# The effects of orientation in a decentralized recommender

Mathieu Pasquet


Research Master Thesis


Research Master Thesis

# The effects of orientation in a decentralized recommender

*Author:*
Mathieu Pasquet

*Supervisor:*
Davide Frey
Team: ASAP


**Abstract**

Recommender systems are in use in every shopping system, suggesting new items to buy, new songs to listen to, new movies to watch... However, news recommendation is often still done on an explicit subscription basis, which does not yield sufficient results, being either too broad or too narrow in its scope. WhatsUp is a new distributed news recommender using implicit subscriptions to suggest links a user may want to read.

Our goal was to study the effect orientation in a decentralized push-based recommender (WhatsUp), with a particular focus on content-based recommendation. We have applied several techniques to try to improve upon the current State of the Art. One of these methods was content-based, which yielded promising results for the most complete data set. This work lays the foundations for new experiments with data more closely correlated to the situation we wanted.

# Contents

# Introduction

Recommender systems play an important role in our everyday online interactions, with websites like IMDB, Amazon, Ebay, Facebook that keep suggesting respectively movies, items or new friends based on what they know about us.

More specifically, news items are hard to suggest because they can span over a multitude of topics while being highly dependant on the speed of propagation. Traditional recommender systems like the websites referenced above are not designed to work on such a continuous stream of information. Classical means of receiving news like RSS feeds, or subscription-based websites are either too specific (no serendipity, as in RSS feeds), or too broad (getting every item in the news feed of users you are subscribed to, as in subscription-based systems).

Therefore, a distributed recommender using epidemic protocols, WhatsUp [BFG+13], has been designed in order to avoid many of the issues find in classical recommender systems. However, WhatsUp relies on collaborative filtering (CF) [SK09] that does not take into account the content of the news, but only the similarity between user ratings.

The first part of this report will state the context of this internship regarding the state of the art in recommendation, and describes the foundations of the WhatsUp recommender system.

The second part of this report will describe the goal of this internship, how to investigate the effect of orientation, and more precisely content-based orientation in a distributed recommendation system, in more details. Two ideas to experiment with orientations are detailed.

Those ideas are evaluated experimentally in the next section, where we will study the results obtained this far on both techniques, and also give some insight on the experimental protocol and evaluation metrics.

# Chapter 1

# State of the art

In computer science, a *recommendation system* is a service that aims at providing a suggestion about one or several *items* to a *user* [ZFF11]. Recommending movies, food, or other items between a group of friends by establishing who has the same tastes in order to provide relevant advice has been around for a very long time. What is new, however, is that the sheer computing power available and the growth of online communities allow computer systems to provide automated recommendations based on the opinions of thousands to millions of users instead of a small sample of acquaintances.

In a recommendation system, a user has a *profile*, that is a set of his interactions with the system, being either item views, song plays, purchases, ratings... A recommendation system will leverage this information in order to present him the most pertinent next item that he may want to select.

Recommendation systems attract millions of users every day, on websites like the online shop Amazon. Indeed, their recommendation system presents users with items that they might want to buy, or items that other people bough at the same time. Netflix recommends movies or TV shows by processing its rating database. Twitter suggests to a user a number of people he could follow based on his history. Facebook and Google try to find someone you could know.

## 1.1 Collaborative filtering recommendation

Collaborative Filtering [HKBR99] is used in many real-world applications that contain meaningful user data, such as Amazon, Last.fm, Ebay and others. One of the main reasons for its uses is that collaborative filtering algorithms do not require any knowledge about the items they are operated on, as the memory will be built by the users, and not beforehand by the website.

Breese *et al* [BHK98] distinguish several collaborative filtering techniques:

- Memory-based
- Model-based
- Hybrid

An alternate classification was suggested in [SFHS07] with *probabilistic* and *non-probabilistic* classes, but the techniques are the same.

### 1.1.1 Memory-based

Memory-based algorithms require all the ratings, items and users to be in memory when performing a recommendation.

