RECOMMENDER SYSTEMS AND COMMUNITY DETECTION IN GRAPHS
ΣΥΣΤΗΜΑΤΑ ΠΡΟΤΑΣΗΣ & ΑΝΑΓΝΩΡΙΣΗ ΚΟΙΝΟΤΗΤΩΝ ΣΕ ΓΡΑΦΟΥΣ

By

EVANGELOS CHARALAMPAKIS

A THESIS

BSc, Department of Computer Science, University of Crete, 2009

Submitted in partial fulfillment of the requirements for the degree
MASTER OF SCIENCE

DEPARTMENT OF INFORMATICS ENGINEERING
SCHOOL OF ENGINEERING
TECHNOLOGICAL EDUCATIONAL INSTITUTE OF CRETE

2018

Approved by:
Professor
Paraskevi Fragopoulou
and
Assistant Professor
Haralambos Papadakis

Statement of Originality

The work contained in this thesis has not been previously submitted for a degree or diploma at any other higher education institution or any other purpose. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except as specified in references, acknowledgements or in footnotes. I certify that the intellectual content of this thesis is the product of my own work and all the assistance received in preparing this thesis and sources have been acknowledged.

Evangelos Charalampakis

Acknowledgements

I would like to express my very great appreciation to my professors, Mrs. Fragopoulou Paraskevi and the assistant professor Mr. Papadakis Haralambos for their valuable and constructive suggestions during the planning and development of this research work. Their willingness to give time so generously and their patient to me, has been very much appreciated.

Abstract

This thesis consists of two parts.

The first part presents an overview of the literature and the proposals that exist in general, with which techniques work, what properties each one has, the pros and cons of each method, what problems they are facing and how these systems are evaluated. Each algorithm is presented along with examples from the real world.

At the end of the first part of the thesis is an extensive description as well as results from an experiment based on the SCoR algorithm, an algorithm constructed to produce optimal recommendations based on the modified Vivaldi algorithm.

The second part is about community detection systems in social media. This has to do with systems that, based on some “relationships” of the end-users of a social network, discover the "hidden" community of users hidden behind these explicit relationships. There is a description of how such a community can be formed at the graph level, depending on the logic of the algorithm that is required to discover this graph.

We present the available techniques that are used by the various approaches. In addition, the best-known algorithms of the field are presented with results from experiments that have been performed and the advantages and disadvantages are presented for each case.

At the end of the second part, we present the results of the experiment of the SCCD algorithm, which is again based on the Vivaldi algorithm and performs accurate community detection.

Keywords

Community detection, Recommender systems, Social networks, Dynamic graphs, Information filtering, Synthetic coordinates, Vivaldi, SCoR, SCCD

Περίληψη

Η εργασία αυτή αποτελείται από δύο σκέλη.

Το πρώτο σκέλος έχει να κάνει με τα συστήματα συστάσεως που υπάρχουν γενικά, τις τεχνικές στις οποίες βασίζονται, τις ιδιότητες έχει το καθένα, τα υπέρ και τα κατά της κάθε μεθόδου, τα προβλήματα που καλούνται να αντιμετωπίσουν καθώς και τους τρόπους με τους οποίους αυτά τα συστήματα αξιολογούνται. Γίνεται μια παρουσίαση κάθε καταγεγραμμένης τεχνικής – αλγορίθμου και γίνεται αναφορά σε συγκεκριμένα παραδείγματα από τον πραγματικό κόσμο.

Στο τέλος του πρώτου σκέλους υπάρχει εκτενής περιγραφή του αλγορίθμου SCoR καθώς και τα αποτελέσματα των πειραμάτων που έγιναν με βάση τον SCoR, έναν αλγόριθμο που κατασκευάστηκε για να παράγει βέλτιστες συστάσεις δεδομένων και βασίζεται στον τροποποιημένο αλγόριθμο Vivaldi.

Το δεύτερο σκέλος αυτής της μεταπτυχιακής εργασίας έχει να κάνει με τα συστήματα αναγνώρισης κοινοτήτων στα μέσα κοινωνικής δικτύωσης. Εστιάζει σε συστήματα τα οποία με βάση κάποιες επιλογές των τελικών χρηστών ενός κοινωνικού δικτύου, έρχονται να ανακαλύψουν τις «κρυφές» κοινότητες των χρηστών όπως αυτές διαμορφώνονται από τις «κοινές» επιλογές των χρηστών. Γίνεται μια περιγραφή για το πως μπορούν να σχηματιστούν αυτές οι κοινότητες σε επίπεδο γράφων, ανάλογα με τη λογική του αλγορίθμου ο οποίος καλείται να τις εντοπίσει.

Παρουσιάζονται οι διαθέσιμες τεχνικές που μπορούν να χρησιμοποιηθούν από τους διάφορους αλγορίθμους. Επιπλέον παρουσιάζονται οι πιο γνωστοί αλγόριθμοι του χώρου με αποτελέσματα από πειράματα που έχουν γίνει και αποτυπώνονται τα υπέρ και τα κατά της κάθε περίπτωσης.

Στο τέλος του δεύτερου σκέλους γίνεται παρουσίαση των αποτελεσμάτων του πειράματος του αλγορίθμου SCCD ο οποίος είναι πάλι βασισμένος στον τροποποιημένο αλγόριθμο του Vivaldi.

Λέξεις κλειδιά

Συστήματα Αναγνώρισης Κοινοτήτων, Συστήματα Συστάσεως, Κοινωνικά Δίκτυα, Δυναμικοί Γράφοι, Φιλτράρισμα Δεδομένων, Συνθετικές Συντεταγμένες, Vivaldi, SCoR, SCCD

Περιεχόμενα

List of Tables

List of Figures

List of Equations

Chapter 1: Recommender Systems

Contents

1.1 Recommender systems

The growth of the internet gives people the ability to access large volume of information. Electronic commerce sites, like E-bay and Amazon have millions of items to choose from, music sites like Spotify, contain millions of songs and these days most people are connected to some sort of social media.

It can be hard to find what you're looking for with this huge amount of information. To solve this problem, companies began to form and create systems that are able to filter out unwanted content and recommend content / products that someone would be eager to watch / buy with high probability.

These systems are called Recommender or Recommendation Systems. Over time, these systems are becoming more accurate in making predictions. There is a big variety of types of recommendation systems as well as different ways to implement them.

The abundance of information available on the Web and in Digital Libraries, in combination with their dynamic and heterogeneous nature, has determined a rapidly increasing difficulty in finding what we want when we need it and, in a manner, which best meets our requirements. As a consequence, the role of user modeling and personalized information access is becoming crucial: users need personalized support in sifting through large amounts of available information, according to their interests and tastes. Many information sources embody recommender systems as a way of personalizing their content for users. [1]

- Recommender systems have the effect of guiding users -in a personalized way- to find interesting or useful objects in a large space of possible options.
- Recommendation systems use as input a person's interests in order to generate a list of recommended items for the individual person [2].
- The goal of a recommender system is to generate meaningful recommendations to a collection of users for items or products that might interest them [2].

Before the widespread use of computers and internet, people relied on traditional stores to purchase the items they needed. But traditional stores have a limited amount of space and, therefore, are able only to carry a limited amount of goods. The decision of what goods to carry would simply be based on what was selling at the time or by store owner's preference. This meant there was a small selection from which customers could choose. [3]

However, this changed when the widespread use of computers and internet came around. Sites like Amazon and E-bay became popular, and they could store a nearly infinite number of items on their virtual shelves. This created some problems, however. With so many options, online customers had a difficult time finding the items they would want; because online stores want to sell as many items as they can, they needed to find a way to filter out the user's unwanted items. Online stores had the additional goal of showing items to customers that they would want to purchase which, if done successfully, would drive up store sales.

These factors inspired the demand of companies / organizations to create recommendation systems.

Classification

Recommender systems are usually classified according to their approach to rate/predict user preference values. Recommender systems are usually classified into the following categories, based on how recommendations are made:

- **Content-based**: User should be recommended with items similar to those ones that the same user preferred in the past
- **Collaborative filtering**: User should be recommended with items that people with similar tastes and preferences with him, liked / chose in the past
- **Hybrid approaches**: These methods combine collaborative filtering and content-based methods. [4]

In addition to recommender systems that predict the absolute values of ratings that individual users would give to the yet unseen items, there has been work done on preference-based filtering, i.e., predicting the relative preferences of users. For example, in a movie recommendation application, preference-based filtering techniques would focus on predicting the correct relative order of the movies, rather than their individual ratings [5].

Techniques

There are several techniques used in recommendations systems. Below some of them are summarized.

Association rules: Association rule mining is one of the most popular data mining methods and it is used widely for marketing reasons. This class of algorithms extracts

rules that predict the occurrence of an item based on the presence of other items in a transaction.

Bayesian classifiers: Bayesian classifiers consider all features and classes of a learning problem as random, continuous or discrete variables. Using conditional probabilities and the Bayes' Theorem, the goal of a Bayesian classifier is to maximize the posterior probability of the class of any item to classify given data. Using ratings as classes with discrete values, a Bayesian classifier can be applied to real-valued rating data. [6]

The naïve Bayesian approach assumes that the features (the users, or the items) are independent given a class, the class being for instance a rating taking a discrete value. With this simplification, the probability of the class given all features can be computed very efficiently.

Neural networks approaches: Neural Networks (NN) approaches (or Artificial Neural Networks) are very common in machine learning, signal processing and data mining applications. Many Artificial Neural Networks exist, the most famous class being the multilayer perceptron using the backpropagation algorithm. Neural Networks are not often used in recommender systems. [7]

There might be several reasons:

- Classical large neural networks in high dimensional problems are known to learn (converge) very slowly.
- For classification tasks in content-based systems, maybe there is no need to complex non-linear classifiers.
- NN have a black box effect, that is to say the output of a NN cannot easily be interpreted.

K-Nearest Neighbor approaches: K-Nearest Neighbor (KNN) approaches, also called memory-based approaches, are a mainstream in the recommendation system field. They generalize the association rule principles to compare either items or users globally, on binary or real-valued data. They are well adapted to the core functions of recommenders. They can easily be adapted for regression so they can provide ranking and rating predictions. They are also well-suited for item-to-item recommendation.

Their only weakness is the lack of scalability when the number of objects to compare during the KNN search increases. The exact algorithm is intrinsically quadratic: the time to build the model is proportional to the square of the number of objects to compare. [5]

Matrix factorization: As the Netflix Prize competition has demonstrated, matrix factorization models are superior to classic nearest-neighbor techniques for producing product recommendations, allowing the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels. [8]

Other techniques: Several techniques used in the Information Retrieval domain are also often used in Recommender Systems when items can be represented by text documents.

1.2 Recommender System Properties

This section points out a range of properties that are commonly considered when deciding which recommendation approach to select. As different applications have different needs, the designer of the system must decide on the important properties to measure for the concrete application at hand. Several of the properties present a traded-off, the most obvious example perhaps is the decline in accuracy when other properties (e.g. diversity) are improved. It is important to understand and evaluate these trade-offs and their effect on the overall performance. [9]

User Preference: We need to choose one out of a set of candidate algorithms, an obvious option is to run a user study (within subjects) and ask the participants to choose one of the systems. This evaluation does not restrict the subjects to specific properties, and it is generally easier for humans to make such judgments than to give scores for the experience. [9]

Prediction Accuracy: Prediction accuracy is by far the most discussed property in the recommendation system literature. At the base of the vast majority of recommender systems lie a prediction engine. This engine may predict user opinions over items (e.g. ratings of movies) or the probability of usage (e.g. purchase).

A basic assumption in a recommender system is that a system that provides more accurate predictions will be preferred by the user. Thus, many researchers set out to find algorithms that provide better predictions.

Prediction accuracy is typically independent of the user interface and can thus be measured in an offline experiment. Measuring prediction accuracy in a user study measures the accuracy given a recommendation. This is a different concept from the prediction of user behavior without recommendations and is closer to the true accuracy in the real system. [9]

Coverage: As the prediction accuracy of a recommendation system, especially in collaborative filtering systems, in many cases grows with the amount of data, some algorithms may provide recommendations with high quality, but only for a small portion of the items where they have huge amounts of data. The term coverage can refer to several distinct properties of the system that we discuss below. [9]

Confidence: Confidence in the recommendation can be defined as the system's trust in its recommendations or predictions. Collaborative filtering recommenders tend to improve their accuracy as the amount of data over items grows. Similarly, the confidence in the predicted property typically also grows with the amount of data.

In many cases the user can benefit from observing these confidence scores. When the system reports a low confidence in a recommended item, the user may tend to further research the item before making a decision. For example, if a system recommends a movie with very high confidence, and another movie with the same rating but a lower confidence, the user may add the first movie immediately to the watching queue but may further read the plot synopsis for the second movie, and perhaps a few movie reviews before deciding to watch it. [9]

Trust: While confidence is the system trust in its ratings, with the term "trust" we refer to the user's trust in the system recommendations. For example, it may be beneficial for the system to recommend a few items that the user already knows and likes. This way, even though the user gains no value from this recommendation, she observes that the system provides reasonable recommendations, which may increase her trust in the system recommendations for unknown items. Another common way of enhancing trust in the system is to explain the recommendations that the system provides.

Trust in the systems is also called the credibility of the system. If we do not restrict ourselves to a single method of gaining trust, such as the one suggested above, the obvious method for evaluating user trust is by asking users whether the system recommendations are reasonable. [9]

Novelty: Novel recommendations are recommendations for items that the user did not know about.

In applications that require novel recommendation, an obvious and easy to implement approach is to filter out items that the user already rated or used. [9]

Serendipity: Serendipity is a measure of how surprising the successful recommendations are. For example, if the user has rated positively many movies where a certain star actor appears, recommending the new movie of that actor may be novel,

because the user may not know of it, but is hardly surprising. Of course, random recommendations may be very surprising, and we therefore need to balance serendipity with accuracy.

One can think of serendipity as the amount of relevant information that is new to the user in a recommendation. For example, if following a successful movie recommendation, the user learns of a new actor that she likes, this can be considered as serendipitous. In information retrieval, where novelty typically refers to the new information contained in the document (and is thus close to our definition of serendipity), Zhang et al. [63] suggested to manually label pairs of documents as redundant. Then, they compared algorithms on avoiding recommending redundant documents.

