0 model by increasing its degrees of freedom from 1 to E; i.e., the size of the edge set. This is rational, since Baseline-0 is widely used in the literature and its results are among state-of-the-art systems. Therefore, improving such a successful model by introducing new parameters constitutes our first attempt at graph construction.

At each iteration, $p_2\left(x_i \to x_j\right)$ is computed and compared to its goal value δ_{ij}. We utilize a multiplicative correction method for the weights. Since the weights are going to be set $\omega_{ij} = \alpha_{ij}\delta_{ij}$, in order to make $p_2\left(x_i \to x_j\right)$ tend to δ_{ij}, the α_{ij} should be corrected with respect to $\frac{\delta_{ij}}{p_2\left(x_i \to x_j\right)}$.

PW-1 is constructed as follows:

- First, initialize all the coefficients with $\alpha_{ij}^0 = \frac{1}{2}$ (c.f. (6))
- While the termination condition is not satisfied, do the following starting from $t = 1$:
    - For all $\alpha_{ij}^{t-1} > 0$:
        - Set $\alpha_{ij}^t = \alpha_{ij}^{t-1} \times \text{smooth}\left(\frac{\delta_{ij}}{p_2\left(x_i \to x_j\right)}\right)$
    - If $\alpha_{ij}^t < \theta$, set $\alpha_{ij}^t = 0$. (The threshold θ should be selected based on the time constraint)
        - Increment t.

Theoretically, the smooth(x) function can be any increasing function. We propose to use the following function:

$$\text{smooth}(x) \triangleq 1 + \text{sign}(x-1) \cdot |x-1|^r \tag{7}$$

where r can be any real number and the sign$(\cdot)$ function returns the sign of its input. We have tested the smooth(x) for several values of r. The results of $1 \le r \le 3$ have no significant difference. Therefore, we choose the simplest form for the function as follows:

$$r = 1 \Rightarrow \text{smooth}(x) \triangleq x \tag{8}$$

The time complexity of this method is equal to the computation time for $p_2\left(x_i \to x_j\right)$ values at each iteration. Letting T to be the number of iterations before a convergence, the complexity of this method equals $O\left(T \cdot V \cdot \log^2 V\right)$. Concerning the space complexity, this method only introduces the α_{ij} parameters for each edge. Therefore, the space complexity of this method equals $O(E)$ or for the scale-free networks $O(V \cdot \log V)$.

This method only considers the influence of one node on its neighbors to model the edge weights. But it has more depth compared to Baseline-0 where a constant coefficient $\alpha_{ij} = \alpha$ is used for all the edges. Our experiments show that this method has an acceptable performance in modeling the influence graph.

The next two methods use the gradient descent optimization and have a more solid theoretical foundation.

3.1.2 MMSE-2: minimized mean squared error method

The influence graph model should ideally satisfy Eq. (3). This condition leads to the following loss function (mean squared error):

$$\mathcal{L}_i = \frac{1}{n} \sum_{(x_i, x_j) \in E} \left(p\left(x_i \to x_j\right) - \delta_{ij}\right)^2 \tag{9}$$

The $\mathcal{L}_i$ function denotes the loss corresponding to x_i.

Instead of introducing a sparsity regularization term in the loss function, we set a threshold parameter θ to be used for pruning graph edges before the learning process begins; a simple and popular approach exploited by many previous works (Erkan and Radev 2004). As such, only the edges whose corresponding δ_{ij} is greater than θ are retained in the edge set E. The value for the threshold θ is a tradeoff between the time complexity and precision: a higher θ leaves less edges inside E and the learning algorithm will run faster with the cost of a decreased precision.

Computing the exact value of $p\left(x_i \to x_j\right)$ requires time-consuming Monte-Carlo simulations. Consequently, the exact gradient computation is only possible via numerical methods. In order to make the method practical, the exact influence probability can be replaced by its approximation $p_2\left(x_i \to x_j\right)$. Due to its closed form, the gradient is easily computed by:

$$\frac{\partial p_2(x_i \to x_j)}{\partial \omega_{ik}} = \left(1 - p_2(x_i \to x_j)\right) \cdot \frac{\omega_{kj}}{1 - \omega_{ik}\omega_{kj}} \tag{10}$$

$$\frac{\partial \mathcal{L}_i}{\partial \omega_{ik}} = \frac{1}{n} \sum_{(x_i, x_j) \in E} \frac{2\omega_{kj}}{1 - \omega_{ik}\omega_{kj}} \cdot \left(1 - p_2(x_i \to x_j)\right) \cdot \left(p_2(x_i \to x_j) - \delta_{ij}\right) \tag{11}$$

Finally, by choosing a learning step ξ, the iterative learning algorithm updates the weights as follows:

$$\omega_{ik} \leftarrow \omega_{ik} + \xi \cdot \frac{\partial \mathcal{L}_i}{\partial \omega_{ik}} \tag{12}$$

Similar to PW-1, the time complexity of this method is $O(T \cdot V \cdot \log^2 V)$ for computing $p_2(x_i \to x_j)$ for each edge. This method requires no extra memory.

This method is more straightforward compared to PW-1. It directly goes for the model's main objective; i.e., $p_2(x_i \to x_j) = \delta_{ij}$. To do so, we follow a famous and widely used optimization technique which is to minimize the mean squared error using gradient descent algorithm. The optimization is performed on the edge weights and ω_{ik} is only present in $p_2(x_i \to x_j)$ for all x_i's neighbors. Hence, the only terms with nonzero derivative with respect to ω_{ik} are $p_2(x_i \to x_j)$, the probabilities along the outgoing links of x_i. That is why we define the loss function $\mathcal{L}_i$ to be the mean loss along the outgoing links of x_i and use its gradient to update the outgoing links of x_i.

MMSE-2 has a local view in learning the weights, but its view is wider than PW-1. GM-3, presented in the consequent section, learns the weights globally.