One of the first memory-based systems GroupLens [RIS+94], also being one of the first collaborative filtering systems, was aimed at predicting the score a user would assign to an Usenet [Use] news. Grouplens used the Pearson correlation formula [Pea01] to assign weights on a continuous scale from 1 (perfect correlation) to -1 (perfect opposition) to the ratings of different users.

$$corr(\alpha, \beta) = \frac{\sum_i (v_{\alpha,i} - \bar{v}_\alpha)(v_{\beta,i} - \bar{v}_\beta)}{\sqrt{\sum_i (v_{\alpha,i} - \bar{v}_\alpha)^2 \sum_i (v_{\beta,i} - \bar{v}_\beta)^2}}$$

Here, $\alpha$ and $\beta$ are the users, $i$ is the set of items rated by both $\alpha$ and $\beta$, and the summations are made over the ratings of the items, and $\bar{v}$ is the average of the ratings of this user, in order to normalize the score of positive/negative users.

### 1.1.2 Model-based

Model-based algorithms have the advantage of being often lighter than their memory-based counterparts [SFHS07], as they use offline periodic computations to update their model. Moreover, they also often allow explanation of the recommendation results from the model, something that is not possible with memory-based algorithms. They are based on the assumption that users can be located in clusters of users with a similar behavior [KSJ09].

The Bayesian Network is a type of statistical model to represent a set of conditional dependencies between random variables. In the context of collaborative filtering, it means using a learning algorithm [CH97] to build this Bayesian network, as done in [BHK98], and then use it to create decision trees that will model the probable behavior of the user. Figureé1.1 gives an example of it in practice.

Ungar *et al* [UF98] suggest that successive use of a k-means clustering algorithm [HW79] to regroup the users and items into appropriate matching categories may help making recommendations. Chee *et al* [CHW01] later created RecTree, a model-based algorithm that creates a binary tree made of cliques of users who have similar interests. To achieve this, they use k-means clustering as a hierarchical procedure and split the dataset in two when constructing the tree from the root to the leaves.

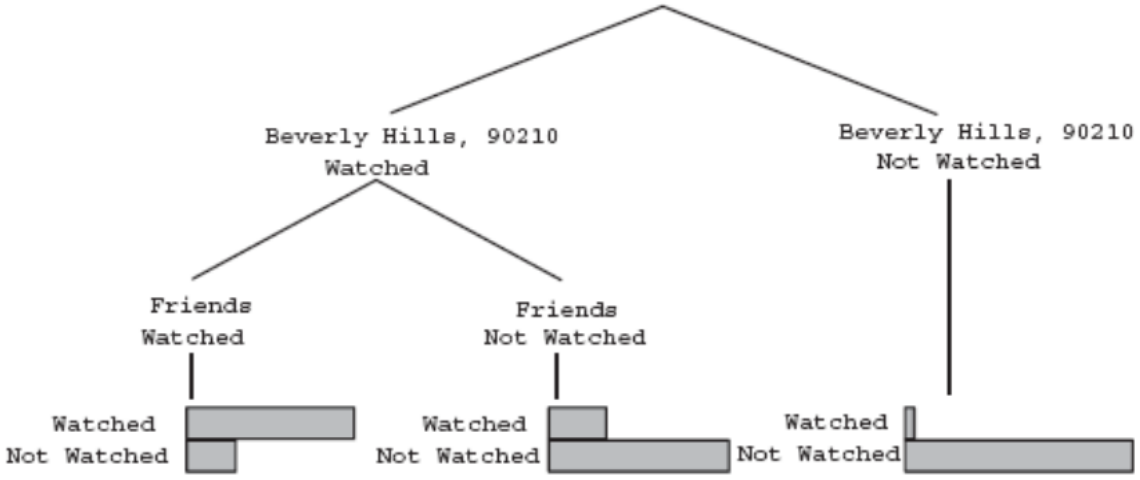



Figure 1.1: An example of decision tree from a Bayesian Network (from [BHK98]), here The system determines whether a user has watched "Melrose Place" based on whether he watched "Beverly Hills" and "Friends".

### 1.1.3 Hybrid

Hybrid recommender systems use both memory-based and model-based algorithms. Probabilistic models are often used in order first limit the number of users before running a memory-based collaborative filtering algorithm.