One can also think of serendipity as deviation from the "natural" prediction. That is, given a prediction engine that has a high accuracy, the recommendations that it issues are "obvious". Therefore, we will give higher serendipity scores to successful recommendations that the prediction engine would deem unlikely. [9]

Diversity: Diversity is generally defined as the opposite of similarity. In some cases, suggesting a set of similar items may not be as useful for the user, because it may take longer to explore the range of items. Consider for example a recommendation for a vacation, where the system should recommend vacation packages. Presenting a list with five recommendations, all for the same location, varying only on the choice of hotel, or the selection of attraction, may not be as useful as suggesting five different locations. The user can view the various recommended locations and request more details on a subset of the locations that are appropriate to her. [9]

Utility: Many e-commerce websites employ a recommendation system in order to improve their revenue by, e.g., enhancing cross-sell. In such cases the recommendation engine can be judged by the revenue that it generates for the website. In general, we can define various types of utility functions that the recommender tries to optimize. For such recommenders, measuring the utility, or the expected utility of the recommendations may be more significant than measuring the accuracy of recommendations. It is also possible to view many of the other properties, such as diversity or serendipity, as different types of utility functions, over single items or over lists. [9]

Risk: In some cases, a recommendation may be associated with a potential risk. For example, when recommending stocks for purchase, users may wish to be risk-averse,

preferring stocks that have a lower expected growth, but also a lower risk of collapsing. On the other hand, users may be risk-seeking, preferring stocks that have a potentially high, even if less likely, profit. [9]

Robustness: Robustness is the stability of the recommendation in the presence of fake information, typically inserted on purpose in order to influence the recommendations. As more people rely on recommender systems to guide them through the item space, influencing the system to change the rating of an item may be profitable to an interested party. [9]

Privacy: In a collaborative filtering system, a user willingly discloses his preferences over items to the system in the hope of getting useful recommendations. However, it is important for most users that their preferences stay private, that is, that no third party can use the recommendation system to learn something about the preferences of a specific user.

It is generally considered inappropriate for a recommendation system to disclose private information even for a single user. For this reason analysis of privacy tends to focus on a worst case scenario, illustrating theoretical cases under which user's private information may be revealed. [9]

Adaptivity: Real recommendation systems may operate in a setting where the item collection changes rapidly, or where trends in interest over items may shift. Perhaps the most obvious example of such systems is the recommendation of news items or related stories in online newspapers. In this scenario stories may be interesting only over a short period of time, afterwards becoming outdated. When an unexpected news event occurs, such as the tsunami disaster, people become interested in articles that may not have been interesting otherwise, such as a relatively old article explaining the tsunami phenomenon. [9]

Scalability: As recommender systems are designed to help users navigate in large collections of items, one of the goals of the designers of such systems is to scale up to real data sets. As such, it is often the case that algorithms trade other properties, such as accuracy or coverage, for providing rapid results even for huge data sets consisting of millions of items. [9]

1.3 Major Challenges

Researchers in the field of recommender systems face several challenges which pose danger for the use and performance of their algorithms. Here we mention only the major ones:

Data sparsity: Since the pool of available items is often exceedingly large (major online bookstores offer several millions of books, for example), overlap between two users is often very small or none. Further, even when the average number of evaluations per user/item are high, they are distributed among the users/items very unevenly and hence majority of users/items may have expressed/received only a few ratings. So, an effective recommender algorithm must take the data sparsity into account.

Scalability: While the data is mostly sparse, for major sites it includes millions of users and items. It is therefore essential to consider the computational cost issues and search for recommender algorithms that are either little demanding or easy to parallelize (or both). Another possible solution is based on using incremental versions of the algorithms where, as the data grows, recommendations are not recomputed globally (using the whole data) but incrementally (by slightly adjusting previous recommendations according to the newly arrived data).

Cold start: When new users enter the system, there is usually insufficient information to produce recommendation for them. The usual solutions of this problem are based on using hybrid recommender techniques combining content and collaborative data and sometimes they are accompanied by asking for some base information (such as age, location and preferred genres) from the users. Another way is to identify individual users in different web services.

Diversity vs. accuracy: When the task is to recommend items, which are likely to be appreciated by a particular user, it is usually most effective to recommend popular and highly rated items. Such recommendation, however, has very little value for the users because popular objects are easy to find (often they are even hard to avoid) without a recommender system. A good list of recommended items hence should contain also less obvious items that are unlikely to be reached by the users themselves. Approaches to this problem include direct enhancement of the recommendation list's diversity and the use of hybrid recommendation methods.

Vulnerability to attacks: Due to their importance in e-commerce applications, recommender systems are likely targets of malicious attacks trying to unjustly promote

or inhibit some items. There is a wide scale of tools preventing this kind of behavior, ranging from blocking the malicious evaluations from entering the system to sophisticated resistant recommendation technique. However, this is not an easy task since the strategies of attackers also get more and more advanced as the developing of preventing tools.

The value of time: While real users have interests with widely diverse time scales (for example, short term interests related to a planned trip and long-term interests related to the place of living or political preferences), most recommendation algorithms neglect the time stamps of evaluations. It is an ongoing line of research whether and how value of old opinions should decay with time and what are the typical temporary patterns in user evaluations and item relevance. [10]

1.4 Recommendation Evaluation Metrics

The evaluation of recommender systems is often done in terms of accuracy, that is to say, of predictive performance for the task of rating prediction. This implies a dataset of user ratings on items, usually in the form of user's rating logs. As in a classical machine learning test protocol, the predicted scores on the items and the actual notes on the items are compared. Several measures are employed, the most famous being the **Mean Absolute Error (MAE)** and the **Root Mean Squared Error (RMSE).**

Rating-based metrics:

The **Root Mean Squared Error**, RMSE, is perhaps the most popular metric used in evaluating accuracy of predicted ratings. The system generates predicted ratings $\hat{r}_{ui}$ for a test set T of user-item pairs (u, i) for which the true ratings $\hat{r}_{ui}$ are known.

Typically, $\hat{r}_{ui}$ are known because they are hidden in an offline experiment, or because they were obtained through a user study or online experiment.:

The RMSE between the predicted and actual ratings is given by:

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\mathscr{T}|} \sum_{(u,i)\in\mathscr{T}} (\hat{r}_{ui} - r_{ui})^2}$$

Equation 1 - RMSE metric

Mean Absolute Error (MAE) is a popular alternative, given by:

$$\mathrm{MAE} = \sqrt{\frac{1}{|\mathscr{T}|} \sum_{(u,i)\in\mathscr{T}} |\hat{r}_{ui} - r_{ui}|}$$

Equation 2 - MAE metric

Compared to MAE, RMSE disproportionably penalizes large errors, so that, given a test set with four hidden items RMSE would prefer a system that makes an error of 2 on three ratings and 0 on the fourth to one that makes an error of 3 on one rating and 0 on all three others, while MAE would prefer the second system.

1.5 Contend Based Recommender Systems

Content-based filtering methods are based on a description of the item and a profile of the user's preference. In a content-based recommender system, keywords are used to describe the items; besides, a user profile is built to indicate the type of item this user likes. In other words, these algorithms try to recommend items that are similar to those that a user liked in the past (or is examining in the present). In particular, various candidate items are compared with items previously rated by the user and the best-matching items are recommended. This approach has its roots in information retrieval and information filtering research.

To abstract the features of the items in the system, an item presentation algorithm is applied. A widely used algorithm is the TFIDF [11] representation (also called vector space representation).

To create user profile, the system mostly focuses on two types of information:

1. A model of the user's preference.
2. A history of the user's interaction with the recommender system.

Basically, these methods use an item profile (i.e. a set of discrete attributes and features) characterizing the item within the system. The system creates a content-based profile of users based on a weighted vector of item features. The weights denote the importance of each feature to the user and can be computed from individually rated content vectors using a variety of techniques. Simple approaches use the average values of the rated item vector while other sophisticated methods use machine learning techniques such as Bayesian Classifiers, cluster analysis, decision trees, and artificial neural networks in order to estimate the probability that the user is going to like the item. Direct feedback

from a user, usually in the form of a like or dislike button, can be used to assign higher or lower weights on the importance of certain attributes.

A key issue with content-based filtering is whether the system is able to learn user preferences from user's actions regarding one content source and use them across other content types. When the system is limited to recommending content of the same type as the user is already using, the value from the recommendation system is significantly less than when other content types from other services can be recommended. For example, recommending news articles based on browsing of news is useful, but it's much more useful when music, videos, products, discussions etc. from different services can be recommended based on news browsing. [12]

The content-based approach to recommendation has its roots in information retrieval and information filtering research. Because of the significant and early advancements made by the information retrieval and filtering communities and because of the importance of several text-based applications, many current content-based systems focus on recommending items containing textual information, such as documents, Web sites (URLs), and Usenet news messages. The improvement over the traditional information retrieval approaches comes from the use of user profiles that contain information about users' tastes, preferences, and needs. The profiling information can be elicited from users explicitly, e.g., through questionnaires, or implicitly—learned from their transactional behavior over time. [5]

Pros and cons of content-based filtering systems

The content-based methods allow overcoming some limitations of the collaborative filtering:

- They can provide recommendations for a new item even if no rating is available for it;
- They can manage situations where different users do not share identical items, but only similar items according to their intrinsic characteristics (metadata).

However, the content-based filtering methods require rich descriptions of items and well-built and well-informed user profiles. These ideal cases are rare in real applications. This dependence on the quality and structure of data is the main weakness of methods based on content. Since it is difficult in many areas, to obtain structured and complete descriptions of the items, content-based methods have mainly been applied to

catalogs of textual information, such as documents, pages, websites, or message forums. Another weak point of content-based methods is their tendency to recommend items very similar to items already seen and rated by users, a phenomenon called overspecialization. The “help to discover” aspect, and the originality of the recommendations, are strongly reduced as compared to collaborative filtering. On the other hand, the fact that users can get recommendations without sharing their profile ensures their privacy.

Content-based approaches appear symmetrical and complementary methods to collaborative filtering. Where collaborative methods involve centralized data on thousands of users, content-based approaches generally require usages of only one user. Where collaborative filtering is content-agnostic (i.e. independent to the items' metadata), content-based approaches will be effective only for catalogs of items represented by rich metadata. The complementary of content-based methods to collaborative methods made them good candidates to hybridization techniques

Disadvantages

In the content-based approach, the system must be capable of matching the characteristics of an item against relevant features in the user's profile. In order to do this, it must first construct a sufficiently-detailed model of the user's tastes and preferences through preference elicitation. This may be done either explicitly (by querying the user) or implicitly (by observing the user's behavior). In both cases, the cold start problem would imply that the user has to dedicate an amount of effort using the system in its 'dumb' state – contributing to the construction of their user profile – before the system can start providing any intelligent recommendations.

In the collaborative filtering approach, the recommender system would identify users who share the same preferences (e.g. rating patterns) with the active user, and propose items which the like-minded users favored (and the active user has not yet seen). Due to the cold start problem, this approach would fail to consider items which no-one in the community has rated previously.

The cold start problem is also exhibited by interface agents. Since such an agent typically learn the user's preferences implicitly by observing patterns in the user's behavior – “watching over the shoulder” – it would take time before the agent may perform any adaptations personalized to the user. Even then, its assistance would be limited to activities which it has formerly observed the user engaging in

Limitations and Extensions

Although there are different approaches to learning a model of the user's interest with content-based recommendation, no content-based recommendation system can give good recommendations if the content does not contain enough information to distinguish items the user likes from items the user doesn't like. In recommending some items, e.g., jokes or poems, there often isn't enough information in the word frequency to model the user's interests. While it would be possible to tell a lawyer joke from a chicken joke based upon word frequencies, it would be difficult to distinguish a funny lawyer joke from other lawyer jokes. As a consequence, other recommendation technologies, such as collaborative recommenders, should be used in such situations.

In some situations, e.g., recommending movies, restaurants, or television programs, there is some structured information (e.g., the genre of the movie as well as actors and directors) that can be used by a content-based system. However, this information might be supplemented by the opinions of other users. A final usage of content in recommendations is worth noting. Simple content-based rules may be used to filter the results of other methods such as collaborative filtering. For example, even if it is the case that people who buy dolls also buy adult videos, it might be important not to recommend adult items in a particular application. Similarly, although not strictly content-based, some systems might not recommend items that are out of stock. [13]

1.6 Collaborative Filtering Recommender Systems

The fundamental assumption of Collaborative Filtering is that if users *X* and *Y* rate *n* items similarly, or exhibit similar behavior (e.g., buying, watching, listening), and hence will rate or act on other items similarly [14].

Collaborative Filtering techniques use a database of preferences for items by users to predict additional topics or products a new user might like. In a typical Collaborative Filtering scenario, there is a list of *m* users $\{u_1, u_2, \ldots, u_m\}$ and a list of *n* items $\{i_1, i_2, \ldots, i_n\}$, and each user, u_i, has a list of items, I_{ui}, which the user has rated, or about which their preferences have been inferred through their behaviors. The ratings can either be explicit indications, and so forth, on a 1–5 scale, or implicit indications, such as purchases or click-throughs [14].

Alice: (like) Shrek, Snow White, (dislike) Superman
Bob: (like) Snow White, Superman, (dislike) spiderman

Chris: (like) spiderman, (dislike) Snow white

Tony: (like) Shrek, (dislike) Spiderman

Figure 1 - An example of user-item matrix

	Shrek	**Snow White**	**Spider-man**	**Super-man**
Alice	Like	Like		Dislike
Bob		Like	Dislike	Like
Chris		Dislike	Like	
Tony	Like		Dislike	?

Figure 2. An example of user-item matrix

Collaborative filtering (CF) is a technique used by some recommender systems. CF has two senses, a narrow one and a more general one. In general, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Applications of collaborative filtering typically involve very large data sets. CF methods have been applied to many different kinds of data including: sensing and monitoring data, such as in mineral exploration, environmental sensing over large areas or multiple sensors; financial data, such as financial service institutions that integrate many financial sources; or in electronic commerce and web applications where the focus is on user data, etc. The remainder of this discussion focuses on collaborative filtering for user data, although some of the methods and approaches may apply to the other major applications as well.

In the newer, narrower sense, collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue x than to have the opinion on x of a person chosen randomly. For example, a collaborative filtering recommendation system for television tastes could make predictions about which television show a user should like given a partial list of that user's tastes (likes or dislikes). Note that these predictions are specific to the user, but use information gleaned from many users. This differs from the simpler approach of giving an average (non-specific) score for each item of interest, for example based on its number of votes [12].

Methodology

CF systems have many forms, but many common systems can be reduced to two steps:

1. Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
2. Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user

This falls under the category of user-based CF. A specific application of this is the user-based Nearest Neighbor algorithm.

Alternatively, item-based collaborative filtering invented by Amazon.com (users who bought *x* also bought *y*), proceeds in an item-centric manner:

- Build an item-item matrix determining relationships between pairs of items
- Infer the tastes of the current user by examining the matrix and matching that user's data

Another form of CF can be based on implicit observations of normal user behavior (as opposed to the artificial behavior imposed by a rating task). These systems observe what a user has done together with what all users have done (what music they have listened to, what items they have bought) and use that data to predict the user's behavior in the future, or to predict how a user might like to behave given the chance. These predictions then have to be filtered through business logic to determine how they might affect the actions of a business system. For example, it is not useful to offer to sell somebody a particular album of music if they already have demonstrated that they own that music.