3.1.3 GM-3: global matrix method

In GM-3, unlike the previous two methods, the learning problem is solved globally. But similar to MMSE-2, the influence probability function is replaced with its approximation. The problem is simplified to finding $[\omega_{ij}]_{n \times n}$ such that:

$$\delta_{ij} = 1 - \prod_{\substack{1 \le k \le n \\ k \ne i}} \left(1 - \omega_{ik}\omega_{kj}\right) \tag{13}$$

Since both $(1 - \omega_{ii}\omega_{ij})$ and $(1 - \omega_{ij}\omega_{jj})$ refer to the direct link $(1 - \omega_{ij})$, Eq. (13) can be rewritten to:

$$\delta_{ij} = 1 - \frac{\prod_{1 \le k \le n} \left(1 - \omega_{ik}\omega_{kj}\right)}{1 - \omega_{ij}} \tag{14}$$

Taking logarithm from both sides, Eq. (14) turns into:

$$\begin{aligned} -\log\left(1 - \delta_{ij}\right) &= \sum_k -\log\left(1 - \omega_{ik}\omega_{kj}\right) + \log\left(1 - \omega_{ij}\right) \\ &\approx \sum_k \left(\omega_{ik}\omega_{kj} + \frac{\omega_{ik}^2\omega_{kj}^2}{2} + \cdots\right) \\ &\quad - \left(\omega_{ij} + \frac{\omega_{ij}^2}{2} + \cdots\right) \end{aligned} \tag{15}$$

Where the approximation $\log(1 + x) \approx x + \frac{x^2}{2} + \frac{x^3}{3} + \cdots$. is used in the last equation. The matrix form of (15) is as follows:

$$\begin{aligned} \left[-\log\left(1 - \delta_{ij}\right)\right] &\approx \left[\omega_{ij}\right]^2 + \frac{\left[\omega_{ij}^2\right]^2}{2} + \frac{\left[\omega_{ij}^3\right]^2}{3} + \cdots - \left[\omega_{ij}\right] \\ &\quad - \frac{\left[\omega_{ij}^2\right]}{2} - \frac{\left[\omega_{ij}^3\right]}{3} \end{aligned} \tag{16}$$

Finally, the loss function together with the regularization term is defined as follows:

$$\mathcal{L} = \mathrm{tr}\left(BB^{\mathrm{T}}\right) + \lambda \left| \left[\omega_{ij}\right] \right| \tag{17}$$

where $\mathrm{tr}(X)$ is the trace of the square matrix X, λ is the regularization impact and B. is the error of Eq. (16) which is going to be minimized:

$$B = \left[-\log\left(1 - \delta_{ij}\right)\right] - \sum_{k=1} \frac{1}{k} \left(\left[\omega_{ij}^k\right]^2 - \left[\omega_{ij}^k\right] \right). \tag{18}$$

Given the differentiable loss function, the gradient descent method can be used to obtain the $[\omega_{ij}]$. matrix. But, in spite of its good theoretical form, GM-3 has $O(n^3)$ complexity which makes it impractical.

4 Model evaluation

As discussed earlier, our approach for solving submodular problems has two phases:

1. Building the graph model out of the data point relations in a submodular function;
2. Performing influence maximization over the built graph.

While a complete evaluation of our approach requires both of these phases to be applied on a submodular problem, a model evaluation only on the first phase is useful to intrinsically show its performance. It has to be noted that this section presents the model evaluation (only the first phase) and, as a result, no submodular maximization is performed in this section. The complete evaluation containing both the

model and the submodular maximization will be presented in Sect. 6.

The only way to intrinsically evaluate the model is to use the influence spread function itself as the input submodular function. Remember that our model's output is the edge weights of the constructed graph. Consequently, in order to intrinsically evaluate the first phase, a problem with known underlying graph should be worked on, so that the edge weights of the known underlying graph can be used as gold data. This leads us to the influence spread function for which the underlying graph is readily known.

We first compute the $[\delta_{ij}]$ matrix corresponding to a known graph (with known weights $\left[\omega_{ij}^{\text{gold}}\right]$). Then we use PW-1 and MMSE-2 to learn the $\left[\omega_{ij}^{\text{learned}}\right]$ matrix from the computed matrix. Finally, the evaluation is performed by comparing the gold weights $\left[\omega_{ij}^{\text{gold}}\right]$. and the learned weights $\left[\omega_{ij}^{\text{learned}}\right]$.

4.1 Experimental setup

The intrinsic evaluation is conducted on the following real-world networks obtained from (Leskovec 2018):

- *CA-GrQc, CA-HepTh, CA-CondMat* These graphs are the collaboration network from the e-print arXiv and cover scientific collaborations between authors of papers submitted to General Relativity and Quantum Cosmology category, High Energy Physics Theory and condensed matter, respectively (Leskovec et al. 2007).
- *Email-Enron* This dataset contains the email communications of Enron. The nodes represent the Enron email addresses, and an undirected link between i and j indicates that either of them has sent an email to the other (Klimt and Yang 2004, Leskovec et al. 2009).
- *DBLP* The DBLP computer science bibliography provides a comprehensive list of research papers in computer science. This graph is a co-authorship network where two authors are connected if they publish at least one paper together (Yang and Leskovec 2015).

Table 1 Network statistics

Network	#nodes	#edges
CA-GrQc	5242	28,980
CA-HepTh	9877	51,971
CA-CondMat	23,133	93,497
Email-Enron	36,692	367,662
DBLP	317,080	2,099,732

The network statistics are shown in Table 1. For the edge weights, we use the following two models. These models are somehow a generalization of popular weight models used in the influence maximization literature:

- UN: The uniform model that chooses the weights from interval $[0.001, 0.1]$ uniformly at random; and
- TMv: the Transitive Multi-valency Model that will be introduced consequently.

Triadic closure (Granovetter 1973) and clustering coefficient (Watts and Strogatz 1998) in the social network theory are two strongly related concepts that demonstrate the transitive behavior in social networks. In the context of influence spread, the transitivity of nodes influence on their neighbors can be stated as follows:

Influence Transitivity The influence of a node u on a neighbor v is dependent upon the portion of nodes directly influencing v who are themselves directly influenced by u.

In other words, let the set of nodes directly influencing v (also known as its in-neighbors) be denoted as N_v^- and the set of nodes directly influenced by u (also known as its out-neighbors) be denoted as N_u^+. The influence transitivity states that

$$e_{uv} = f\left(\frac{\left|N_v^- \cap N_u^+\right|}{\left|N_v^-\right|}\right) \tag{19}$$

where e_{uv} is the weight of the edge connecting u to v.

In this study, we use our new edge weight model based on the influence transitivity which sets the edge weights as follows

$$e_{uv} = \left(\frac{\left|N_v^- \cap N_u^+\right|}{\left|N_v^-\right|}\right)^{1.5} \cdot R_{uv} \tag{20}$$

where R_{uv} is a random multi-valency attenuator chosen uniformly at random from the set $\{3^{-2}, 3^{-3}, 3^{-4}, 3^{-5}\}$. This attenuator together with the 1.5 exponent is included to avoid the influence saturation due to the large edge weights. This choice is inspired by the famous Trivalency model (Chen et al. 2010) where the weights are randomly selected from $\{10^{-1}, 10^{-2}, 10^{-3}\}$. We chose base 3 instead of 10 to reach a more balanced and realistic model.