With Personality Diagnosis [PHLG00], Pennock *et al* propose a model of personality detection using aspects of the memory-based CF algorithms and aspects of the model-based algorithms. They use a memory-based technique to observe the ratings, and then use a Bayesian network to provide explanations about the user's ratings by modelling "personality types" among users.

Xue *et al* [XLY+05] use the k-means clustering algorithm with several passes to identify clusters within the total set of users, using the Pearson coefficient defined in section 2.1.1.

### 1.1.4 Issues of collaborative filtering

One issue with collaborative filtering is that it does not take into account the content itself. Indeed, if a user A has mostly the same set of preferences in books as a user B, but user A can read both english and french while user B can only read french, A will like english books, while B can like the same books in their translated version, but not the original. In a standard collaborative filtering recommendation system, this will introduce a bias that may lead to the system recommending english books to B and decrease the quality of the results, because the collaborative filtering approach is not inherently *content-aware*. This is an advantage because it lowers the cost of adding an item to the system, but also an issue because the recommendation results may be less relevant than what is wanted.

Another issue with pure collaborative filtering systems is *cold-start* [SPUP02a], which is when a user or an item does not have enough ratings. In this case, no similarity metric will work as the

collaborative filtering relies on user ratings for the similarity metrics or model-based approaches. Those issues call for *content-based recommendation*.

## 1.2 Content-based recommendation

Content-based recommendation uses models (often from the machine learning domain) to gather and exploit information about the items themselves and the user's preferences in them, rather than concentrating only on user ratings and similarity.

### 1.2.1 Keyword-based representation

In content-based recommendation, the algorithms often ([L+95, MW99, Mou97]) use keywords (also called *tags*) that are either added manually, inferred through the name of the item, or extracted from the document if the item is a textual document (as in the case of news recommendation). The keyword representation of a document is often a simple keyword matching in the content of the document, or a Vector Space Model (VSM) with the Textual Frequency-Inverse Document Frequency (*TF-IDF*) [Sal89] weighting scheme. In the Vector Space Model, each document is represented as a vector of term weights, where each weight represents the degree of association between the document and the term.

The set of words present in the documents is obtained through a set of language processing operations such as tokenization, stop-words removal and stemming. This set is then processed through the TF-IDF weighting scheme, whose core properties are:

- rare terms are not penalized against frequent terms (Inverse Document Frequency assumption)
- several occurences of a word in a document are not less relevant than single occurences (Textual Frequency assumption)
- Long documents are not preferred to short documents (normalization)

$$TF - IDF(t, d, D) = \underbrace{TF(t, d)}_{TF} \cdot \underbrace{log \frac{|D|}{|\{d' \in D : t \in d'\}|}}_{IDF}.$$

Here, $t$ is the term to weigh inside the document $d$, and $D$ is the set of documents present inside the recommendation system. TF is a simple frequency-counting function for $t$ in $d$, and IDF measures whether the term is common across the documents, qualifying its rarity.

The similarity between two vectors (of frequencies computed through TF-IDF) can then be computed through the well-known cosine similarity metric [PNSK+06]:

$$sim(A, B) = \frac{\sum_i A_i \cdot B_i}{\sqrt{\sum_i A_i^2} \cdot \sqrt{\sum_i B_i^2}},$$

$A$ and $B$ being two vectors, and $A_i$, $B_i$ term frequencies inside the vectors.

### 1.2.2 Linear classifiers

The goal of statistical classification is to leverage an object's characteristics to find the class in which it belongs. A linear classifier does this by classifying through the value of a linear combination of those characteristics. One of the most used linear classifier in recommendation is the Naïve Bayesian Classifier [LGS11]. Schein et al. [SPUP02b] have identified it as a well-performing test classification algorithm, and it has been used in many works. Naïve Bayes can be expressed with two models [FLNP00]: the multinomial model and the multivariate (linear gaussian) Bernouilli model.