Relying on a scoring or rating system which is averaged across all users ignores specific demands of a user and is particularly poor in tasks where there is large variation in interest (as in the recommendation of music). However, there are other methods to combat information explosion, such as web search and data clustering

CF categories	Representative techniques	Main advantages	Main shortcomings
Memory-based CF	*Neighbor-based CF (item-based/user-based CF algorithms with Pearson/vector cosine correlation) *Item-based/user-based top-*N* recommendations	*easy implementation *new data can be added easily and incrementally *need not consider the content of the items being recommended *scale well with co-rated items	*are dependent on human ratings *performance decrease when data are sparse *cannot recommend for new users and items *have limited scalability for large datasets
Model-based CF	*Bayesian belief nets CF *clustering CF *MDP-based CF *latent semantic CF *sparse factor analysis *CF using dimensionality reduction techniques, for example, SVD, PCA	*better address the sparsity, scalability and other problems *improve prediction performance *give an intuitive rationale for recommendations	*expensive model-building *have trade-off between prediction performance and scalability *lose useful information for dimensionality reduction techniques
Hybrid recommenders	*content-based CF recommender, for example, Fœb *Content-boosted CF *hybrid CF combining memory-based and model-based *CF* algorithms, for example, Personality Diagnosis	*overcome limitations of CF and content-based or other recommenders *improve prediction performance *overcome CF problems such as sparsity and gray sheep	*have increased complexity and expense for implementation *need external information that usually not available

Table 1 - Overview of collaborative filtering techniques

Memory-based Collaborative Filtering

This mechanism uses user rating data to compute similarity between users or items. This is used for making recommendations. This was the earlier mechanism and is used in many commercial systems. It is easy to implement and is effective. Typical examples of this mechanism are neighborhood-based CF and item-based/user-based top-N recommendations [3]. For example, in user-based approaches, the value of ratings user *u* gives to item *i* is calculated as an aggregation of some similar users rating to the item:

$$r_{u,i} = \text{aggr}_{u' \in U} r_{u',i}$$

Equation 3 - value of ratings user u *gives to item* i

where *U* denotes the set of top *N* users that are most similar to user *u* who rated item *i*. Some examples of the aggregation function include:

$$r_{u,i} = \frac{1}{N} \sum_{u' \in U} r_{u',i}$$

Equation 4 - value of ratings user u *gives to item* i example 1

$$r_{u,i} = k \sum_{u' \in U} \text{simil}(u, u') r_{u',i}$$

Equation 5 value of ratings user u *gives to item* i example 2

$$r_{u,i} = \overline{r_u} + k \sum_{u' \in U} \text{simil}(u, u')(r_{u',i} - \overline{r_{u'}})$$

Equation 6 value of ratings user u *gives to item* i example 3

where *k* is a normalizing factor defined as

$$k = 1 / \sum_{u' \in U} |\text{simil}(u, u')|$$

Equation 7 normalizing factor

and $\bar{r}_i$ is the average rating of user *u* for all the items rated by that user.

The neighborhood-based algorithm calculates the similarity between two users or items, produces a prediction for the user taking the weighted average of all the ratings. Similarity computation between items or users is an important part of this approach. Multiple mechanisms such as Pearson correlation and vector cosine-based similarity are used for this.

The Pearson correlation similarity of two users *x, y* is defined as

$$\text{simil}(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \overline{r_x})(r_{y,i} - \overline{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \overline{r_x})^2 \sum_{i \in I_{xy}} (r_{y,i} - \overline{r_y})^2}}$$

Equation 8 - Pearson correlation similarity of two users x, y

where I_{xy} is the set of items rated by both user *x* and user *y*.

The cosine-based approach defines the cosine-similarity between two users x and y as:

$$\text{simil}(\text{x}, \text{y}) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \times \|\vec{y}\|} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x} r_{x,i}^2} \sqrt{\sum_{i \in I_y} r_{y,i}^2}}$$

Equation 9 - cosine-similarity between two users x and y

The user based top-N recommendation algorithm identifies the *k* most similar users to an active user using similarity-based vector model. After the *k* most similar users are found, their corresponding user-item matrices are aggregated to identify the set of items to be recommended. A popular method to find the similar users is the locality-sensitive hashing, which implements the nearest neighbor mechanism in linear time.

The advantages with this approach include: the explain ability of the results, which is an important aspect of recommendation systems; it is easy to create and use; new data can be added easily and incrementally; it need not consider the content of the items being recommended; and the mechanism scales well with co-rated items.

There are several disadvantages with this approach. Firstly, it depends on human ratings. Secondly, its performance decreases when data gets sparse, which is frequent with web related items. This prevents the scalability of this approach and has problems with large datasets. Although it can efficiently handle new users because it relies on a data structure, adding new items becomes more complicated since that representation usually relies on a specific vector space. That would require to include the new item and re-insert all the elements in the structure.

Model-based Collaborative Filtering

Models are developed using data mining, machine learning algorithms to find patterns based on training data. These are used to make predictions for real data. There are many model-based CF algorithms. These include Bayesian networks, clustering models, latent semantic models such as singular value decomposition, probabilistic latent semantic analysis, Multiple Multiplicative Factor, Latent Dirichlet allocation and markov decision process-based models. [15]

This approach has a more holistic goal to uncover latent factors that explain observed ratings. Most of the models are based on creating a classification or clustering technique to identify the user based on the test set. The number of the parameters can be reduced based on types of principal component analysis.

There are several advantages with this paradigm. It handles the sparsity better than memory-based ones. This helps with scalability with large data sets. It improves the prediction performance. It gives an intuitive rationale for the recommendations.

The disadvantages with this approach are in the expensive model building. One needs to have a tradeoff between prediction performance and scalability. One can lose useful information due to reduction models. A number of models have difficulty explaining the predictions.

Hybrid approaches

A number of applications combines the memory-based and the model-based CF algorithms. These overcome the limitations of native CF approaches. It improves the prediction performance. Importantly, it overcomes the CF problems such as sparsity and loss of information. However, they have increased complexity and are expensive to implement. Usually most of the commercial recommender systems are hybrid, for example, Google news recommender system. [12]

Disadvantages

A CF system does not necessarily succeed in automatically matching content to one's preferences. Unless the platform achieves unusually good diversity and independence of opinions, one point of view will always dominate another in a particular community. As in the personalized recommendation scenario, the introduction of new users or new items can cause the cold start problem, as there will be insufficient data on these new entries for the collaborative filtering to work accurately. In order to make appropriate recommendations for a new user, the system must first learn the user's preferences by analyzing past voting or rating activities. The CF system requires a substantial number of users to rate a new item before that item can be recommended.

Challenges of collaborative filtering

Synonyms: Synonyms refers to the tendency of a number of the same or very similar items to have different names or entries. Most recommender systems are unable to discover this latent association and thus treat these products differently.

For example, the seemingly different items “children movie” and “children film” are actually referring to the same item. Indeed, the degree of variability in descriptive term usage is greater than commonly suspected. The prevalence of synonyms decreases the recommendation performance of CF systems. Topic Modeling (like the Latent Dirichlet

Allocation technique) could solve this by grouping different words belonging to the same topic.

Grey sheep: Grey sheep refers to the users whose opinions do not consistently agree or disagree with any group of people and thus do not benefit from collaborative filtering. Black sheep are the opposite group whose idiosyncratic tastes make recommendations nearly impossible. Although this is a failure of the recommender system, non-electronic recommenders also have great problems in these cases, so black sheep is an acceptable failure.

Shilling attacks: In a recommendation system where everyone can give the ratings, people may give lots of positive ratings for their own items and negative ratings for their competitors. It is often necessary for the collaborative filtering systems to introduce precautions to discourage such kind of manipulations.

Diversity and the Long Tail: Collaborative filters are expected to increase diversity because they help us discover new products. Some algorithms, however, may unintentionally do the opposite. Because collaborative filters recommend products based on past sales or ratings, they cannot usually recommend products with limited historical data. This can create a rich-get-richer effect for popular products, akin to positive feedback. [12]

1.7 Hybrid Approaches

Recent research has demonstrated that a hybrid approach, combining collaborative filtering and content-based filtering could be more effective in some cases. Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combining them; by adding content-based capabilities to a collaborative-based approach (and vice versa); or by unifying the approaches into one model. Several studies empirically compare the performance of the hybrid with the pure collaborative and content-based methods and demonstrate that the hybrid methods can provide more accurate recommendations than pure approaches. These methods can also be used to overcome some of the common problems in recommender systems such as cold start and the sparsity problem.

Netflix is a good example of hybrid systems. They make recommendations by comparing the watching and searching habits of similar users (i.e. collaborative

filtering) as well as by offering movies that share characteristics with films that a user has rated highly (content-based filtering).
A hybrid recommender system is one that combines multiple techniques together to achieve some synergy between them.

Collaborative: The system generates recommendations using only information about rating profiles for different users. Collaborative systems locate peer users with a rating history similar to the current user and generate recommendations using this neighborhood. [16]

Content-based: The system generates recommendations from two sources: the features associated with products and the ratings that a user has given them. Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on product features.

Demographic: A demographic recommender provides recommendations based on a demographic profile of the user. Recommended products can be produced for different demographic niches, by combining the ratings of users in those niches.

Knowledge-based: A knowledge-based recommender suggests products based on inferences about a user's needs and preferences. This knowledge will sometimes contain explicit functional knowledge about how certain product features meet user needs.
The term hybrid recommender system is used here to describe any recommender system that combines multiple recommendation techniques together to produce its output. There is no reason why several different techniques of the same type could not be hybridized, for example, two different content-based recommenders could work together, and a number of projects have investigated this type of hybrid: NewsDude, which uses both naive Bayes and kNN classifiers in its news recommendations is just one example.

Seven hybridization techniques:

- Weighted: The score of different recommendation components are combined numerically.
- Switching: The system chooses among recommendation components and applies the selected one.
- Mixed: Recommendations from different recommenders are presented together.
- Feature Combination: Features derived from different knowledge sources are combined together and given to a single recommendation algorithm.
- Feature Augmentation: One recommendation technique is used to compute a feature or set of features, which is then part of the input to the next technique.
- Cascade: Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones.
- Meta-level: One recommendation technique is applied and produces some sort of model, which is then the input used by the next technique.

Chapter 2: Presentation of Recommender Systems

Contents

This section presents some representative recommender systems, collaborative, thematic and hybrid. The dates in parentheses refer to the years of publication of reference articles, not necessarily the date of implementation in a case of an operational system.

2.1 Tapestry (1992): The Precursor

Tapestry, is often presented as the first recommendation system to be called collaborative. It is based on the work of Goldberg [17]. Tapestry allowed a flow of electronic documents to be recommended, to a user group at Xerox PARC. It was actually a hybrid system in the sense that access to documents may be made by the content of the documents but also on the basis of annotations made by other users on these same documents. The system consisted of several modules including: a database of documents and annotations on these documents, a system to read, navigate and annotate documents, the filtering system based on a query language similar to SQL. The requests and the associated language were a major limitation of the system which made it difficult to use. Users had to create queries to retrieve documents. These queries were based on documents or annotations. The requests were then stored and periodically re-executed. It was possible to formulate queries like "give me messages talking about New York and annotated at least 20 times in the past 2 weeks". More than the collaborative aspect, it is the hybrid aspect of Tapestry (access to content data and usage data on their contents) which was the most significant. For the collaborative feature, in fact, the system did not offer an automatic process to analyze the usages performed: users had to explicitly query the uses of other users. It is more a precursor of the automatic recommenders than an automatic recommender. [18]

2.2 GroupLens (1994): The Collaborative Filtering Approach

GroupLens made recommendations of newsgroup messages and was developed at the University of Minnesota [19]. GroupLens worked on a principle similar to Tapestry: a group of users annotate documents, in a simplified way: First, annotations have been reduced to a rating of interest for each document. Second, the query language, the main difficulty of Tapestry, was replaced by an automatic score prediction feature, calculated from the correlation of ratings from users. [20]

GroupLens validated the principle of collaborative filtering on a large scale, but also confirmed its weaknesses such as the cold start problem: the system was not capable of making good predictions before a certain critical mass of users. This could deter early

users to become involved in the system. GroupLens was the first article introducing the term "collaborative filtering". The GroupLens system was based on a method of K-nearest neighbors on users. The Pearson coefficient was used as the measure of similarity between the users.

GroupLens is often considered as the founder of the collaborative filtering approach.

2.3 MovieLens (1997)

A reference website and a reference database MovieLens and MovieLens Unplugged are recommender systems from GroupLens, a research team from the Department of Computer Science of the University of Minnesota. The team has applied its collaborative filtering techniques to movie recommendation. MovieLens is a Web service of movie recommendation for scientific research. It is based on the oldest works of GroupLens. We can register on this site via a pseudonym. We are then invited to rate a number of films from a broad catalog. MovieLens asked, for example to rate at least 20 movies before making recommendations. MovieLens' logs of ratings are publicly available on Internet. Long before the 2006-2009 Netflix's challenge, the MovieLens

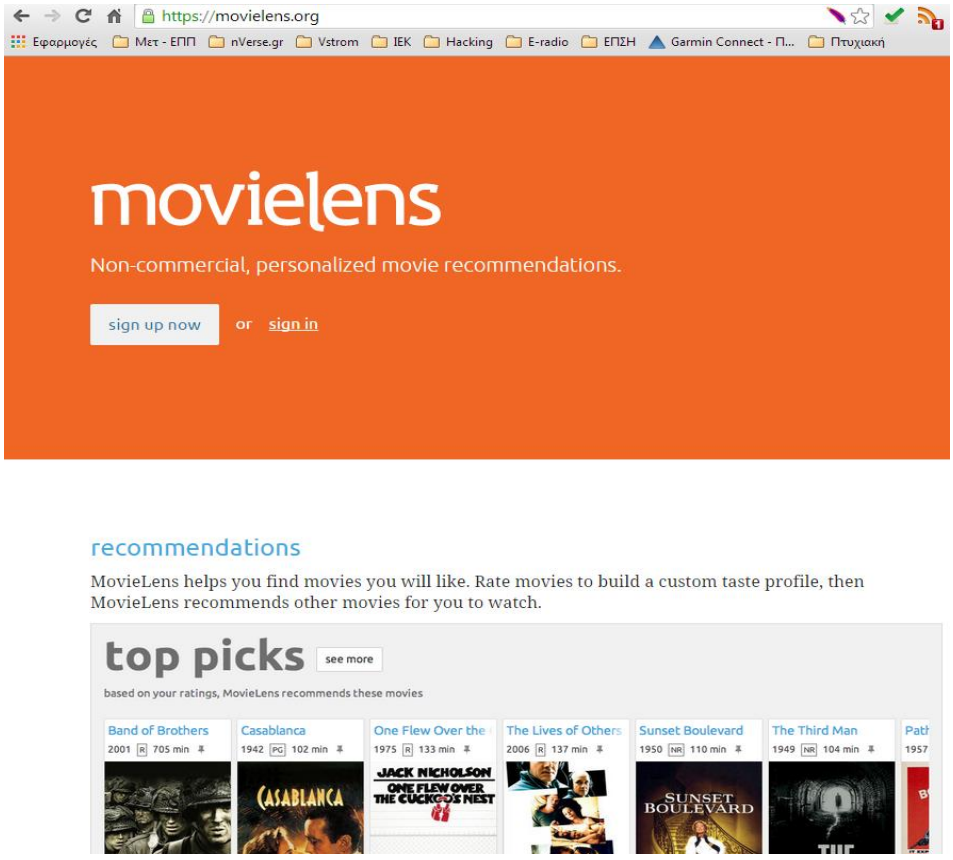

Figure 3. MovieLens is still available at: http://movielens.umn.edu/

database was the main source of experimental data to study the algorithms for collaborative filtering.