In our experiment results, this model is represented by TMv.

4.2 Evaluation

In this section, the performance of PW-1 and MMSE-2 on modeling the influence graph is evaluated. The requirement of having the gold graph to be compared to the learned model is fulfilled by considering the influence spread function in an influence graph as the input submodular function. PW-1 and MMSE-2 try to learn the edge weights given only the influence spread function. Since the influence spread function is the only submodular function which is actually obtained from an influence graph, this is the only possible intrinsic evaluation method for our proposal. Other extrinsic evaluations will be presented in the next sections.

The evaluation is performed during the following steps:

- Consider a graph G (in our experiments we use two graphs in Table 1).
- Choose a weight model and assign weights to the edges of G based on the model (in our experiments we use two models UN and TMv). These weights are the gold weights $\left[\omega_{ij}^{\text{gold}}\right]$ and are kept secret from the next two steps (learning process).
- In the resulting weighted influence graph, do the following for all pair of nodes (i, j):
  - Run the Monte-Carlo simulations to compute the influence probability of node j being influenced by activating node i and name it δ_{ij}.
- Use PW-1 and MMSE-2 to learn the weights of the edges and name them $\left[\omega_{ij}^{\text{learned}}\right]$.

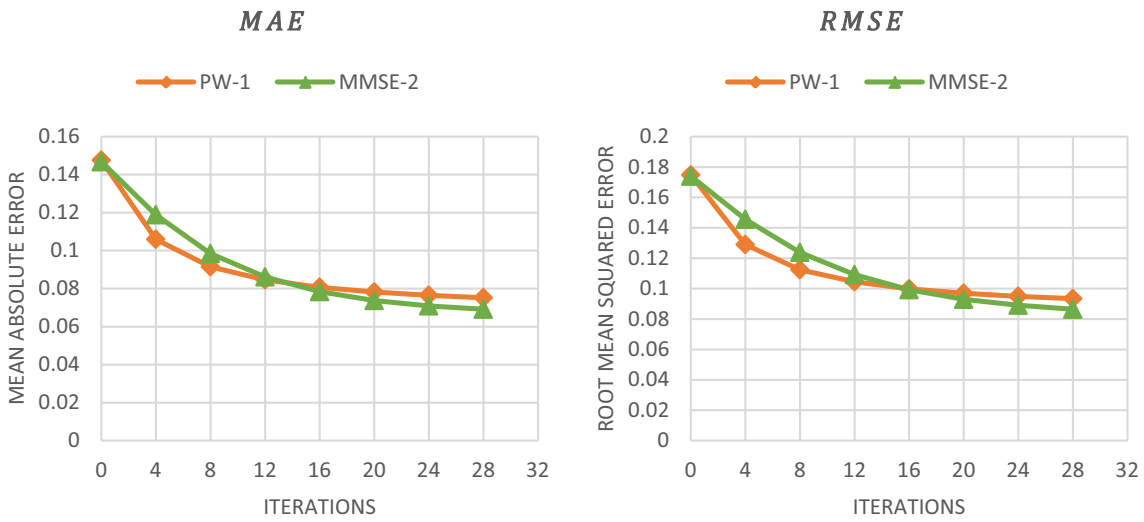

Fig. 1 Error measures of CA-GrQc graph with model TMv

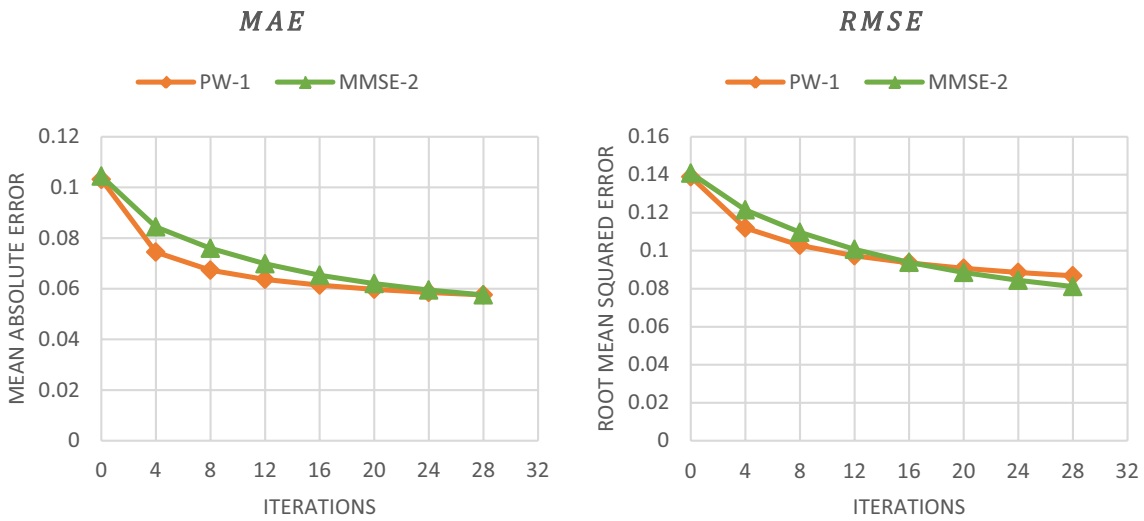

Fig. 2 Error measures of CA-GrQc graph with model UN

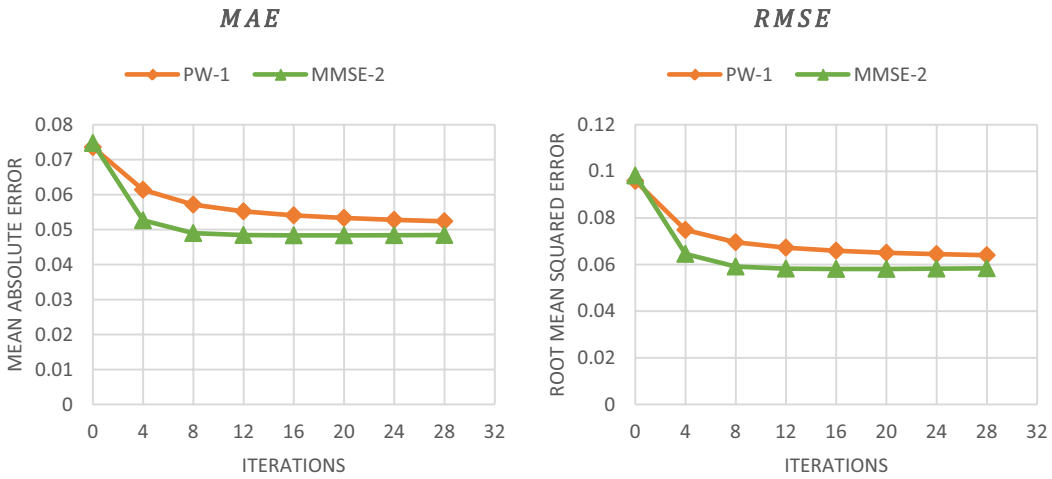

Fig. 3 Error measures of CA-HepTh graph with model TMv

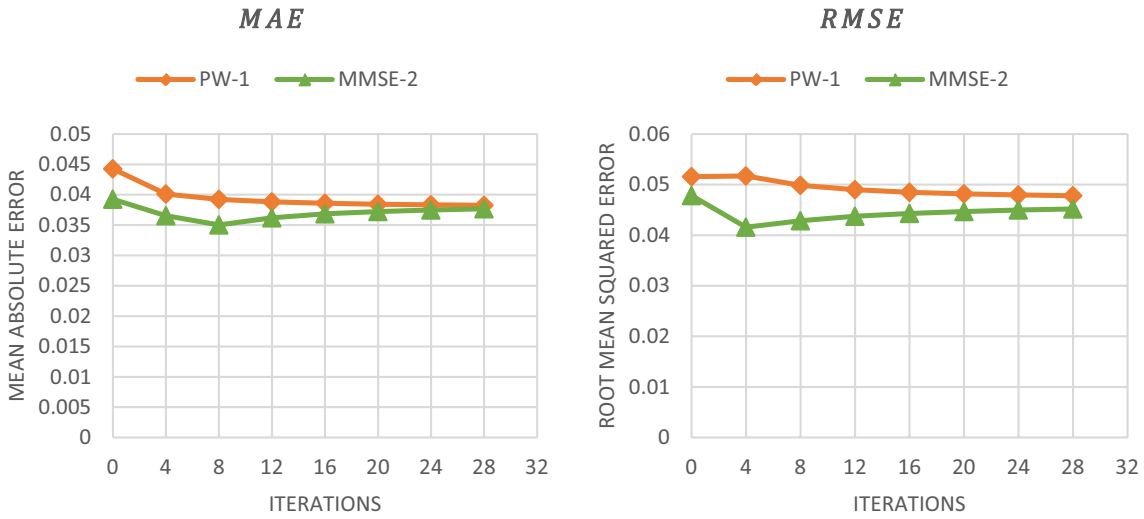

Fig. 4 Error measures of CA-HepTh graph with model UN

- Report the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) of $\left[\omega_{ij}^{\text{learned}}\right]$ compared to $\left[\omega_{ij}^{\text{gold}}\right]$ in terms of the iterations of learning method.

Figures 1, 2, 3 and 4 demonstrate the MAE and RMSE measures of two graphs CA-GrQc and CA-HepTh with models TMv and UN. Each figure contains two plots corresponding to PW-1 and MMSE-2. These plots show that MMSE-2 slightly performs better than PW-1, but the difference is not significant in our test cases. In both methods, the weights are initialized using Baseline-0 before the first iterations. This means that the first point in each plot (iteration 0) demonstrates the result of Baseline-0.