### 1.2.3 Rocchio and Relevance Feedback

Rocchio's algorithm [Roc71] is an algorithm working in the Vector Space Model that is designed to incrementally improve the relevance of a query by relying on user feedback on the results. This system was primarily designed to improve the quality of results for a users' query, but a query is similar to a recommendation except that a query is initiated by a user, whereas a recommendation is generally an automated suggestion by the recommendation system. Several content-based recommenders have used this method to learn user profiles, such as YourNews [ABG+07], Syskill & Webert [PB97] or Fab [BS97].

The users rate the results of the recommendation regarding the pertinence of the results compared to what they are interested in.

$$Q_{i+1} = \alpha \cdot Q_i + \beta \cdot \sum_{related} \frac{D_i}{|D_i|} + \gamma \cdot \sum_{non-related} \frac{D_i}{|D_i|}$$

In the above formula, $Q_i$ is the user's recommendation at iteration $i$, $\alpha$, $\beta$, and $\gamma$ are the parameters that control the influence of respectively the original recommendation, the related items, and the unrelated items (both based on user feedback).

## 1.3 Decentralized recommendation

Collaborative Filtering in a decentralized fashion has been studied before; PipeCF [HXYS04] was one of the first attempts to use peer-to-peer networks to try to achieve both low maintenance overhead and scalability, by addressing the profiles in the Chord Distributed Hash Table (DHT) [SMK+01]. However, there are doubts concerning the scalability of this approach [BFG+13], as it is unclear whether the system can cope with the reactivity and the load expected in modern systems.

PocketLens [MKR04] was also one of the first decentralized recommenders, but PocketLens had a much more important focus on privacy and trust, based on the idea that a centralized recommender could violate its privacy policy and sell the user profiles, and bias the recommender behavior in order to suggest items that bring more profit, for example. To achieve this, they establish five possible architectures for PocketLens: one with a central server that stores all the ratings, while the recommendation is made client side; one using the gnutella [RFI02] network to discover random peers; one using again the gnutella network, but this time to forward queries to the relevant nodes; one using the Chord DHT to publish the user ratings; and one re-using results from [CGS97] and [Can02] to propose an item-to-item model that works on encrypted data.

TiVo [AVS04] is a distributed recommender for TV shows. TiVo uses both collaborative filtering and content-based recommendation (to avoid cold-start issues). It uses a central server to push a number of potential recommendations to client-side boxes, that will then perform a collaborative filtering. Its dedicated hardware, centralized server and proprietary undisclosed coefficient that makes an apparently quadratic algorithm give fast predictions make it unsuitable for other uses.

More recently, the Tribler [Tri10] decentralized search engine for the Bittorent protocol was able to make file recommendations to a user after several searches. Tribler uses epidemic protocols to build neighborhoods of correlated users, and the cosine metric we defined earlier to evaluate the similarity between users. However, addressing content in Bittorent is a very stable context, whereas managing items that have a short lifespan and are inherently dynamic such as news items is a different problem.

## 1.4 WhatsUp

WhatsUp [BFG+13] is a recent decentralized news recommender effort using gossip protocols. Its goal is to diverge from classical publish-subscribe approaches for content recommendation where the filtering is either too broad or too restricted. It uses collaborative filtering in a decentralized manner in order to achieve both *scalability* and *accuracy*.

## 1.5 Epidemic protocols

Gossip protocols, also called epidemic protocols, are a class of protocols that are inspired by the autonomous propagation of rumor or infectious diseases in human networks. They are built on the ability to select random peers from the network and then proceed with information exchange. They were first introduced by Demers *et al* in [DGH+87], for database replication. This paper introduced two gossiping principles: anti-entropy and rumor mongering:

**Rumor mongering** is built around the concept of rumor: a new information (*rumor*) received by a node will become a *hot rumor* and this node will seek to propagate this new information to other nodes. The rumor stops being a hot rumor after a few cycles.

**Anti-Entropy** is, on the other hand, designed to have each node contact randomly (and periodically) another node regardless of whether any of the pairs has new information. Nodes synchronize their states upon this exchange.