2.4 Amazon.com (2003)

Scalable item-to-item collaborative filtering recommendation, Amazon, is an example of successful recommendation engine's technology. This technology is the basis of the marketing strategy of the website. The main function used is based collaborative item-to-item contextual recommendations, this was introduced very early on the site (late 90's). It is based on the logs of purchases and corresponds to the calculation of a similarity matrix of items with an optimized algorithm to scale with the imposing volumes handled by Amazon. Amazon popularized the famous feature "people who bought this item also bought these items". [21]

An example of the interface provided by Amazon for item-to-item contextual recommendation is given below. Amazon may now be classified as a hybrid system as a recommendation features based on some metadata (genre, author) also works in push mode with personalized web page and e-mailing.

Figure 4. Amazon.com's item-to-item contextual recommendation to anonymous user

Amazon's main innovations are:

- To have designed a high scalability recommender system

- To have built its entire marketing strategy on a recommender engine, intelligently integrating the recommendation functions into its website.

2.5 MORE (2006): Hybrid Recommendation by Switch

MORE [22] is a movie recommendation system prototype, using a switch-based hybridization. The switch mode between a collaborative and thematic system uses the size of the user's profile. MORE was only evaluated by simulation. It was evaluated on the MovieLens database (1 million ratings, ~6000 users, ~3700 movies). An acquisition of thematic information was made on the IMDd database (International Movie Database, www.imdb.com): the information of genre, actors, director and keywords were joined to the titles of movies. MORE uses a K-nearest neighbor model, applied on users in the collaborative system, or applied on items in the content-based system. In the case of the collaborative system, the chosen similarity function is the Pearson coefficient applied to the vector of ratings of users, and users can be considered neighbors of a target user u if and only if their similarity with u is positive. In the case of the thematic system, the similarity between items is the cosine of the vectors of characteristics of the items. Two types of hybridization were tested.

The system works in a collaborative mode by default because this method is considered the most efficient. In the first type of hybridization, called the "substitute" hybrid method, the system switches to the thematic filtering when a user has less than K=5 reliable neighbors (according to a similarity threshold). In the second type of hybridization, called " switching" hybrid method, the system switches to the thematic filtering if a user has less than S=40 ratings (threshold chosen empirically).

The authors have not done extensive testing with different parameters K and S but noted that with the tested values the substitute hybrid method increased both the predictive performance (MAE) and overall coverage. Coverage is the percentage of items in the catalog that can be analyzed to return a score. The substitute and switching hybridization modes are the main particularities of MORE.

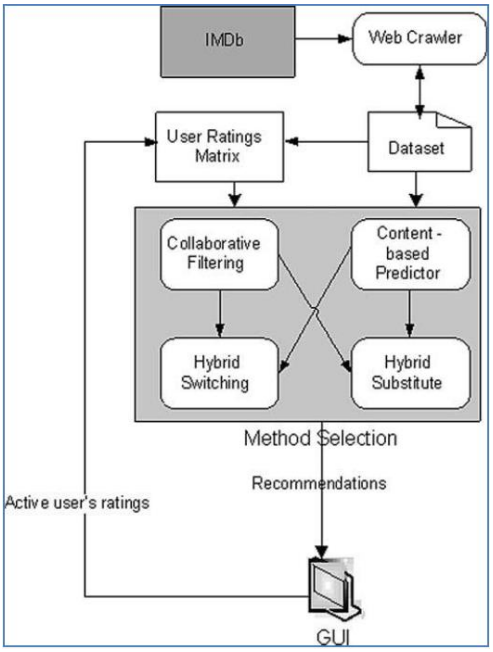

Figure 5. Architecture of MORE

(MAE) and overall coverage. Coverage is the percentage of items in the catalog that can be analyzed to return a score.

2.6 YouTube (2010)

YouTube (www.youtube.com) is a famous online video website community. The recommendation system of YouTube [23] provides contextual and personalized recommendations. The system makes recommendations but does not score the items: This is a Top-N-Recommendation, if one chose a typology based on functionality. The YouTube's data typically represents a high scalability issue: it has more than one billion videos, most without metadata, and users are multi-million daily. YouTube's recommender system is based on association rules for short periods of navigation of each user (usually 24 hours).

The very high scalability on very sparse data, and the optimization techniques to address these issues, are the main particularities of YouTube's recommender.

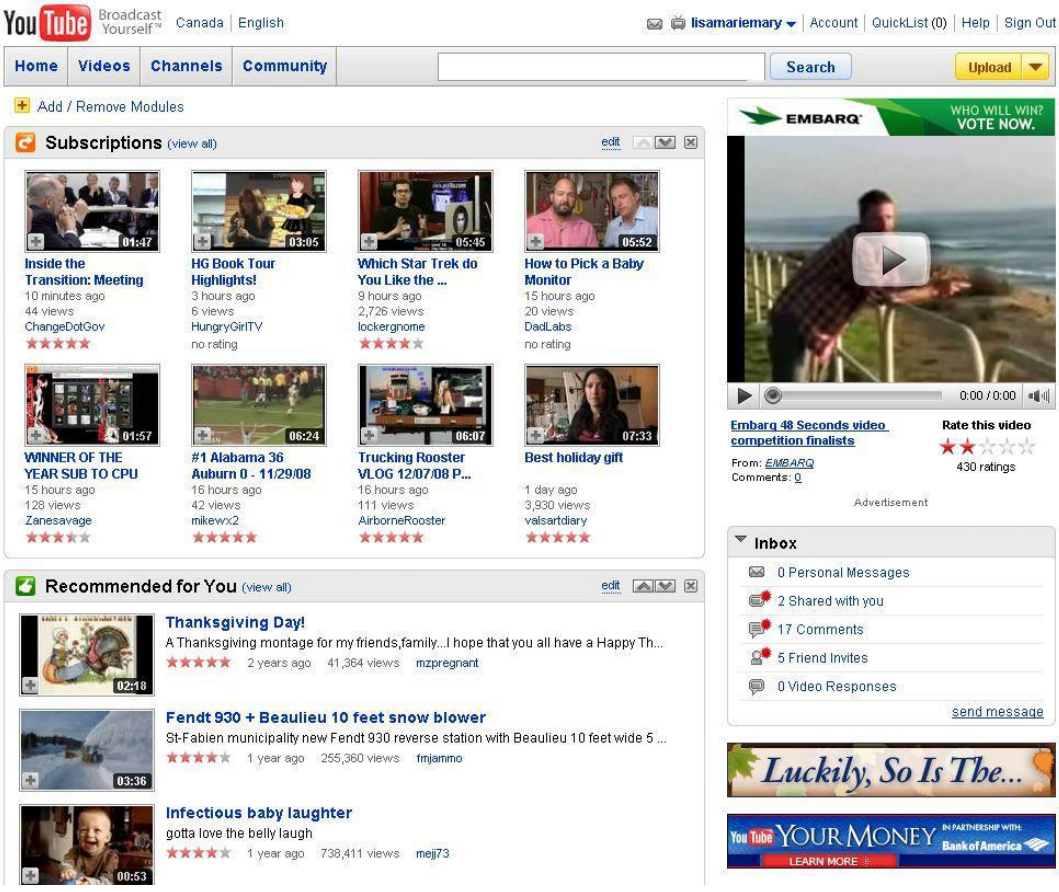

Figure 6. Personalized recommendation with YouTube

2.7 Recommendation on Twitter: Twittomender (2010)

Twitter is a recent real time service on the Internet, enabling a community of users to post and follow short text messages based on SMS and limited to 140 characters. A message issued by a user on Twitter is called a tweet.

Twitter is considered as a real time micro-blogging social network using personal textual content generation. Users can explicitly designate the other users they wish to listen to tweets.

The proposed recommendation system [24] is called Twittomender and uses the Twitter API to access to user data. Twittomender recommends to users of Twitter other users to follow, therefore followees, based on their tweets and the social graph in Twitter. The system uses both content analysis of the tweets and collaborative analysis.

Twittomender performs a profiling of each user taking into account his tweets, its followees and its followers. The recommendation system is based on an open source search engine, Lucene, that indexes the tweets. Tweets, as users replace traditional documents in this framework.

The main innovations of Twittomender are:

- Carrying out the recommendation in the context of social networks for instant communication (micro-blogging),
- Using a search engine as support to the recommendation, for thematic and collaborative profiling, using document terms, itemIDs and userIDs at the same time.

2.8 Real Applications

Thanks to the ever-decreasing costs of data storage and processing, recommender systems gradually spread to most areas of our lives. Sellers carefully watch our purchases to recommend us other goods and enhance their sales, social web sites analyze our contacts to help us connect with new friends and get hooked with the site, and online radio stations remember skipped songs to serve us better in the future. In general, whenever there is plenty of diverse products and customers are not alike, personalized recommendation may help to deliver the right content to the right person. A recommender system may hence have significant impact on a company's revenues: for example, 60% of DVDs rented by Netflix are selected based on personalized recommendations. Recommender systems not only help decide which products to offer to an individual customer, they also increase cross-sell by suggesting additional

products to the customers and improve consumer loyalty because consumers tend to return to the sites that best serve their needs.

Since no recommendation method serves best all customers, major sites are usually equipped with several distinct recommendation techniques ranging from simple popularity-based recommendations to sophisticated techniques many of which we shall encounter in the following sections. Further, new companies emerge (see, for example, string.com) which aim at collecting all sorts of user behavior (ranging from pages visited on the web and music listened on a personal player to ‘‘liking’’ or purchasing items) and using it to provide personalized recommendations of different goods or services. [10]

Site	What is recommended
Amazon	Books/other products
Facebook	Friends
WeFollow	Friends
MovieLens	Movies
Nanocrowd	Movies
Jinni	Movies
Findory	News
Digg	News
Zite	News
Meehive	News
Netflix	DVDs
CDNOW	CDs/DVDs
eHarmony	Dates
Chemistry	Dates
True.com	Dates
Perfectmatch	Dates
CareerBuilder	Jobs
Monster	Jobs
Pandora	Music
Mufin	Music
StumbleUpon	Web sites

Table 2 - Recommendations per website

2.9 SCoR: A Novel Recommender System

Based on the Vivaldi [25] algorithm, SCoR [26] is a Synthetic Coordinate based Recommender System, that assigns synthetic coordinates to nodes (users and items), so that the distance between a user and an item provides an accurate prediction of the user’s preference for that item.

The goal of the algorithm is to calculate unknown recommendations that accurately predict the preference of user *u* for item *i*, when *u* has not yet rated *i*, based on the indirect knowledge of existing user-item ratings. The results of the algorithm’s

evaluation, shows that SCoR outperforms the most popular algorithmic techniques in the field, approaches like matrix factorization and collaborative filtering.

Beside the evaluation results and the recommendation system proposed, SCoR can give extra annotations of the user and item datasets based on an analysis of the nodes positions in the *n*-dimensional Euclidean space. SCoR is able to provide the sets of users with similar preferences to each other and the sets of items with similar ratings from users. Furthermore, SCoR is a parameter free, thus requiring no fine tuning to achieve high performance, and is more resistance to the cold-start problem compared to other algorithms.

The detected problem of the SCoR algorithm is that the accuracy is reduced in cases of very sparse datasets (i.e. very few users with very few ratings), which is also a problem for the Vivaldi algorithm.

The algorithm

As mentioned above, the SCoR algorithm is an updated version of Vivaldi [25] algorithm. All users and items are placed in a single, multi-dimensional Euclidean space. The Vivaldi synthetic coordinates algorithm is then employed to properly place them in points in space so that the distance between any user-item pair is directly respective to the preference of that user for the specific item. Indirect relationships between users who have rated the same item, and items rated by the same user are them accurately reflected. The Euclidean distance between a user and an item of unknown rating (for that user) is used to make the prediction (recommendation).

Starting with the input of:

a) a set of users $u \in$ the set $U = \{1, 2, ..., N\}$ of unique identifiers of users which have rated items.
b) a set of items $i \in$ the set $I = \{1, 2, ..., M\}$ of unique identifiers of items that have been rated by users.
c) the ratings $r(u, i) \in R$ denotes the rating (declared preference) of user u for item i.

The first modification of the Vivaldi algorithm is that network nodes are replaced by user nodes and item nodes, while latencies are replaced by distances calculated by the ratings of users for items. As a result, a bipartite graph is formed, consisting of users nodes on one side and items nodes on the other (instead of just network nodes). For each $(u,i, r(u,i))$ triplet, an edge is added between nodes u and i. Each edge is assigned

a weight *dd(u, i)*, based on rating *r(u, i)*, which reflects the distance of edge *(u, i)*. A *(u, i)* pair with high *r* value (high preference of user *u* for item *i*) corresponds to an edge with small distance and vice versa.

The smallest rating, *minR*, is assigned a distance of 100, whereas the highest rating *maxR* is assigned a distance of 0. Given these values, the distance *dd(u, i)* is calculated as follows:

$$\mathrm{dd}(u,i) = 100 \cdot \left(\frac{maxR - r(u,i)}{maxR - minR} \right)$$

Equation 10 - the distance dd(u, i)

where *minR*, *maxR* denote the minimum (low preference) and maximum (high preference) ratings, respectively. Thus, latency between network nodes in Vivaldi, is replaced with the distance calculated by the declared rating between the user and the item according to the above equation. Vivaldi then iteratively updates the positions of the nodes to satisfy the desired distances for all edges until each node's position stabilizes.

Ideally, if a user *u* has rated item *i* with value *r(u, i)*, then after Vivaldi has converged, the distance of nodes *u* and *i* should be *dd(u, i)*, calculated according to the above equation.

The second modification of the Vivaldi algorithm is to allow weighted neighbor selection, for user, item pairs. As algorithm progresses, not all user-item pairs are equally successful in achieving the desired distance. In order to help less successful pairs, improve their performance, we make a second execution of the algorithm, this time relying more heavily on user- item links that better achieved their desired distance. To implement this step, after Vivaldi converges, they assign a weight to each neighbor, based on the observed error between their desired distance *dd(u, i)* and the actual distance *d(u, i)* and the Mean Square Error of the node *MSE(u)*.

Dataset annotations using SCoR

Beside from providing a recommendation system, SCoR is able to provide annotations of the user and item datasets based on an analysis of the nodes positions in the *n*-dimensional Euclidean space. It is able to provide the sets of users with similar preferences to each other and the sets of items with similar ratings from users.

Using SCoR, the Euclidean distance $d(u_1, u_2)$ between two user nodes u_1, u_2 is directly related to the absolute difference in their recommendations. When the algorithm converges, it holds that the lower the distance of two nodes in the *n*-dimensional Euclidean space, the smaller their recommendation difference. The distance between two nodes is equal to the corresponding maximum value of their recommendation difference. If the distance between two nodes is low, it means that users u_1, u_2 have similar preferences for any item. Similar analysis can be applied to items. The lower the distance between two items, the lower the difference between their ratings.

Experimental Results

The SCoR framework is evaluated with well-known, real world datasets in our experiments.