Table 2 Average running time of one iteration of PW-1 and MMSE-2 (in s) on small graphs

Graph	Model	PW-1	MMSE-2
CA-GrQc	TMv	0.04	0.11
	UN	0.06	0.13
CA-HepTh	TMv	0.03	0.03
	UN	0.03	0.06

In CA-GrQc graph, our methods reduce the errors to nearly half of its initial value in both TMv and UN models after only 28 iterations (Figs. 1, 2).

The impact of our methods on reducing the error is lower in the CA-HepTh graph, especially with the UN model (Figs. 3, 4). In the UN model, the error measures are nearly

flat during the iterations of learning methods. The main reason for such behavior is that the initial weights (proportional to the similarity measure) have already a very low error and the iterations of the methods cannot reduce it further.

Table 2 contains the running times of one iteration of PW-1 and MMSE-2 on small graphs CA-GrQc and CA-HepTh with models TMv and UN. The entries indicate that our methods are quite practical with running times of milliseconds. As was expected by design, MMSE-2 has a higher running time than PW-1.

The main issue with evaluating our model on big graphs lies on generating the gold data for our model to be tested upon. In fact, our method can be applied on big graphs to learn the edge weights, but the results cannot be compared against the gold data, i.e., the edge weights for the entire graph; since it is infeasible to compute the influence probabilities along all the edges for moderate-sized graphs. To overcome this obstacle, we generate the gold data only for a subset of edges, train our model on the entire graph, and evaluate our model using the precomputed subset of gold edges. Since the gold subset is selected randomly, we repeat this experiment several times and report the average and standard deviation of MAE and RMSE. It has to be noted about our evaluation that all of the edge weights are learned and the reported running time is a measurement of the time complexity of the complete algorithm.

We test our method on three larger graphs: CA-CondMat, Email-Enron and DBLP with around 23 k, 36 k and 317 k nodes as well as 93 k, 360 k and 2 M edges, respectively. The golden set consists of outlinks of 10 k, 20 k and 80 k randomly selected nodes, respectively. Random selection is performed 100 times and the average and standard deviation of the errors (in the form of MAE and RMSE) is reported. Figures 5, 6 and 7 demonstrate the error measures of MMSE-2 method on learning the edge weights of large graphs. Each plot contains three levels:

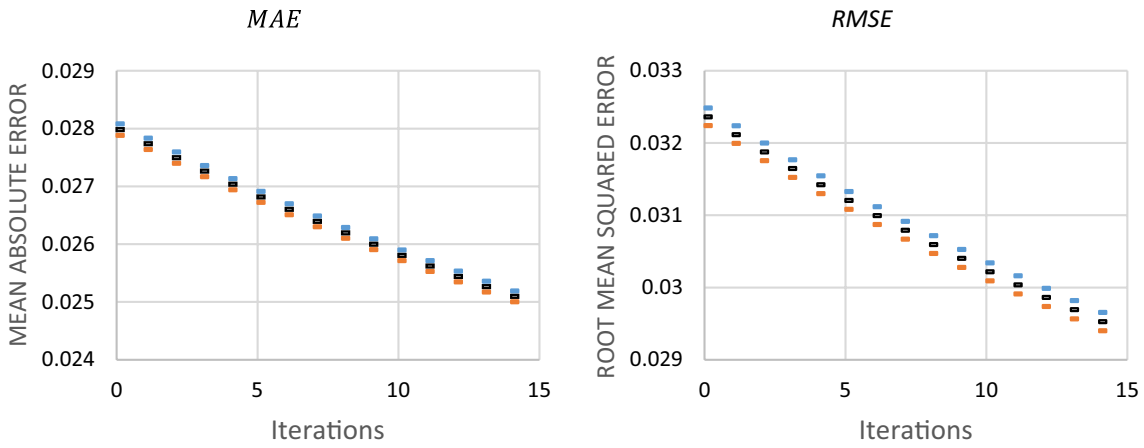

Fig. 5 MMSE-2 error measures of CA-CondMat graph with model TMv

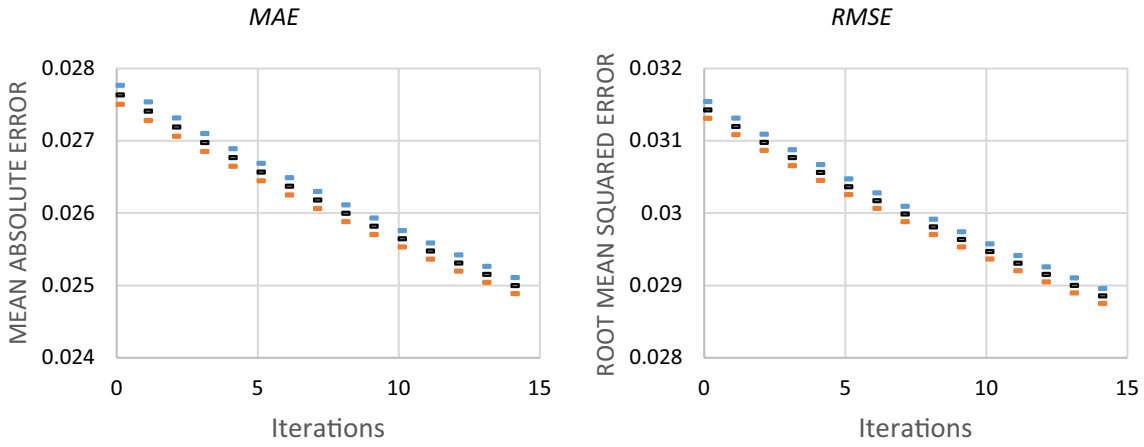

Fig. 6 MMSE-2 error measures of Email-Enron graph with model TMv

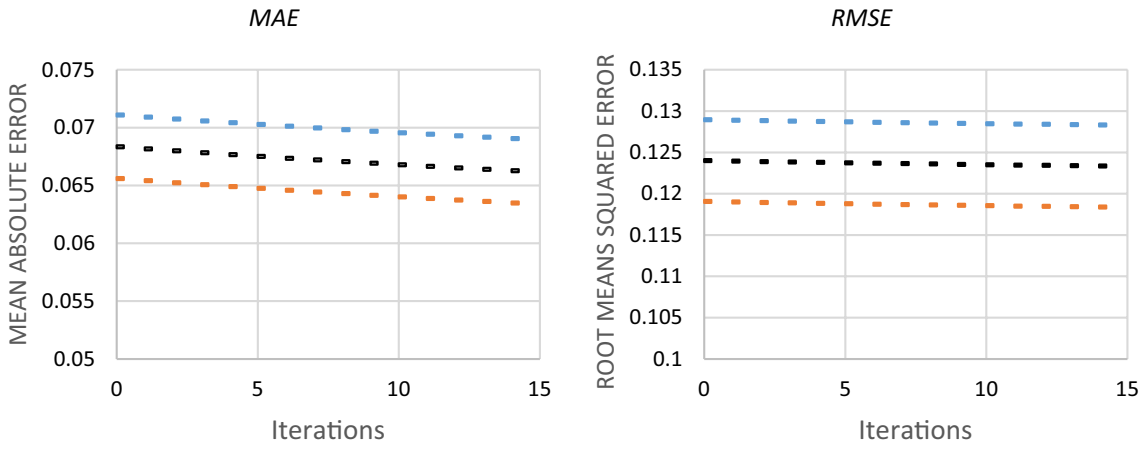

Fig. 7 MMSE-2 error measures of DBLP graph with model TMv

Table 3 Average running time of one iteration of MMSE-2 (in s) on large graphs

Network	Time (s)
CA-CondMat	0.09
Email-Enron	1.53
DBLP	14.05

the average at the center with two levels above and below the average with a distance equal to the standard deviation. The relatively larger standard deviation for the DBLP graph (Fig. 7) is due to the larger gap between its number of nodes (317 k) and the size of randomly selected golden sets (80 k nodes). These plots show that our method (MMSE-2) is successful to learn the edge weights of large graphs as well. Table 3 contains the running time of one iteration of MMSE-2 method performed on large graphs. This table shows that our method can feasibly be applied on large graphs like DBLP with more than 2 M edges.

5 NLP applications

In this section, two NLP submodular problems are discussed: text summarization and *k*-means initialization for topic detection. Our proposed method for solving submodular problems, which consists of first modeling the influence graph and then performing an influence maximization algorithm on the learned graph, is applied to both of the tasks. We will show by experiments that in both of the tasks our method outperforms the available systems as well as the simple Baseline-0 which is currently regarded as a strong competitor in the literature.

5.1 Text summarization

It is shown in Lin and Bilmes (2011) that most of the popular methods in extractive summarization and even the summary evaluation measures are submodular functions. In fact, the information coverage function over text has the diminishing return property. Suppose $A \subseteq B$ to be two sentence sets that are selected as the summary of document D. Furthermore, suppose that the f_{cover} function computes the amount of information covered from D by a set of sentences. Obviously, the common information between a sentence $x \notin B$ and the set B is not less than the common information between x and A. This means that the marginal benefit of adding x to B is not greater than that of adding x to A:

$$\forall A \subseteq B, x \notin B \quad f_{\text{cover}}(x|A) \geq f_{\text{cover}}(x|B) \tag{21}$$

The extractive summarization task can be stated in the context of Definition 1 as follows. The input document S is composed of n sentences $\{x_1, \ldots, x_n\}$. The similarity function $\delta : S^2 \rightarrow [0, 1]$ can be any function that measures the similarity (lexical or semantic) between two sentences. In this paper, we use a TF-IDF based score function similar to the well-known Okapi BM25. Ultimately, the summarization can be thought of as a two phases task:

1. Learn the influence graph model from the similarity matrix $[\delta_{ij}]$ (using one of the PW-1 or MMSE-2);
2. Use an influence maximization algorithm such as IMM (Tang et al. 2014) or SWIM (Vardasbi et al. 2017) to select and report the most influential sentences as the summary.