Anti-entropy has the advantage over rumor mongering that the propagation of the messages will eventually reach all the nodes, while in rumor mongering, all the hot rumors could stop while some nodes still have not received the update. On the other hand, anti-entropy implies a heavier burden on the network, because messages are sent continuously, even if there are no events to spread.

In the original proposal [DGH+87], both approaches relied on the ability to maintain a global knowledge of the system, whether it is through a centralized index of the nodes, which does not scale because of the important query traffic, or through keeping a complete view of the network at each node, which does not scale either because of the update traffic.

While those issues can be ignored on a network containing a few hundred computers, as it was the case with the original proposal in [DGH+87], it does not scale with the high number of computers present in a network of data centers, or in a peer-to-peer network composed of thousands of personal computers.

To solve this problem, a notion of *gossip-based peer sampling* [JVG+07] has been introduced. The principle of Peer Sampling Services/Peer Sampling Algorithms is to use gossip (anti-entropy) to maintain these views of the network (containing random peers) at each node. Nodes then exchange these views periodically with random nodes and shuffle their views, creating a degree-limited, dynamic random graph.

### 1.5.1 WhatsUp specifics: Clustering and dissemination protocols

WhatsUp has two main parts: a clustering protocol, *WUP*, and a dissemination protocol, Biased EpidEmic Protocol (*BEEP*).

**Definition 1.** A **news item** is composed of a title, a short description, and a link pointing to more information. A user publishing an item associates it with a timestamp of its creation time and a dislike counter field acting as a Time To Live (TTL). The title is also associated with a hash that is computed upon reception of the news item.

**Definition 2.** An **user profile** is where information about news items are stored as a set of triplets: identifier, timestamp and score. The identifier and timestamp are defined above, and the score is the level of interest shown by the user about the item, 1 means interesting, and 0 means not interested.

#### WUP: Clustering protocol

WUP is composed of several bricks: first, an underlying Random Peer Sampling (RPS) [JVG+07] protocol that guarantees connectivity between different nodes while keeping a low clustering coefficient and diameter in the graph, then on the upper layer [VVS05] it also has a clustering protocol (which is called WUP) whose goal is to link similar nodes together. Each node has a local *view* for each protocol where it maintains its partial knowledge of the network; it is a set of nodes represented each by a node identifier, an IP address, a profile as well as a timestamp denoting the time at which the information was generated.

Periodically, each node selects the oldest node from each of its views, and sends it a message containing half of its views for the RPS, and its whole view for WUP. The update of the RPS view is then done by keeping a random sample of the union of the received view and the current one. The WUP protocol works differently, keeping the ones which have a profile similar to that of the local node using a cosine metric that maximizes the number of similar items liked in both profiles. This metric also has a mechanism for preventing spam by taking into account the number of disliked items when evaluating a score for a node.

Taking a node $n$ with a profile $P_n$ and a neighbor $c$ with a profile $P_c$, $sub(P_n, P_c)$ is defined as the subset of the scores in $P_n$ corresponding to items in $P_c$. The spam limiter is achieved by dividing this metric with the number liked by $c$ in this subset. The metric is as follows:

$$Similarity(n, c) = \frac{sub(P_n, P_c) \cdot P_c}{\|sub(P_n, P_c)\|\|P_c\|}.$$

Additionally, WUP employs *item profiles*, that are similar to user profiles, but whose goal is to retain the preferences of each user the item passes through. It means that a news item passing through a different set of users will have a different item profile, and more accurately reflect the interest of the network it traversed.

A user profile is updated whenever the user expresses an opinion about a news item it receives by clicking on a *like* or *dislike* button, or when the users creates the news item himself. A new tuple is then inserted in the profile.

An item profile is updated when a user receives a news item for the first time; it will go through all the items in its user profile and call the *addToNewsProfile* function described below on each item.

If the item is not present in the item profile, then the node adds the tuple with the id, timestamp and score to it. If it is present, then the node will update the score of the item in the news profile by averaging its own value and the value present in the item.