- Netflix prize dataset (smallnetflix) [27]
- MovieLens dataset (ml)
- Jester dataset [28]
- Jester2 dataset [28]

Each dataset is comprised of two files. A training set which includes the "known" ratings and a validation dataset which is the ground truth against which we test the predictions of the algorithms.

To evaluate its performance SCoR is compared with recent recommender algorithms:

1. ALS [29]
2. ALS_COORD [30]
3. BIASSGD [31]
4. BI- ASSGD2 [31]
5. RBM [32]
6. SGD [33]
7. SVDPP [31]
8. P_MEAN [34]
9. USER-USER algorithm [35]

The results of the algorithm evaluation, shows that SCoR outperforms the most popular algorithmic techniques in the field, approaches like matrix factorization and collaborative filtering.

Dataset:	**smallnetflix**	**ml**	**jester**	**jester2**
ALS	1.16	0.964	0.943	1.38
ALS_COORD	1.14	0.932	0.917	1.30
BIASSGD	0.958	0.897	0.872	0.909
BIASSGD2	0.967	0.888	0.861	0.923
SCoR	**0.940**	**0.875**	**0.854**	**0.894**
RBM	0.941	0.900	0.880	0.912
SGD	0.961	0.898	0.872	0.906
SVDPP	0.989	0.944	0910	0.953
P-MEAN	0.950	0.913	0.886	1.025
USER-USER	0.96	0.905	0.869	1.072

Table 3 - RMSE of the ten recommender systems for the four datasets

Figure 7 presents the performance of the ten recommender systems for the four datasets. According to this table, SCoR gives the highest performance for all datasets. This table summarizes the main results of the experiments.

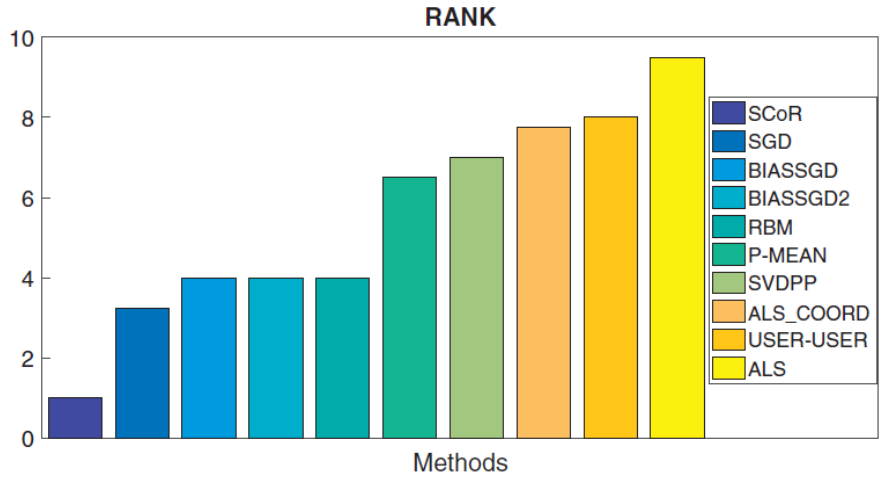

Figure 7. The RANK of the ten systems computed for the four datasets

The computation of the RANK [36] metric per algorithm, which is defined as the average order in performance of each system over the four datasets, can be used as a metric to summarize the algorithms' performance.

$$RANK(s) = \frac{\sum_{dat \in Datasets} Order(s, dat)}{|Datasets|}$$

where *Order(s, dat)* is the order in performance of system *s* on the dataset *dat*, e.g. *Datasets = {small netflix, ml, jester, jester2}* and |*Datasets*| is the number of datasets.

According to the *RANK* definition it holds that the lower the *RANK*, the better the algorithm (Haindl & Mikeš, 2016). According to this experiment SCoR belongs in the first class (top performance system), since it clearly outperforms the rest of the systems with *RANK = 1*.

Chapter 3: Community Detection

Contents

3.1 Definition of Community Detection

Several different definitions of what is a community exist. However, two of them are the most popular among the academic community.

1. A strong community is defined as a group of nodes for which each node has more edges to nodes of the same community than to nodes outside the community. This definition is relatively strict, since it does not allow for overlapping communities and creates a hierarchical community structure, since the entire graph can be a community itself.
2. A generalized community, is defined as a subgraph in which the sum of all node degrees within the community is larger than the sum of all node degrees towards the rest of the graph. [37]

Other definitions are related to the optimization of some ``fitness" criterion like for example the intercommunity edge density, or the intercommunity edge cut. In general, community detection is the problem of finding a partition of a graph into subgraphs that maximizes some quality criterion which reflects the density of the subgraph(s). [38]

3.2 Community Detection in Social Networks

Social Network Analysis is the methodical analysis of social networks. In order to study efficiently a social network for the purpose of understanding its dynamics, a simplified and abstract model is required. A popular modeling method is to represent a network's objects and their relationships by a graph, allowing to use a large set of generic methods provided by the field of Discrete Mathematics. [39]

In this context, individual actors within the network are represented by nodes and the relationships formed between them such as friendship or organizational position are represented by edges. The use of this specific modeling tool was considered a fail-safe choice as the technique had already been successfully applied to a wide range of scientific disciplines including mathematics, physics, biology social sciences and criminology. [40]

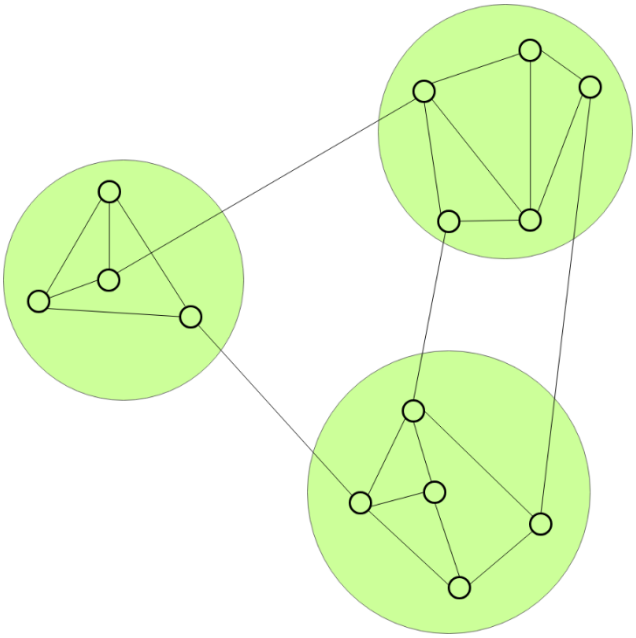

Figure 8. The community structure of a simple graph consisting of three communities

Basic Network Definitions

Before we get our hands on the basic problems on social networks we must first focus on a series of important preliminaries that concern Graph Theory, including some basic definitions. In mathematics and computer science, graphs represent mathematical structures used to model pairwise relations between objects from a certain collection. The interconnected objects are represented by mathematical abstractions called vertices or nodes which are connected by edges. [41]

Graphs can be implemented in a visual level by a set of dots or circles for the vertices, joined by lines that simulate the relations between them. What makes Graphs such a powerful mathematical tool is the flexibility they present. Graphs have the ability to adapt to the nature of the represented data. This ability to cope with so diverse datasets is based on a variety of different Graph definitions. [42]

Non-Directed Network

A network whose edges have no orientation is referred as a Non-Directed Network. The edge that connects node *x* to node *y* is exactly the same with the edge that connects node *y* to node x. It is the simplest and most common type of networks.

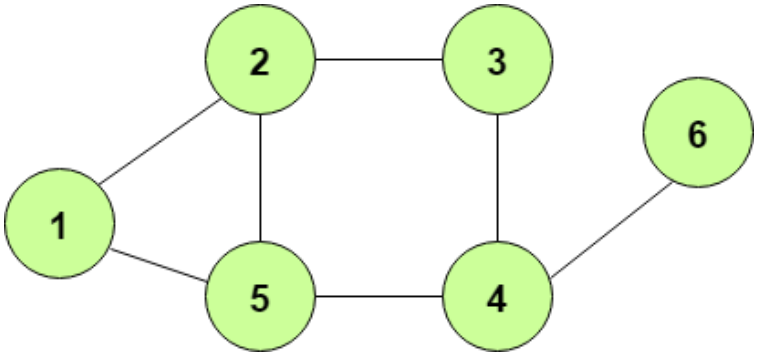

Figure 9. A labeled graph with six vertices and sever edges

Directed Network

A directed network or digraph is an ordered pair *D = (V, A)*. V refers to the set of nodes while A refers to a set of ordered pairs of nodes called arcs. An arc *a = (x, y)* is considered to be directed from node *x* to node *y*. *y* is called the head or direct successor, and *x* is called the tail or direct predecessor of the arc. If a path leads from vertex *x* to vertex *y* via one or more other nodes, *y* is considered to be successor or reachable from *x*. A directed graph as described above, is considered to be symmetric if for every arc in *D*, the corresponding inverted arc also exists in *D*. A symmetric loop-less directed graph *D = (V, A)* is equivalent to a simple undirected graph *G = (V, E)*, where the pairs of inverse arcs in *A* correspond one-to-one with the edges in *E*. As a consequence, the edges in *D* are twice as many as the edges in *G*.

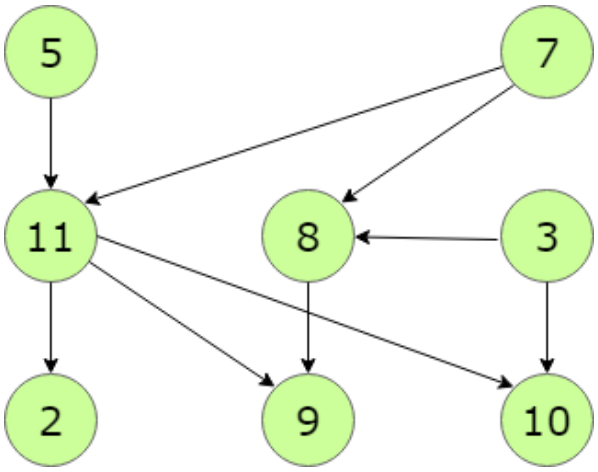

Figure 10. A simple directed acyclic graph.

A stricter variation of this definition is called oriented network. The term refers to a network in which no more than one of *(x, y)* and *(y, x)* may exist simultaneously.

$$\sum_{v \in V} deg^{+}(v) = \sum_{v \in V} deg^{-}(v) = |A|$$

Figure 11 - Oriented network

For a node, the number of head endpoints adjacent to a node is called the in-degree of the node and the number of tail endpoints is its out-degree. The in-degree is denoted $deg^{-}(v)$ and the out-degree as $deg^{+}(v)$. A vertex with $deg^{-}(v) = 0$ is called a source, as it is the origin of each of its incident edges. Similarly, a vertex with $deg^{+}(v) = 0$ is called a sink.

If for every node v in V $deg^{+}(v) = deg^{-}(v)$, the graph is called a balanced digraph. Regarding its connectivity, a digraph G is called weakly connected if the undirected underlying graph obtained by replacing all directed edges of G with undirected edges is a connected graph. A digraph is strongly connected if it contains a directed path from u to v and a directed path from v to u for every pair of vertices u, v. The strong components are the maximal strongly connected sub-graphs.

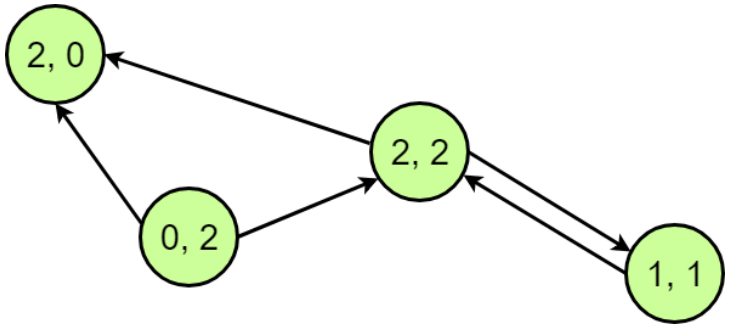

Figure 12. A digraph with vertices labeled (indegree, outdegree)

Mixed Network

A mixed network G is graph in which some edges may be directed and some other may not be directed. It can be expressed through an ordered triple $G = (V, A, E)$ with V, A and E are as defined earlier.

Multinetwork

In discrete mathematics, the term multinetwork refers to a graph that allows the existence of multiple edges. Edges are considered to be multiple or parallel, if they have the same tail and head creating multiple direct paths between these nodes. Some authors also allow multigraphs to have loops, edges that connects a vertex to itself, while others call these pseudo-graphs, reserving the term multigraph for the case with no loops. This definition can be combined with these of directed or undirected graphs resulting in a hybrid graph. For example in flight booking

systems, multigraphs might be used to model the possible flight connections offered by an airline.

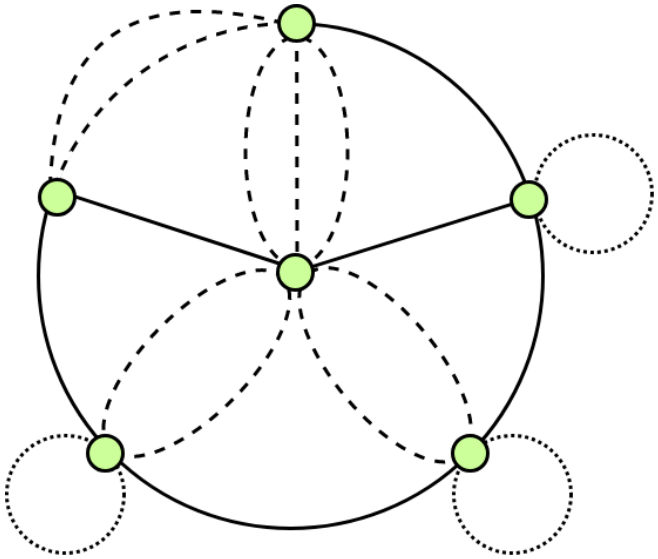

Figure 13. A multigraph with multiple edges (dashed) and several loops (dotted)

Weighted Network

Certain applications demand more information to be represented by the graph. In this case a real or a rational number which is called weight or cost can be assigned to each edge in order to represent, for example, costs, lengths or capacities, etc. depending on the problem at hand. Although some authors use the term network as a synonym for a weighted graph, unless stated otherwise, a graph is always assumed to be unweighted. Classic problems connected with weighted graphs include:

- Minimum spanning tree
- Shortest-path problem
- Max-flow/min-cut theorem

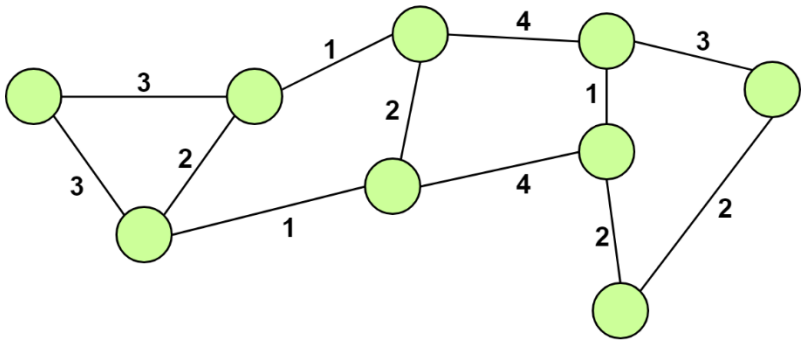

Figure 14. A simple weighted network.