5.2 *K*-means initialization for topic detection

Topic detection refers to the task of document clustering based on their topic. Naturally, a considerable amount of solutions use known clustering algorithms for topic detection (Yang et al. 1998; Weng et al. 2011; Spina et al. 2014).

K-means and its variants constitute the largest share of clustering techniques in the literature. It is known that the

K-means algorithm is highly sensitive to the initial centers of clusters (Celebi et al. 2013). The cluster centers in a good clustering should have the following properties:

- The center should be in close relation to other cluster members; and
- The centers of different clusters should have a meaningful distance from each other.

Noting the above two properties; we model the problem of locating the cluster centers as a coverage problem. We assume that each point can be *covered* by its nearby points based on their pairwise similarity. Since the coverage problem has a submodular nature, our proposed method can be used to solve it.

Similar to other clustering algorithms performed on the topic detection task, a similarity function should be defined to measure the distance/similarity between pairs of documents. We chose the cosine similarity between the vector representations of documents obtained from the Latent Dirichlet Allocation (LDA) method. Using LDA in the context of topic detection has shown very impressive results in the literature (Weng et al. 2011; Huang et al. 2012; Rosen-Zvi et al. 2004). To be concrete, we run the LDA on the set of documents that are going to be clustered. The LDA assigns a real vector to each document. The similarity δ_{ij} between the ith and j th document is defined as the cosine similarity between their corresponding vectors.

Locating the centers for topic clusters is performed during a two phases task:

1. Learn the influence graph model from the similarity matrix $[\delta_{ij}]$ (using one of the PW-1 or MMSE-2);
2. Use an influence maximization algorithm such as IMM (Tang et al. 2014) or SWIM (Vardasbi et al. 2017) to select and report the most influential documents as the initial cluster centers.

6 Experiments

The intrinsic evaluation alone does not show the significance of our method. In order to show its practical significance, we use two case studies as extrinsic evaluation measures. Namely, we use the extractive summarization and the initial cluster center selection for the topic detection task as our case studies. Our experiments show that in both of these tasks, PW-1 and MMSE-2 outperform the simple and widely used Baseline-0.

6.1 Experimental setup

Extractive summarization is performed on dataset DUC-04, similar to most of the summarization tasks (Mei et al. 2010; Lin and Bilmes 2011, 2012). DUC (Document Understanding Conferences) datasets are generated by the USA National Institute of Standards and Technology (NIST). DUC-04 dataset contains 50 subtasks with ten documents. Each subtask is accompanied with four gold multi-document summaries produced by human experts. The reported results in Sect. 6.2 are averaged through the subtasks.

For the topic detection task, we have used two standard datasets: TDT2 (Cieri et al. 1999) and Reuters-21578 (Lewis 2004) containing the news from 96 and 135 different topics, respectively. Documents in the original datasets can belong to more than one topic. Since topic detection is a hard clustering task, we only consider the documents with one topic. Our pruning reduces the topic count to 30 and 65 and the document count to 9.4 k and 8.3 k for TDT2 and Reuters-21578, respectively.

6.2 Summarization

It is shown in Sect. 4.2 that MMSE-2 performs slightly better than PW-1. Therefore, we compare the state-of-the-art extractive summarization methods with MMSE-2. We also include the results of Baseline-0 to indicate the significance of our learning method. In Baseline-0 the weights are set proportional to the pairwise similarities. We set $\omega_{ij} = \frac{1}{2}\delta_{ij}$ for Baseline-0.

We use the widely accepted metric ROUGE (Recall-Oriented Understudy for Gisting Evaluation) to evaluate the summaries (Lin 2004). More specifically, we utilize ROUGE-1 that is the overlap of unigrams between the system and reference summaries.

Table 4 compares the performance of state-of-the-art summarization systems using ROUGE-1 for evaluation. This table shows that our method significantly improves the results. The results of Table 3 are borrowed from the references mentioned in the corresponding cells.

Table 4 Comparison of summarization methods

DUC-04	R-1
Best system in DUC-04 (peer 65)	38.28
Wang et al. (2009)	39.07
Lin and Bilmes (2011)	39.35
Kulesza and Taskar (2012)	38.71
Baseline-0 (with $\frac{1}{2}$ coefficient)	40.15
Lin and Bilmes (2012)	40.43
Yasunaga et al. (2017)	38.22
MMSE-2	40.83

Table 5 Performance of different algorithms on k-means initialization of topic detection

	Reuters-21578			TDT2		
	F-value	Precision	Recall	F-value	Precision	Recall
k-means++ (average)	0.381	0.851	0.245	0.736	0.850	0.649
Baseline-0	0.398	0.839	0.261	0.739	0.843	0.672
MMSE-2	0.406	0.862	0.266	0.758	0.855	0.680

Table 6 Probability of getting better results than k-means++

	Reuters-21578			TDT2		
	F-value	Precision	Recall	F-value	Precision	Recall
SD of k-means++	0.013	0.007	0.011	0.013	0.007	0.018
p(Baseline-0 > k-means++)	0.911	0.060	0.925	0.582	0.166	0.892
p(MMSE-2 > k-means++)	0.975	0.941	0.969	0.947	0.741	0.954

Table 7 Clustering accuracy results on Reuters-21578 dataset with 10 largest clusters

Reuters-21578	Clustering accuracy (%)	Normalized mutual information (%)
LDA (argmax)	52.24	44.79
Oracle K-means (upper-bound)	64.73	40.23
Baseline-0 (with $\frac{1}{2}$ coefficient)	55.11	30.82
MGCTM (Xie and Xing 2013)	56.01	50.10
Li et al. (2017)	67.19	51.97
MMSE-2	62.33	34.73

6.3 Topic detection

In this section, the performance of MMSE-2 for locating the initial cluster centers is compared to k-means++ (Arthur and Vassilvitskii 2007). Since k-means++ uses randomness in selecting the initial centers, the algorithm is repeated 50 times and the results average is reported. Table 5 shows the results of k-means++, Baseline-0 and MMSE-2 on finding the initial centers for the k-means algorithm. As is observed in the table, both Baseline-0 and MMSE-2 outperform the semi-intelligent selection of k-means++. Furthermore, learning the weights (MMSE-2) instead of simply setting them proportional to similarities (Baseline-0) significantly improves the results.