**Biased EpidEmic Protocol (BEEP)**

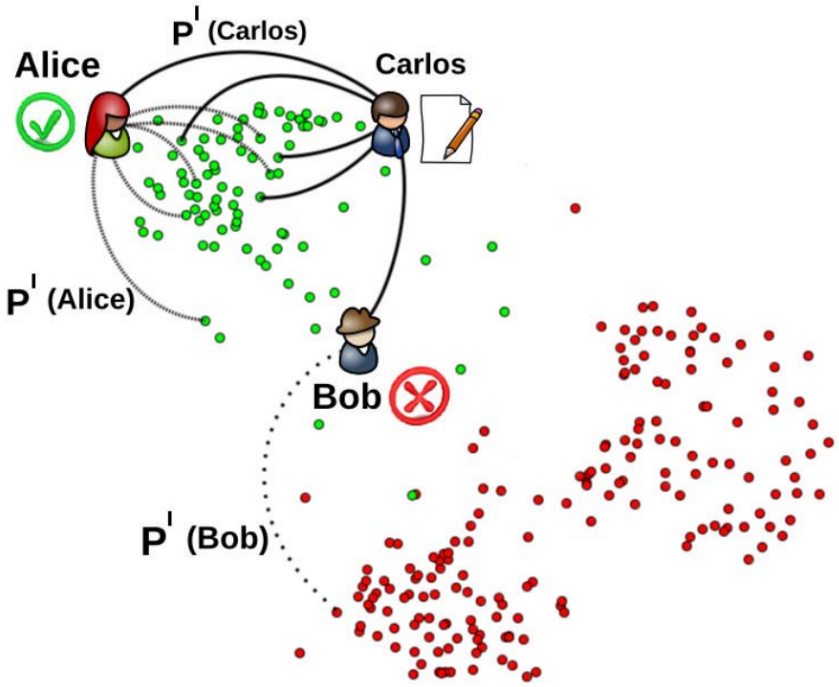



Figure 1.2: Orientation and amplification mechanism of BEEP (from [BFG+13]). Here, Bob receives an item from Carlos and does not like it, so he either discards it if the dislike counter has reached the TTL, otherwise he forwards it to someone from his RPS view that may like it.

BEEP is a dissemination protocol with two main mechanisms: *orientation* and *amplification*.

The goal of orientation is to forward news items to nodes that are likely to be interested in them, while amplification is designed to vary the number of dissemination targets according to the usefulness of the forwarding action. It follows the Susceptible, Infected, Removed (SIR) model [DM10]. A node receiving a new item applies updates as mentioned in the WUP protocol description, and then forwards the news item to a number of other nodes (*fanout*). If the item was already seen, then it simply drops the item.

The forwarding process is different depending on the like or dislike of the item. If the item is disliked, then the node increments the dislike counter field which acts as a TTL where if the dislike counter exceeds a threshold, the forwarding stops. If the TTL is still not reached, the node will pick a peer from its RPS view that has the most similar profile to the item profile and forward the item to it. The authors picked a fanout of 1 because it has interesting properties like serendipity and limiting the risk of propagating an uninteresting news item to too many nodes, but the algorithm still works with a fanout of more than one.

If the item is liked, then the node will forward it to random nodes from its WUP view, as nodes in the WUP view have a reasonably similar user profile with the current node. The random dissemination is here to avoid high levels of clusterization.

### Interactions betwen WUP and BEEP

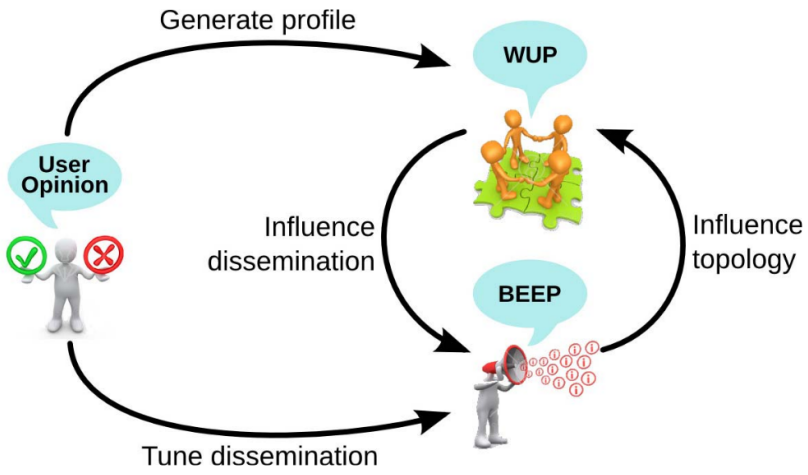



Figure 1.3: Interactions between the user opinion, WUP, and BEEP (from [BFG+13])