Regular Network

In graph theory, the term regular graph is used to describe a graph where each node has the same degree, in other words the same number of neighbors. In case we are dealing with directed graphs, another restriction must also be satisfied. The in-degree and out-degree for every vertex has to be equal. A regular graph with nodes of *k*-degree can also be referred as a *k*-regular graph.

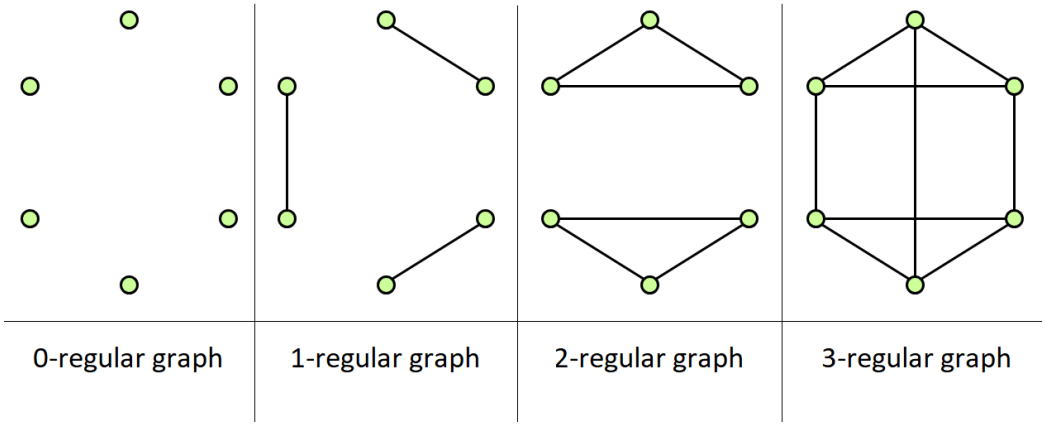

Figure 15. A regular graph

Complete Network

The term "complete", sometimes stated as "fully connected", defines a simple undirected graph in which all pairs of two distinct vertices are connected by a unique edge. The complete graph on n vertices has $n(n - 1)/2$ edges, and is denoted by K_n. Another way to see a complete graph is as a $(k-1)$ regular graph. One very interesting ascertainment on complete graphs is that the only way to disconnect the graph is by removing the complete set of vertices. The complement graph of a fully connected graph is an empty graph. Essentially, complete graphs form their own maximal cliques.

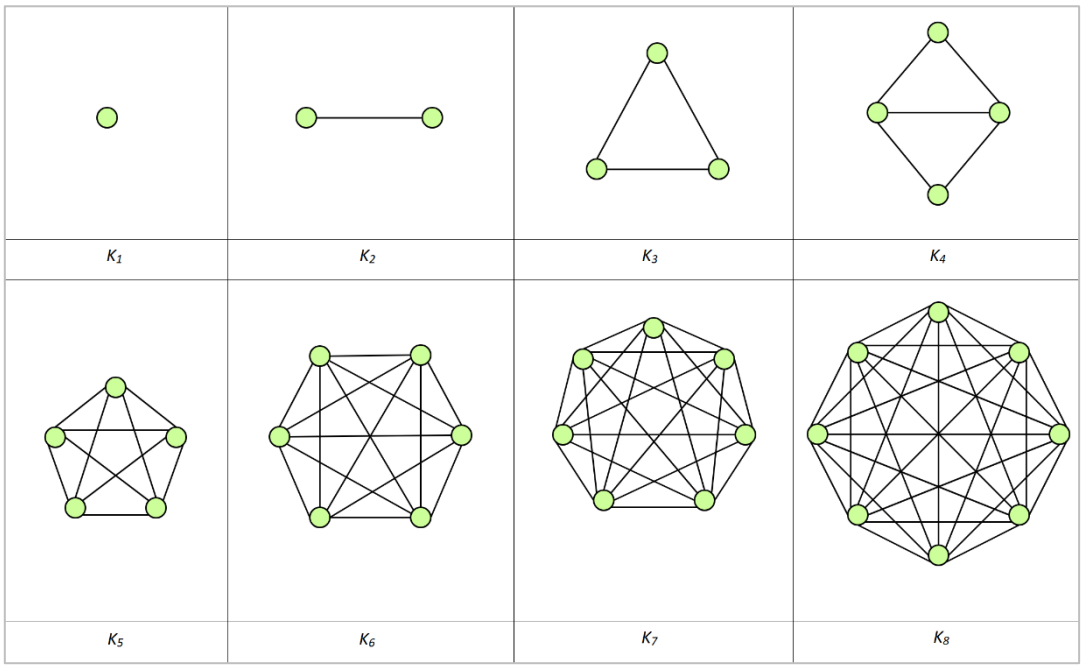

Figure 16. Table of basic complete networks with K<=8

Walk

A walk from node *x* to node *y* is any sequence of adjacent edges that begins with an edge containing *x* and ends with an edge containing node *y*. Nodes and edges can be visited several times in a walk, as long as these constraints are not violated.

Trail

On the other hand, a trail is defined as a walk where the sequence of adjacent edges does not contain any edge for more than one times. Though, the same constraint does not apply for nodes, so a node can be visited through different edges numerous times.

Path

Path is defined as a walk in which neither edges nor nodes are visited more than once. If the starting node coincides with the destination node then this path is considered to be a Eulerian path.

Distance

In network theory, the shortest-path problem is the problem of finding a path between two nodes in a graph such that the sum of the weights of its constituent edges is minimized which is analogous to the problem of finding the shortest path between two intersections on a road map. In this context the graph's vertices correspond to intersections and the edges correspond

to road segments, each weighted by the length of its road segment. In a non-weighted network, the length of a path corresponds to the minimum hops required in order to reach node *y* starting from node *x*. On the other hand, in a weighted network, the length of a path is processed by summing the weights of the edges it includes, resulting in a slightly different definition of the distance. When a node *y* is not reachable from a node *x*, their distance is theoretically infinite. However, in many situations, this infinite value causes problems, and this distance is then considered to be zero. The most important algorithms for solving this problem are:

- Dijkstra's algorithm
- Bellman-ford algorithm
- A* search algorithm
- Floyd-Warshall algorithm
- Johnson's algorithm
- Perturbation theory

Node Sets

Dyad and Triad

Dyad is the smallest set of nodes. It consists of two nodes which can be linked together using none, one or multiple edges. Triad on the other hand, as its name betokens concerns group of nodes containing three nodes.

Triangle

Triangle's cornerstone is the triplet. A triplet consists of three nodes that are connected by either two (open triplet) or three (closed triplet) non-directed edges. A triangle consists of three closed triplets, one centered on each of the nodes.

K-Clique

K-Clique corresponds to a maximally connected subset of nodes. Each node must be connected to every other node that participates into the clique. Its size corresponds to the number of nodes it contains.

Component

A component is a sub-network in which any node is reachable from any other one by a walk. Put concisely, it is a maximal connected subgraph. For an undirected network, a component is a set of connected nodes with no links with other nodes from the same network. But for directed networks, it is less straightforward. A component is said to be strongly connected if there is a

directed walk between each pair of nodes. It is called weakly connected if there is at least an undirected walk between each pair of nodes. A network with only one component is said to be connected. An isolated node (i.e. a node with a degree zero) is a component of its own. [43]

3.2 Networks Properties

Homophily and Heterophily

Homophily refers to the tendency for nodes to connect to other vertices with which they share (respectively don't share) common attributes and characteristics. In other words, homophily seems to be responsible for emphasizing or even causing the relationships between the nodes. For example, in a social network where edges represent friendship connections, it is more likely that a user is connected to an equally young person or a user with who shares common interests. In literature this property can be also found as assortative mixing. [44]

Clustering Coefficient

In network theory clustering coefficient is representing the extent to which nodes of a specific network tend to shape clusters. Evidence suggests that in most real-world networks, and in particular social networks, nodes tend to create tightly connected groups characterized by a relatively high density of ties. Impressively, in real-world networks, this likelihood tends to be greater than the average probability of a tie randomly established between two random vertices [45].

Two versions of this metric exist in the literature, depending on their perspective. The global perspective definition was designed to give an overall indication of the clustering in the network, whereas the measure that follows a local approach gives an indication of the embeddedness of single vertices. The global clustering coefficient is based on triplets of nodes and is defined as the number of closed triplets over the total number of triplets. This measure gives us an indication of the clustering in the whole network and can be applied to both non-directed and directed networks. [21]

Small World Property

Small world property was first introduced by Stanley Milgram and a group of other researchers in an effort to examine the average path lengths for social networks of people living in the United States. The research proved to be groundbreaking suggesting that human society network is characterized by short path lengths. This characteristic is often associated with the idea of “the six degrees of separation”. [46]

According to the idea everyone is on average approximately six steps away, by way of introduction, from any other person in the world, so that a chain of "a friend of a friend" statements can be made, on average, to connect any two people in six steps or fewer. As far as mathematics are concerned, a small-world network is a graph in which most nodes are not neighbors of one another, but most nodes can be reached from every other with a relatively low cost. More specifically, a small-world network is defined to be a network where the typical distance L between two randomly chosen nodes (the number of hops required) grows proportionally to the logarithm of the number of nodes N in the network, that is:

$$L \; x \; log \; N$$

Network Resilience

The term signifies the extent to which a network tends to maintain its topological properties when changes occur on its structure. By removing a series of edges or nodes, network's connectivity is reduced having an important impact on information flow which, when passing a critical point can lead to bottlenecks. [47]

Chapter 4: Community Detection Algorithms

Contents

4.1 Community Detection Algorithms

Variations appear in the methodology used to identify communities. Certain algorithms follow an iterative approach starting by characterizing either the entire network, or each individual node as community, and splitting or merging communities, respectively. These methods produce a hierarchy of nested communities. By merging or splitting communities, one can build a hierarchical tree of community partitions called dendrogram. Several researchers aim to find the entire hierarchical community dendrogram, while others try to identify only the optimal community partition. [48] More recent approaches aim to identify the community surrounding one or more seed nodes. Some researchers aim to discover distinct (non-overlapping) communities, while others allow for overlaps. [49]

Newman's Algorithm

One of the well-known communities finding algorithms was developed by Girvan Newman [50]. This algorithm follows a hierarchical approach based on which communities are detected by removing edges iteratively from the graph. An edge that belongs to many shortest paths between nodes has high betweenness and has to be removed, because it is more likely to be an inter-community edge. By removing gradually edges, the graph is split and its hierarchical community structure is revealed. [50]

The algorithm is computationally intensive, because following the removal of an edge, the shortest paths between all pairs of nodes have to be recalculated. It reveals not only individual communities, but the entire hierarchical community dendrogram of the graph. An important element of the algorithm is the modularity calculation, which is used to evaluate the quality of a community partition resulting and also as a termination criterion for the algorithm.

Although there is a wide range of betweenness measures available, shortest-path betweenness is used due to the lower computational cost and the satisfactory results. Shortest-path betweenness can be calculated by finding the shortest paths between all pairs of vertices and summing up how many of those run along each edge. Experimental results for implementations of the algorithm based on different betweenness metrics have shown no significant impact on the quality of the community structure output.

The general form of Newman’s algorithm is as follows:

1. Calculate betweenness scores for all edges of the network
2. Find the edge with the highest betweenness value and remove it from the network

3. Recalculate betweenness for all remaining edges
4. Repeat from step (2)

The output of this process can be represented as a dendrogram depicting the successive splits of the network. The detection process can be stopped at any point the output community structure is judged to be satisfactory. An accurate way to determine if a network division is satisfactory is to use an appropriate evaluation metric. A metric that can carry out this task is the modularity metric. Modularity essentially indicates the extent to which a given community partition is characterized by high number of intra- community edges compared to inter-community ones. Essentially, the algorithm proceeds as long as network partitions with higher modularity are produced after edge removals. An appropriate modularity threshold is applied in order to identity the optimal community structure and the algorithm to terminate.

CiBC

Compared to other community detection algorithms applied in various scientific fields, CiBC was designed to fulfill the very specific task of identifying Web communities from a web server content, in order to improve the performance of CDNs.

CDN stands for Content Delivery Network [51]. A content delivery network (CDN) is a large distributed system of servers deployed in multiple data centers all over the world. The main purpose of a CDN is to provide web content to end-users with high availability and high performance. CDNs serve a large fraction of the Internet content today, including web objects (text, graphics, URLs and scripts), downloadable objects (media files, software, documents), applications (e-commerce, portals), live streaming media, on-demand streaming media, and social networks more importantly social networks and web 2.0 applications. [52]

The algorithm is based on a slightly different definition from what is traditionally considered as a network community. Usually, a community is defined as a subgraph for which each node has more edges to nodes of the same community than to nodes outside the community (strong community). A more flexible definition, called generalized community, is the following: a community is a subgraph in which the sum of all node degrees within the community is larger than the sum of all node degrees towards the rest of the graph. Apart from the capability of identifying overlapping communities, CiBC has one more innovative characteristic, its hybrid nature, using both local and global graph's properties in order to accomplish its mission. Community detection is performed in three phases, with each phase including further steps. [53]

In the first phase, the Betweeness Centrality (BC) is calculated for each node of the graph. BC is a metric used to measure how "central" a node is in the graph. Last step before proceeding with phase two includes the sorting of the nodes of the graph by ascending BC value.

The second phase concentrates on the initialization of the cliques. This is achieved using an iterative procedure starting with the nodes with the lowest BC values. Although a high BC value may indicate that a node is central within a community, it can also be an indication of a node that is central within the graph, connecting different communities. Moreover, if we start with a node characterized by a high BC value, it is highly possible to end up with a single community that includes all nodes of the graph. In each iteration, if the currently-selected node υ is not assigned to any group yet, a new subset of the graph called clique is created. In this clique, we include all nodes that belong to the neighborhood of υ . Moreover, we further expand this clique using Bounded-BFS with typical depth value $\sqrt{N}$ (where N is the number of nodes). By applying this procedure, after the completion of all iterations, a set of groups (cliques) will be created. This fatefully leads to phase three of the algorithm.

The large number of the created groups raises the need for some sort of minimization, in order to get the desired generalized community structure. This task is carried out by merging these groups through an iterative process. We define an $l \times l$ matrix B, where l refers to the number of the groups produced at phase two. Each element B [i, j], with $I \neq j$ of the matrix corresponds to the number of edges that connect directly nodes assigned in group i to nodes assigned in group j. On the other hand, each element B [i, j] with i = j corresponds to the number of edges internal in group i. In each iteration, the pair of groups with maximum $\frac{B[i,j]}{B[i,i]}$ value is selected for merging and then the recalculation and repopulation of matrix B is required. The process terminates when there is no pair of groups with $\frac{B[i,j]}{B[i,i]} \geq 1$

Bridge Bounding

Bridge Bounding is a local methodology for community detection [53]. The algorithm initiates the community detection from a certain seed node and progressively expands the community trying to identify bridges, i.e. edges that act as community boundaries. The edge clustering coefficient is calculated for each edge, looking at the edge's neighborhood, and edges are

characterized as bridges depending on whether their clustering coefficient exceeds a threshold. The method is local, has low complexity and allows the flexibility to detect individual communities. Additionally, the entire community structure of a network can be uncovered starting the algorithm at various unassigned seed nodes, till all nodes have been assigned to a community. [54]

In order to identify a community around a seed node *s* the algorithm uses a flooding technique: Starting at node *s*, nodes in the neighborhood of *s* are gradually attached to the community if the following two conditions are satisfied: neighbor *v* does not belong to any other community and the edge connecting *s* to *v* is not a bridge (community boundary).