Table 6 reports the probabilities that the results in Table 5 are better than the (random) k-means++. The probabilities are computed by the reasonable assumption that the distribution of k-means++ results is normal. In both of the tested datasets, MMSE-2 reaches a higher F-value than k-means++ with a probability of not less than 0.94.

In addition to initial cluster center locators, we compare our method on topic detection with state-of-the-art algorithms on the field. We test our method on Reuters-21578 dataset, since this is the most used dataset in previous works. In order for the comparison to be fair, we only consider the 10 largest clusters of Reuters-21578. This leaves us with 7.2 k documents (instead of 8.3 k documents corresponding to 65 clusters mentioned in Sect. 6.1). Similar to Xie and Xing (2013), Li et al. (2017) we use clustering accuracy and normalized mutual information as described in Zheng et al. (2011). Table 7 contains the results of the highest clustering accuracy systems on topic detection. The LDA (argmax) uses the document representation (topic vectors) and outputs the topic with maximum probability in document representation as the cluster of that document. The Oracle K-means (upper-bound) gives an upper-bound for the K-means performance on this task. This upper-bound is achieved by assuming an omniscience Oracle who knows the gold clustering and feeds the K-means algorithm with the golden initial centers. Clearly, the Oracle initial centers would result in the highest possible K-means performance among K-means initialization methods. As can be seen in Table 7, our method performs better than all but one of state-of-the-art systems and its performance is quite close to the K-means upper-bound.

We have to emphasize that our experiment is designed to show the performance of our generic submodular maximization method presented in this paper. Consequently, it is best to compare our results with other "K-means initialization methods" for topic detection. The results in Li et al. (2017) are superior to our method in topic detection. However, we do not try to present a topic detection algorithm. We are competing with K-means initialization methods, used on the task of topic detection. In this view, having a performance near the upper-bound, i.e., Oracle K-means, shows the significance of our method.

7 Conclusion

Most of the current graph-based algorithms use the pairwise similarities directly to build the edged weights of their graph model. We argued that in cases where the problem at hand has a submodular nature, the graph model can be represented by an influence graph. In such cases, considering the edge weights proportional to the pairwise similarities is not the best method to use. We proposed to use deeper methods which try to *learn* the edge weights such that the expected probability of one node being influenced by another node tends to their similarity value. Specifically, we have proposed two practical methods, PW-1 and MMSE-2, that locally try to adjust the edge weights to meet the aforementioned goal. We have also proposed a theoretically appealing GM-3 that has a global view to the problem. But due to its $O(n^3)$ time complexity, GM-3 is not practical for large problems.

PW-1 and MMSE-2 have been shown to perform well in learning the graph model when the submodular similarity function itself is obtained from an influence graph. In the intrinsic evaluation of these methods, we considered graphs with known edge weights. The gold weights were used to compute the influence probabilities. In the learning stage, the computed influence probabilities were considered as the similarity values and PW-1 and MMSE-2 were used to learn the edge weights. Obviously, the gold weights were not included in the learning process. Finally, the learned weights were compared to the gold weights to show the efficiency of PW-1 and MMSE-2.

Furthermore, we have experimentally shown that our proposed methods are beneficial in two NLP tasks: text summarization and topic detection. We have shown that MMSE-2 outperforms previous successful extractive summarization systems. It also significantly improves the simple method of setting the edge weights proportional to the pairwise similarities. For the topic detection task, we have used the k-means clustering algorithm. The contribution of MMSE-2, in that case, is to detect suitable initial cluster centers. We have shown that our method results in higher F-values compared to the famous k-means++ with a probability not less than 0.94.

This work can be extended in different directions. The proposed method can be applied to more submodular problems, not necessarily NLP related. Certain standard speedups can be introduced to the proposed methods. For example, intelligent node pruning methods may drastically decrease the graph size while having negligible effect on the system performance. Finally, this paper addresses problems in which a relation measure is definable between data points. It cannot be used on general submodular problems where every set has a value and no intrinsic relation measure can be defined between data points. One interesting work is to generalize the view of this paper to contain general submodular problems as well.

References

Agirre E, Martínez D, de Lacalle OL, Soroa A (2006) Two graph-based algorithms for state-of-the-art WSD. In: Proceedings of the 2006 conference on empirical methods in natural language processing. Association for Computational Linguistics, pp 585–593

Alexandrescu A, Kirchhoff K (2007) Data-driven graph construction for semi-supervised graph-based learning in NLP. In: Proceedings of the main conference human language technologies 2007: the conference of the North American chapter of the association for computational linguistics, pp 204–211

Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics, pp 1027–1035

Badanidiyuru A, Mirzasoleiman B, Karbasi A, Krause A (2014) Streaming submodular maximization. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining—KDD'14, pp 671–680

Baralis E, Cagliero L, Mahoto N, Fiori A (2013) GRAPHSUM: discovering correlations among multiple terms for graph-based summarization. Inf Sci 249:96–109

Beliga S, Mestrovic A, Martincic-Ipsic S (2015) An overview of graph-based keyword extraction methods and approaches. J Inf Organ Sci 39(1):1–20

Berton L, Valverde-Rebaza J, de Andrade Lopes A (2015) Link prediction in graph construction for supervised and semi-supervised learning. In: 2015 international joint conference on neural networks (IJCNN). IEEE, pp 1–8

Brin S, Page L (1998) The anatomy of a large-scale hypertextual web search engine. Comput Netw ISDN Syst 30(1–7):107–117

Celebi ME, Kingravi HA, Vela PA (2013) A comparative study of efficient initialization methods for the K-means clustering algorithm. Expert Syst Appl 40(1):200–210

Chekuri C, Jayram TS, Vondrak J (2015) On multiplicative weight updates for concave and submodular function maximization. In: Proceedings of the 2015 conference on innovations in theoretical computer science—ITCS'15, pp 201–210

Chen W, Wang C, Wang Y (2010) Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 1029–1038

Cieri C, Graff D, Liberman M, Martey N, Strassel S (1999) The TDT-2 text and speech corpus. In: Proceedings of the broadcast news workshop'99, p 57

Erkan G, Radev DR (2004) LexRank: graph-based lexical centrality as salience in text summarization. J Artif Intell Res 22:457–479

Galluccio L, Michel O, Comon P, Hero AO III (2012) Graph based k-means clustering. Signal Process 92(9):1970–1984

Granovetter MS (1973) The strength of weak ties. Am J Sociol 78(6):1360–1380

Herings P, Van der Laan G, Talman D (2001) Measuring the power of nodes in digraphs. Technical report, Tinbergen Institute