On Figure 1.3 we can see the interaction between the user and WhatsUp. When a user gives an opinion on an item, it allows WUP to add it to this user's profile; at the same time this influences dissemination through BEEP, as the item is not sent to the same set of users depending on this opinion. We can see that the two protocols have a mutual feedback loop, as they both influence each other.

## 1.6 Collaborative Filtering in WhatsUp

As explained in the previous sections, WhatsUp uses collaborative filtering to send recommendations based on the past behavior of the users, even for those who are not in the same neighborhood. However, it also has the same limitations as centralized collaborative filtering approaches, which means that it is subject to the same issue regarding content, as described in section 2.1.

# Chapter 2

# Problem Statement: investigating the effect of orientation

The goal of this internship is to try to improve the performance of a push-based distributed recommendation system (WhatsUp) through the use of an orientation mechanism.

**Definition 3.** An **orientation** mechanism is used to select the nodes an item will be broadcasted to. The opposite would be to send items at random or to all known nodes, without taking into account the knowledge acquired about the nodes in the neighborhood.

As explained in the Context section, WhatsUp integrates both an orientation/amplification mechanism, and a clustering mechanism. We chose to concentrate our efforts on the orientation part, but there is also ongoing work to apply content-based recommendation to the clustering part.

We investigate two solutions:

- A **profile-enhancement** solution, that tries to improve the efficiency of the similarity metric by providing more peers for the computation.
- A **content-based** solution, where the default similarity metric used for the orientation is replaced by a content-based similarity metric.

## 2.1 Profile enhancement

### 2.1.1 Principles

The rationale between "profile enhancement" is that improving the number of nodes in the item profile (the one conveyed with the item when it gets sent to another node) would help the orientation mechanism, as it would have more data to compute the similarity and therefore be more accurate.

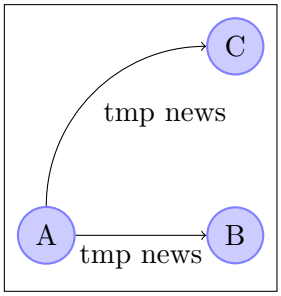
### 2.1.2 First-step enhancement

A very simple protocol for this mechanism could be the following:

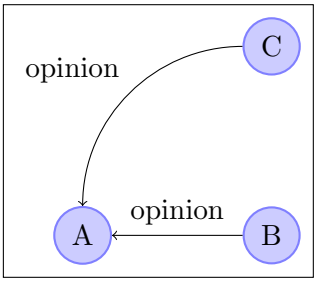
- The initial broadcaster for an item sends a "temporary" news item to nodes from its clustering view.

- The receiving nodes anwser with their opinion on the item (like/dislike).
- The initial broadcaster then sends them back a "final" news item, with the informations added to its news profile.
- The dissemination of the item then continues as it was done before.

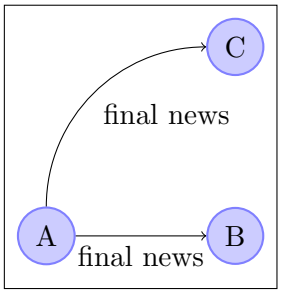
This protocol is straightforward and does not suppose churn or packet loss, and is therefore suitable for simulation purposes.