The term bridge defines an edge connecting two nodes that are members of different communities. The steps described above are repeated for every node until no other node can be attached to the community. Repeating the same procedure for different nodes, inevitably leads to the discovery of the overall community structure of the graph.

Label RankT

LabelRankT is an on-line distributed algorithm for detection of communities in large-scale dynamic networks through stabilized label propagation.

It is based on Label propagation, where a set of random nodes is initialized with different labels. Iteratively, each node adopts the label of the majority of its neighbors. By allowing for more than one label per node, the algorithm is able to detect overlapping communities. During subsequent snapshots, this algorithm only needs to update nodes which have been modified compared to the previous snapshot, or nodes whose neighbors' label has changed during this procedure, making its execution faster.

LabelRankT [55] can be viewed as a LabelRank [56] with one extra conditional update rule by which only nodes involved any change accept the new distribution. LabelRank relies on four operators applied to the labels: (i) propagation, (ii) inflation, (iii) cutoff, and (iv) conditional update. The data structure that lies in the core of this algorithm is the sparse matrix of label distribution. Each node maintains a label distribution locally during the propagation. At the end of the algorithm, LabelRank ranks labels in each node. Nodes with the same highest probability label form a community.

LabelRankT only needs to update nodes that are changed between two consecutive snapshots, including cases where an existing node adds or deletes links, or a node is removed from or newly joins the network.

4.2 Detecting Overlapping Communities

In this Section we shall describe one of the main contributions of this thesis, which is a new approach in detecting overlapping communities. The approach is based on SCCD, a recent, state-of-the-art algorithm designed and developed by the Technological Educational Institute of Crete. In turn, the SCCD [57] algorithm is inspired by Vivaldi.

Vivaldi

Vivaldi [58] is a fully decentralized, light-weight, adaptive network coordinate algorithm that was initially developed to predict Internet latencies with low error. Vivaldi uses the Euclidian coordinate system (in *n*-dimensional space, where n is a parameter) and the associated distance function. Conceptually, Vivaldi simulates a network of physical springs, placing imaginary springs between pairs of network nodes.

Let $G = (V, E)$ denote the given graph comprising a set V of nodes together with a set E of edges. Each node $x \in V$ participating in Vivaldi maintains its own coordinates $p(x) \in W_n$ (the position of node x that is a point in the n-dimensional Euclidian space). The Vivaldi method consists of the following steps:

- Initially, all node coordinates are set at the origin.
- Periodically, each node communicates with another node (randomly selected among a small set node of nodes known to it). Each time a node communicates with another node, it measures its latency and learns that node's coordinates. Subsequently, the node allows itself to be moved a little by the corresponding imaginary spring connecting them (i.e. the positions change a little so as the Euclidian distance of the nodes to better match the latency distance).
- When Vivaldi converges, any two nodes' Euclidian distance will match their latency distance, even though those nodes may never have any communication.

Unlike other centralized network coordinate approaches, in Vivaldi each node only maintains knowledge for a handful of other nodes, making it completely distributed. Each node computes and continuously adjusts its coordinates based on measured latencies to a handful of other nodes. Finally, Vivaldi does not require any fixed infrastructure as for example landmark nodes.

SCCD

As we mentioned, in the core of SCCD lies the spring metaphor which inspired the Vivaldi algorithm. Vivaldi uses the spring relaxation metaphor to position the nodes in a virtual space (the *n*-dimensional Euclidean space), so as the Euclidean distance of any two node positions approximates the actual distance between those nodes. In the original application of Vivaldi, the actual distances were the latencies between Internet hosts. Our algorithm is based on the idea that by providing our own, appropriate, definition of distance between nodes, we can use Vivaldi to position the nodes in a way as to reflect community membership, i.e. nodes in the same community will be placed closer in space than nodes of different communities. In other words, nodes belonging to the same community will form natural clusters in space.

Let *C(x), C(y)* denote the communities' sets of two nodes $x, y \in V$, respectively, of a given graph. Since two nodes either belong to the same community ($C(x) = C(y)$) or not, we define the initial node distance between two nodes *x* and *y* as *d(x, y)*:

$$d(x,y) = \begin{cases} 0, & C(x) = C(y) \\ 1, & C(x) \neq C(y) \end{cases}$$

Equation 11 - Node distance between two nodes x and y

When $C(x) = C(y)$, we have set $d(x, y) = 1$ in order to normalize the distances in range between 0 and 1. Given this definition of distance, we can employ the core part of the Vivaldi algorithm to position the nodes appropriately in the *n*-dimensional Euclidian space ($\mathcal{R}_n$). As one can expect from those dual distances, Vivaldi will position nodes in the same community close-by in space, while place nodes of different communities away from each other. This is the reason for the dual nature of the distance function, otherwise all nodes, regardless of community membership, would gravitate to the same point in space.

In addition, Vivaldi requires a selection of nodes to probe. Each node calculates a "local" set containing nodes of the same community, and a "foreign" set containing nodes of different communities. The size of the local set as well as the size of the foreign set of a node equals the degree of the node. The perfect construction of these sets depends on the a priori knowledge of node community membership, which is the actual problem we are trying to solve. However, even though we do not know the community each node belongs to, there are two facts we can exploit to make Vivaldi work without this knowledge:

- The first is the fact that, by definition, the number of intra-community links of a node exceeds the number of its inter-community links. This means that, if we assume that all of a node's neighbors belong to the same community, this assumption will be, mostly, correct, which in turn means that even though sometimes the node may move to the wrong direction, most of the time it will move to the right direction and thus, will eventually acquire an appropriate position in space. Thus, we let the local set $L(x)$ of a node $x \in V$, be its "neighbor set". $L(x) = \{y \in V: x \sim y\}$ The distance from node x to nodes in $L(x)$ is set to 1 according to Equation (11).
- The second fact we exploit concerns the foreign links. Since we consider all a node's links as local links, we need to find some nodes which most likely do not belong to the same community as that node, and therefor will be considered as foreign nodes. This can simply be done by randomly selecting a small number of nodes from the entire graph. Assuming that the number of communities in the graph is at least three, the majority of the nodes in this set will belong to a different community than the node itself. These nodes will comprise the "foreign set" $F(x)$ of node $x \in V$: $F(x) \subset \{y \in V : x \sim y\}$ (3). The distance from node x to the nodes in $F(x)$ is set to 1 according to Equation (11).

Algorithm 1: The position estimation algorithm.

input : $L(x), F(x),\ \forall x \in V$.
output: $p(x),\ \forall x \in V$.

foreach $x \in V$ **do**
  $p(x) =$ random position in $\Re^n$
  $ite(x) = 0$
end
repeat
  foreach $x \in V$ **do**
    if $getRandomNumber(0, 1) < 0.5$ **then**
      Let v be a random vertex from $L(x)$
      $p(x) = Vivaldi(p(x), p(v), 0)$
    else
      Let v be a random vertex from $F(x)$
      $p(x) = Vivaldi(p(x), p(v), 1)$
    end
    $ite(x) = ite(x) + 1$
    if $ite(x) > 5 \cdot (|L(x)| + |F(x)|)$ **then**
      $ite(x) = 0$
      $maxD = \max_{y \in L(x)}(||p(x) - p(y)||)$
      $minD = \min_{y \in L(x)}(||p(x) - p(y)||)$
      $T_2 = minD + max(\frac{maxD - minD}{3}, \frac{1}{3})$
      $\mu = \frac{minD + maxD}{2}$
      $\sigma_n = \frac{\sqrt{E_{y \in L(x)}[(||p(x) - p(y)|| - \mu)^2]}}{T_2}$
      if $\sigma_n > 0.6$ **then**
        foreach $y \in L(x)$ **do**
          if $||p(x) - p(y)|| > T_2$ **then**
            $L(x) = L(x) - \{y\}$
          end
        end
        foreach $y \in F(x)$ **do**
          if $||p(x) - p(y)|| \leq T_2$ **then**
            $F(x) = F(x) - \{y\}$
          end
        end
      end
    end
  end
until $\forall x \in V\ p(x)$ *is stable*

Figure 17 Algorithm 1 The position estimation algorithm.

The pseudo-code of the position estimation algorithm is given in Algorithm 1 and it is described hereafter. The function *getRandomNumber(0, 1)* returns a random number in *[0, 1]*. Initially, each node is placed at a random position in $\Re_n$. Iteratively, each node $x \in V$ randomly selects

a node from either its *L(x)* or its *F(x)* set (see line 8,11 of Algorithm 1). It then uses Vivaldi to update its current position using the appropriate distance (i.e. 0 or 1) to the selected node (see lines 9, 12 of Algorithm 1). Each node continues this process until it deems its position to have stabilized as much as possible (see line 36 of Algorithm 1). This is done by calculating the sum of the distances between each two consecutive positions of the node between 40 iterations (corresponding to 40 position updates, experiments showed a larger number only slows down the algorithm without adding to efficiency). Each node also calculates the distance, in a straight line, between the two positions before and after the 40 updates. Should this value be less than half the actual traveled distance (the aforementioned sum) for 20 consecutive times, the node declares itself to have stabilized. Each node continues, however, to execute the algorithm until at least 90% of its "foreign" and "local" sets have also stabilized. If the algorithm has not stabilized yet, the 20 oldest distances are removed and the stabilization check will be performed again after another 20 position updates.

Modifying SCCD to detect overlapping communities

SCCD is a highly accurate community detection algorithm which however only detects distinct communities. This means that each node in the resulting partition can be a member of a single community. The aim of the approach described here is the proper use of the algorithm in order to detect overlapping communities, i.e. communities which "share" members.
The proposed process can be summarized in the following steps:

1. Execute SCCD algorithm on the initial graph to identify distinct communities.
2. Create a new version of the graph where any intra-community (as identified in step 1) edges have been removed.
3. Go to step 1, until no edges are left in the graph, or all detected communities are singletons.

A simple scenario, in order to showcase the reasoning of the above process is shown on the following figure:

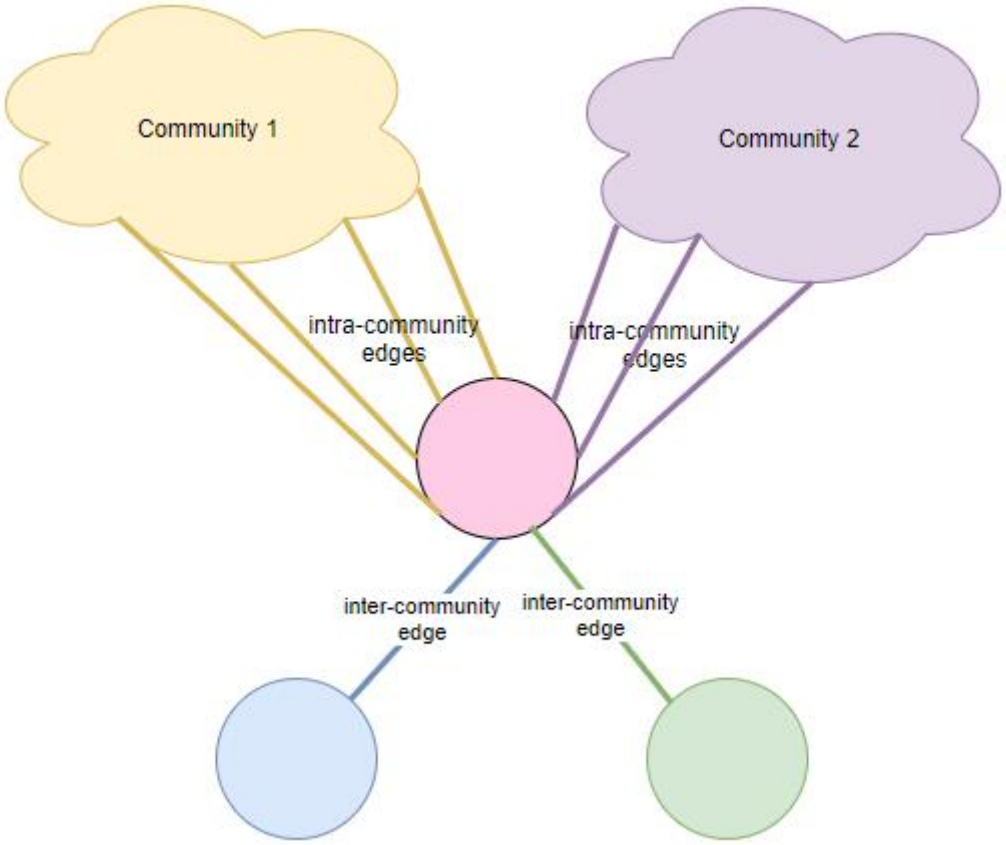

Figure 18 - Community scenario

In this scenario, the central node belongs both to community 1 and community 2. In addition, the node has 2 inter-community edges to nodes which belong to communities in which the central node is not a member of. The first pass of the process will identify the central node as member of community 1. As such, step 2 of the procedure will prune the graph of all yellow colored edges. As a result, in the second pass, the same node will be identified as a member of community 2, since in the new graph, the purple edges comprise the majority of that node's neighbor set.

Appropriate Java code was created to implement step 2. Algorithm 2 shows this source code.

Algorithm 2: Pruning intra-community edges
import java.io.BufferedReader; import java.io.BufferedWriter; import java.io.FileNotFoundException; import java.io.FileReader; import java.io.FileWriter; import java.io.IOException; import java.util.ArrayList; import java.util.Arrays; import java.util.HashSet;

```
import java.util.List;

public class PruneIntraEdges {
   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) throws FileNotFoundException, IOException {
      String edgesInputFile = args[0]+".txt";     //to arxeio me tis akmes
      String communitiesInputFile = args[0]+".coms";          //to arxeio me ta communities
      String []temp = args[0].split("\\.");
      int step = Integer.parseInt(temp[1])+1;
      String edgesOutputFile = temp[0]+"."+step+".txt";    //to apotelesma

List<HashSet<String>>communities= initCommunitiesList(communitiesInputFile);

      BufferedReader br = new BufferedReader(new FileReader(edgesInputFile));
      BufferedWriter bw = new BufferedWriter(new FileWriter(edgesOutputFile));
      String line;
      while ((line = br.readLine()) != null)
      {
         String[] nodes = null;
         if (line.contains("\t")) nodes = line.split("\t");
         else nodes = line.split(" ");
         boolean isForeign = true;
         for (HashSet<String> community : communities)
         {
            if (community.contains(nodes[0]) && community.contains(nodes[1]))
            {
               isForeign = false;
               break;
            }
         }
         if (isForeign) bw.append(line + "\n");
      }
      bw.flush();
      br.close();
      bw.close();
   }

   public static List<HashSet<String>> initCommunitiesList(String communitiesFile) throws FileNotFoundException, IOException {
      List<HashSet<String>> communities = new ArrayList<>();
      BufferedReader br = new BufferedReader(new FileReader(communitiesFile));
      String line;
      while ((line = br.readLine()) != null) {
         line = line.trim();
         HashSet<String> community = new HashSet<String>();
         for (String s : line.split(" "))
```

```
                community.add(s);
            //community.addAll(Arrays.asList(line.split(" ")));
            communities.add(community);
        }
        br.close();
        return communities;
    }
}
```

Figure 19 Algorithm 2: Pruning intra-community edges

4.3 Experimental Results

Two datasets were used for the evaluation experiments. Both datasets contain overlapping communities. They are both part of the SNAP dataset library. The next table contains statistical data for both datasets.