Huang B, Yang Y, Mahmood A, Wang H (2012) Microblog topic detection based on LDA model and single-pass clustering. Int Conf Rough Sets Curr Trends Comput 2012:166–171

Kazemi E, Zadimoghaddam M, Karbasi A (2018) Scalable deletion-robust submodular maximization: data summarization with

privacy and fairness constraints. In: International conference on machine learning, pp 2549–2558
Kleinberg JM (1999) Authoritative sources in a hyperlinked environment. J ACM (JACM) 46(5):604–632
Klimt B, Yang Y (2004) Introducing the Enron corpus. In: CEAS
Kulesza A, Taskar B (2012) Determinantal point processes for machine learning. Found Trends® Mach Learn 5(2–3):123–286
Leskovec J (2018) Stanford large network dataset collection. https://snap.stanford.edu/data/index.html. Accessed 1 May 2019.
Leskovec J, Grobelnik M, Milic-Frayling N (2004) Learning semantic graph mapping for document summarization. In: Proceedings of ECML/PKDD-2004 workshop on knowledge discovery and ontologies
Leskovec J, Kleinberg J, Faloutsos C (2007) Graph evolution. ACM Trans Knowl Discov Data 1(2):1–39
Leskovec J, Lang KJ, Dasgupta A, Mahoney MW (2009) Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. Internet Math 6(1):29–123
Lewis DD (2004) Reuters-21578 text categorization test collection. http://www.daviddlewis.com/resources/testcollections/reuters21578/. Accessed 1 May 2019
Li W, Joo J, Qi H, Zhu S-C (2017) Joint image-text news topic detection and tracking by multimodal topic and-or graph. IEEE Trans Multimed 19(2):367–381
Lin C-Y (2004) ROUGE: a package for automatic evaluation of summaries. In: Proceedings of the workshop on text summarization branches out (WAS 2004), Barcelona, Spain, July 25–26
Lin H, Bilmes J (2010) Multi-document summarization via budgeted maximization of submodular functions. In: HLT'10 human language technologies: the 2010 annual conference of the North American chapter of the association for computational linguistics, pp 912–920
Lin H, Bilmes J (2011) A class of submodular functions for document summarization. Comput Linguist 1:510–520
Lin H, Bilmes J (2012) Learning mixtures of submodular shells with application to document summarization. arXiv preprint arXiv:1210.4871
Matsuo Y, Sakaki T, Uchiyama K, Ishizuka M (2006) Graph-based word clustering using a web search engine. In: Proceedings of the 2006 conference on empirical methods in natural language processing. Association for Computational Linguistics, pp 542–550
Mei Q, Guo J, Radev D (2010) DivRank: the interplay of prestige and diversity in information networks. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1009–1018
Mihalcea R (2004) Graph-based ranking algorithms for sentence extraction, applied to text summarization. In: Proceedings of the ACL 2004 on interactive poster and demonstration sessions. Association for Computational Linguistics, p 20
Mihalcea R, Tarau P (2004) TextRank: bringing order into texts. Proc EMNLP 85:404–411
Mirzasoleiman B, Sarkar R, Krause A (2013) Distributed submodular maximization: identifying representative elements in massive data. Adv Neural Inf Process Syst 26:2049–2057
Mirzasoleiman B, Karbasi A, Sarkar R, Krause A (2016) Distributed submodular maximization. J Mach Learn Res 17(1):8330–8373
Mirzasoleiman B, Jegelka S, Krause A (2018) Streaming non-monotone submodular maximization: personalized video summarization on the fly. In: Thirty-second AAAI conference on artificial intelligence
Nemhauser GL, Wolsey LA, Fisher ML (1978) An analysis of approximations for maximizing submodular set functions—I. Math Program 14(1):265–294
Pilehvar MT, Navigli R (2015) From senses to texts: an all-in-one graph-based approach for measuring semantic similarity. Artif Intell 228:95–128
Rosen-Zvi M, Griffiths T, Steyvers M, Smyth P (2004) The author-topic model for authors and documents. In: Proceedings of the 20th conference on uncertainty in artificial intelligence, pp 487–494
Spina D, Gonzalo J, Amigó E (2014) Learning similarity functions for topic detection in online reputation monitoring. In: Proceedings of the 37th international ACM SIGIR conference on research and development in information retrieval—SIGIR'14, pp 527–536
Tang Y, Xiao X, Shi Y (2014) Influence maximization: near-optimal time complexity meets practical efficiency. Kdd 2014:75–86
Tixier AJ, Meladianos P, Vazirgiannis M (2017) Combining graph degeneracy and submodularity for unsupervised extractive summarization. In: Proceedings of the workshop on new frontiers in summarization, pp 48–58
Vardasbi A, Faili H, Asadpour M (2017) SWIM: stepped weighted shell decomposition influence maximization for large-scale networks. ACM Trans Inf Syst 36(1):1–33
Véronis J (2004) Hyperlex: lexical cartography for information retrieval. Comput Speech Lang 18(3):223–252
Wang D, Zhu S, Li T, Gong Y (2009) Multi-document summarization using sentence-based topic models. In: Proceedings of the ACL-IJCNLP 2009 conference short papers, pp 297–300
Wang C, Yu X, Li Y, Zhai C, Han J (2013) Content coverage maximization on word networks for hierarchical topic summarization. In: Proceedings of the 22nd ACM international conference on conference on information and knowledge management—CIKM'13, pp 249–258
Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world' networks. Nature 393(6684):440–442
Weng, J, Yao Y, Leonardi E, Lee F (2011) Event detection in Twitter. Development, pp 401–408
Xie P, Xing EP (2013) Integrating document clustering and topic modeling. arXiv preprint arXiv:1309.6874
Yang J, Leskovec J (2015) Defining and evaluating network communities based on ground-truth. Knowl Inf Syst 42(1):181–213
Yang Y, Pierce T, Carbonell J (1998) A study of retrospective and on-line event detection. In: Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval—SIGIR'98, pp 28–36
Yasunaga, M, Zhang R, Meelu K, Pareek A, Srinivasan K, Radev D (2017) Graph-based neural multi-document summarization. arXiv preprint arXiv:1706.06681
Zheng M, Bu J, Chen C, Wang C, Zhang L, Qiu G, Cai D (2011) Graph regularized sparse coding for image representation. IEEE Trans Image Process 20(5):1327–1336
Zhou, T, Ouyang H, Chang Y, Bilmes J, Guestrin C (2016) Scaling submodular maximization via pruned submodularity graphs. arXiv preprint arXiv:1606.00399

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.