Name	Type	Nodes	Edges	Communities	Description
DBLP	Undirected, Communities	317,080	1,049,866	13,477	DBLP collaboration network
Amazon	Undirected, Communities	334,863	925,872	75,149	Amazon product network

Table 4 - Testing Datasets

In order to evaluate the proposed method, the Modularity metric was used, in order to identify the quality of the detected communities on each algorithmic step. The following table presents the obtained results on both testing datasets:

Iteration Nr:	Amazon		DBLP	
	Communities	Modularity	Communities	Modularity
1	2538	0.91791	4005	0.64569
2	13516	0.98562	3736	0.78212
3	546	0.99951	1873	0.95238
4	-	-	212	0.99741
Total:	16600		9826	

Table 5 - Obtained Results of testing datasets

Chapter 5: Conclusions and Future Work

Contents

5.1 Conclusions

The purpose of this thesis was two-fold. The first goal was to present a survey of important data mining techniques used in social networks, namely community detection and item recommendation. A thorough survey of various techniques was presented along with their advantages and disadvantages.

In the context of Recommender Systems, a taxonomy was presented of the categories of the various approaches in tackling the problem. The strengths and weaknesses of each approach was presented along with the open challenges still present in the field. Finally, a description of widely used recommender systems was presented in real-world cases.

Regarding the field of Community Detection, the thesis presented various structures, characteristics and properties of networks where community detection techniques are usually applied to. The characteristics and properties are usually used in the various techniques to detect the underlying communities. The most popular techniques for detecting communities were then presented, covering various approaches in tackling the issue as well as detecting communities with different characteristics, such as strong or weak, distinct or overlapping.

The second goal of the current thesis is to present two recent contributions, one in each field. In the field of Recommender Systems, SCoR, a newly proposed algorithm designed in our institution was presented. SCoR has been tested on several real datasets with high variability in density and size. A comparison was presented against seven state-of-the-art recommender systems, proving its effectiveness, stability and higher performance. Under any dataset, SCoR is the first in performance. Apart from the high performance and stability of SCoR, other advantages of the proposed system compared to the other state-of-the-art algorithms are the fact that it does not require any parameter for execution.

In the field of Community Detection, a community finding algorithm was thoroughly presented, which is also based on a custom-tailored version of the Vivaldi network coordinate system. The proposed algorithm has been tested on a large number of benchmark graphs with known community structure comparing it with several state-of-the-art algorithms, proving its effectiveness against all other algorithms. SCCD can accurately detect strong, distinct communities in graphs. In addition, the thesis presented a modification to the proposed algorithm to enable the detection of overlapping communities.

.

5.2 Future Work

Regarding SCoR, several directions can be explored regarding various aspects of the recommender systems. The system can be improved in order to better handle sparse datasets as well as cold-start users. An important axis for future work includes exploring and demonstrating the performance of SCoR under dynamic changes in the dataset. The convergence ability of the algorithm as new users and/or items arrive into the system should be tested under different scenarios. The fact that items tent to arrive in a much faster pace compared to users will be evaluated. Another important research direction would be to explore the behavior of the system under temporal patterns, as items tend to become very popular for a period of time and to fade away subsequently, and users tend to re-rate the same items differently. To study these phenomena the temporal characteristics of the datasets should be taken into consideration in the performed experiments. Finally, incorporating metadata and semantic information in the algorithm could enhance significantly its performance.

Regarding SCCD, one possible improvement would be the modification of the algorithm in order to locate only a single community. It is important to provide the single community detection (per node) instead of entire community detection. Another possible extension of the proposed scheme is the application in weighted networks that can measure the strength of social relationships in social networks. In iterative process of position estimation algorithm, this extension can be done by setting the probability of edge selection according to the edge weight, so that the strong edges would have higher selection probability corresponding to high-tension springs.

References

[1] G. Adomavicius and Y. Kwon, "Improving aggregate recommendation diversity using ranking-based techniques," *IEEE Transactions on Knowledge and Data Engineering*, pp. 896-911, 2012.

[2] P. Melville and Vikas Sindhwani, "Recommender Systems," in *Encyclopedia of Machine Learning*, 2010, pp. 1-9.

[3] Gorrell, G., "Generalized hebbian algorithm for incremental singular value decomposition in natural language processing," in *EACL 2006, 11st conference of the european chapter of the association for computational linguistics, proceedings of the conference, april 3–7, 2006*, trento, italy, 2006.

[4] Park, D. H. , Kim, H. K. , Choi, I. Y. and Kim, J., "A literature review and classification of recommender systems research.," *Expert Systems with Applications*, pp. 10059-10072, 2012.

[5] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, pp. 734-749, Ιούνιος 2005.

[6] N. Friedman, Geiger, D. and Goldszmidt, M., "Bayesian Network Classifiers," *Machine Learning*, 01 November 1997.

[7] C. Bishop, "Pattern Recognition And Machine Learning," pp. ISBN 0-387-31073-8, 2006.

[8] Y. Koren, Robert Bell and Chris Volinsky, "MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS," *Computer*, vol. Volume 42 , no. Issue 8, pp. 30-37, August 2009.

[9] Gunawardana, Guy Shani and Asela, "Evaluating Recommendation Systems," in *Recommender Systems Handbook*, pp. 257-297.

[10] L. Lü, Matúš Medob, Chi Ho Yeungb, Yi-Cheng Zhangb and Ke Zh, "Recommender systems," *Physics Reports*, pp. 1-49, Οκτώβριος 2012.

[11] "Wikipedia, tf–idf," [Online]. Available: https://en.wikipedia.org/wiki/Tf%E2%80%93idf. [Accessed 7 8 2018].

[12] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Recommender_system.

[13] M. Billsus and J. Pazzani, "Content-Based Recommendation Systems," *The Adaptive Web*, pp. 325-341, 2007.

[14] Khoshgoftaar, Xiaoyuan Su and Taghi M., "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, August 2009.

[15] Linden, G., Smith, B. and York, J., "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, pp. 76-80, 2003.

[16] Carrer-Neto, W., Hernández-Alcaraz, M. L., Valencia-García, R. and García-Sánchez, F., "Social knowledge-based recommender system. ap- plication to the movies domain.," *Expert Systems with applications*, vol. 39 (12), p. 10990–11000, 2012.

[17] D. Goldberg, Oki, B. M. and Terry, D., "Collaborative filtering to weave and information tapestry," *Communications of the ACM, 35(12)*, p. 61–70, 1992.

[18] Goldberg, K., Roeder, T., Gupta, D. and Perkins, C., "A constant time collaborative filtering algorithm," *Information Retrieval*, pp. 133-151, 2001.

[19] P. Resnick, Iacovou, N., Suchak, M., Bergstorm, P. and Ried, "Grouplens: An open architecture for collaborative filtering of netnews," *In Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, p. 175–186, 1994.

[20] K. J. A. Herlocker, J. L. and Terveen, L. G., "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, pp. 5-53, 2004.

[21] Zhang, Y. and Zhang, H., "Triangulation inequality violation in internet delay space.," *In Advances in computer science and information engineering* , p. (pp. 331–337). Springer, 2012.

[22] P. Caravelas and G. Lekakos, "MoRe: a Recommendation Systems combining Content-based and Collaborative filtering," in *In Proceedings of the 4th EuroiTV conference*, Athens, Greece, 2006.

[23] J. Davidson, Liebald, B. and Liu, J., "The YouTube Video Recommendation System," *RecSys2010*, pp. 26-30, September 2010.

[24] J. Hannon, Mike Bennett, M. and Smyth, B., "Recommending twitter users to follow using content and collaborative filtering approaches," *RecSys*, pp. 199-206, 2010.

[25] F. Dabek, R Cox, F Kaashoek and R Morris, "Vivaldi: A decentralized network coordinate system. In Proceedings of the 2004 conference on applications, tech- nologies, architectures, and protocols for computer communications . In SIGCOMM ’04," 2004.

[26] H. Papadakis, C. Panagiotakis and P. Fragopoulou, "SCoR: A Synthetic Coordinate based Recommender system," *Expert Systems With Applications*, pp. 8-19, 17 February 2017.

[27] "The smallnetflix recommender systems dataset.," 2012. [Online]. Available: http://www.select. cs.cmu.edu/code/graphlab/datasets/.

[28] K. Goldberg, "The jester recommender systems dataset. http://www.ieor. berkeley.edu/ ∼goldberg/jester-data/," 2003.

[29] Y. Zhou, Wilkinson, D., Schreiber, R. and Pan, R., "Large-scaleparallel collabo- rative filtering for the netflix prize. In Proc. 4th intl conf. algorithmic aspects in information and management, lncs 5034," *Springer*, p. 337–348, 2008.

[30] H.-F. Yu, Hsieh, C.-J., Si, S. and Dhillon, I, "Scalable coordinate descent ap- proaches to parallel matrix factorization for recommender systems. In Proceed- ings of the 2012 ieee 12th international conference on data mining . In ICDM ’12," 2012.

[31] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model. In Proceedings of the 14th acm sigkdd international conference on knowledge discovery and data mining . In KDD ’08 (pp. 426–434) .," 2008.

[32] G. Hinton, *A practical guide to training restricted Boltzmann machines. Technical Report . University of Toronto,* 2010.

[33] Y. Koren, Bell, R. and Volinsky, C., *Matrix factorization techniques for recommender systems. Computer, 42 (8), 30–37,* 2009.

[34] M. D. Ekstrand, Riedl, J. T. and Konstan, J. A., Collaborative filtering recommender systems. Foundations and Trends in Human-Computer Interaction, 2011, p. 81–173.

[35] B. Sarwar, Karypis, G., Konstan, J. and Riedl, J., *Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international confer- ence on world wide web . In WWW ’01 (pp. 285–295). New York, NY, USA: ACM .*, 2001.

[36] M. Haindl and Mikeš, S., *A competition in unsupervised color image segmentation. Pattern Recognition, 57 , 136–151,* 2016.

[37] Derek Greene, Donal Doyle and Padraig Cunningham, "Tracking the evolution of communities in dynamic social networks.," *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining,*, pp. 176-183, 2010.

[38] Nguyen NP , Dinh TN, Shen Y and Thai MT., "Dynamic social community detection and its applications.," *Dynamic Social Community Detection and Its Applications.*, p. PLoS ONE 9(4): e91431, 2014.

[39] C. Panagiotakis, H. Papadakis and P. Fragopoulou, "FlowPro: A Flow Propagation Method for Single Community Detection," in *IEEE Consumer Communications and Networking Conference*, 2014.

[40] Jie Chen and Yousef Saad, "Dense subgraph extraction with application to community detection," *IEEE Transactions on Knowledge and Data Engineering,*, pp. 24:1216-1230, 2012.

[41] Gergely Palla, Imre Derenyi, Illes Farkas and Tam Vicsek, *Uncovering the overlapping community structure of complex networks in nature and society,* 2015.

[42] M. Rosvall and C.T. Bergstrom., "An information-theoretic framework for resolving community structure in complex networks.," *Proceedings of the National Academy of Sciences,* p. 104(18):7327, 2007.

[43] Satu Elisa Schaeffer, "Graph clustering.," *Computer Science Review,* pp. 27 - 64, 2007.

[44] Hamidreza Alvari, Alireza Hajibagheri and Gita Reese Sukthankar, "Community detection in dynamic social networks: A game-theoretic approach.," in *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM 2014, Beijing, China, August 17-20, 2014.

[45] Reid Andersen, Fan Chung and Kevin Lang, "Local graph partitioning using pagerank vectors.," in *FOCS'06. 47th Annual IEEE Symposium on, pages 475-486.,* 2006.

[46] James P. Bagrow and Erik M. Bollt., "Local method for detecting communities.," *Physical Review E,* p. 72(4):46{108, 2005.

[47] Mo-Han Hsieh and Christopher L Magee, "A new method for finding subgroups from networks," *Social Networks,* pp. 32(3):234{244, 2010., 2010.

[48] C. Panagiotakis, H. Papadakis and P. Fragopoulou, "CoViFlowPro: A Community Visualization method based on a Flow Propagation Algorithm," in *International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS) - BICT 2014*, 2014.

[49] Andrea Lancichinetti and Santo Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities.," in *Phys. Rev. E,*, 2009, p. 80(1):016118.

[50] M. Girvan and E. J. Newman, "Finding and evaluating community structure in networks," *Physical Review E,* p. 69(2):026113, 26 02 2004.

[51] D. Katsaros, G. Pallis, K. Stamos and A. Vakali, "Cdns content outsourcing via generalized communities," *IEEE Transactions on Knowledge and Data Engineering,* p. 21:137–151, 2009.

[52] Andrea Lancichinetti and Santo Fortunato, *Community detection algorithms: a comparative analysis.,* Physical Review E, 80(5 Pt 2):056117, 2009.

[53] S. Papadopoulos, A. Skusa, A. Vakali, Y. Kompatsiaris and N. Wagne, "Bridge bounding: A local approach for efficient community discovery in complex networks," Technical Report arXiv:0902.0871, Feb 2009.

[54] Pan Hui, Eiko Yoneki, Shu Yan Chan and Crowcroft Jon, "Distributed community detection in delay tolerant networks.," in *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture,*, 2007.

[55] J. Xie, Mingming Chen and Boleslaw K. Szymanski, "LabelRankT: Incremental Community Detection in Dynamic Networks via Label Propagation," in *DyNetMM Workshop at the SIGMOD/PODS*, New York City, June 22-27, 2013.

[56] J. X. Szymanski and B. K., "Labelrank: A stabilized label propagation algorithm for community detection in networks," in *IEEE Network Science Workshop*, West Point, NY, 2013.

[57] H. Papadakis, C. Panagiotakis and P. Fragopoulou, *SCCD: Distributed detection of communities based on synthetic coordinates, INTERNATIONAL CONFERENCE ON COMPUTATIONAL SOCIAL SCIENCE (ICCSS), 2015.*

[58] F. Dabek, Russ Cox, Frans Kaashoek and Robert Morris, "Vivaldi: A Decentralized Network Coordinate System," in *SIGCOMM'04, Aug. 30–Sept. 3*, Portland, Oregon, USA, 2004.