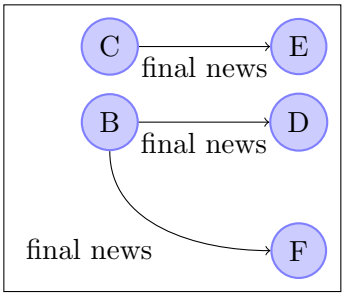

(a) A sends first a "temporary" news




(b) B and C send back their opinion




(c) A sends the final news to B and C

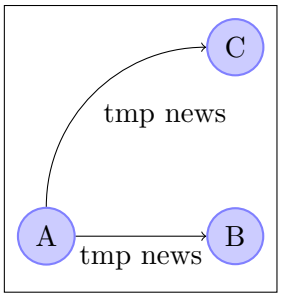



(d) B and C then broadcast the final news

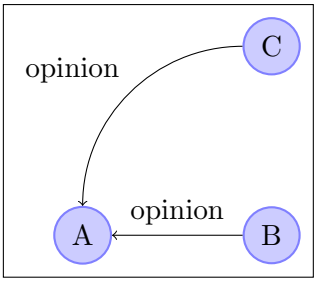
Figure 2.1: First-step enhancement

**Every-step enhancement**

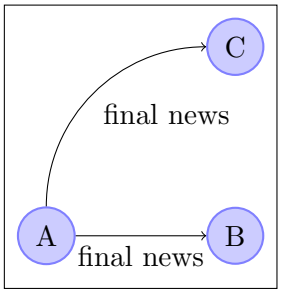
This protocol is a simple variation of the first one, with the last step being a new "temporary" item broadcast (thus restarting the process for new nodes) instead of a "final" news that would then proceed without this protocol.



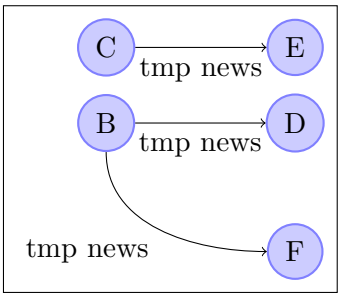

(a) A sends first a "temporary" news




(b) B and C send back their opinion




(c) A sends the final news to B and C




(d) B and C then broadcast a temporary news...

Figure 2.2: Every-step enhancement

## 2.2 Content-Based Recommendation

### 2.2.1 Principles

By default, WhatsUp uses a specific similarity metric to orient the messages according to the nodes profiles. This is an added layer on top of the clustering protocol, which already uses this metric to build the neighborhood.

$$Similarity(n, c) = \frac{sub(P_n, P_c) \cdot P_c}{\|sub(P_n, P_c)\|\|P_c\|}$$

Taking a node $n$ with a profile $P_n$ and a neighbor $c$ with a profile $P_c$, $sub(P_n, P_c)$ is defined as the subset of the scores in $P_n$ corresponding to items in $P_c$. We therefore substitute this metric with a custom one using TF-IDF with the Cosine similarity to get a score for a user and a news item based on its metadata.

We have several choices to pick from for themetadata:

- Use the tags (keywords) of the news items.
  Pros:
  - Fast (small diversity and less computations).
  - Distinguishable categories.
  - Easy to inspect.

  Cons:

  - Not fine-grained: depending on the number of categories, they might be very wide and with plenty of topics grouped under one keyword.
  - Lack of precision.

- Use the whole body of the news items.
  Pros:

  - Better coverage of topics, more words

  Cons:

  - Slow depending on the type of news items and the cache kept.

- Use the title of the news article.
  Pros:

  - Small, ideal for computations.

  Cons:

  - Could offer less precision than the tags, due to the sparse nature of the data.

### 2.2.2 Implementation

We leveraged the Recommender101 [JLGB13] and the Word Vector Tool [Wur99] to build our implementation of content-based orientation inside the WhatsUp simulator.

# Chapter 3

# Evaluation

## 3.1 Data Sets

Several datasets were used to evaluate WhatsUp in the original paper. However, most of those were extracted with collaborative filtering in mind, and cannot be re-used when considering content-based orientation, because they only contain article identifiers and user preferences, with no indication of their content whatsoever.

For this evaluation, we therefore took two datasets that we use for content-based recommendation.

### 3.1.1 MovieLens

MovieLens is a group of data sets from the movielens website [Gro] put together by the grouplens project. It contains ratings of users on a huge number of movies, and includes details about the users and metadata on the movies too. We can therefore perform recommendation based on the movie category; however, this data set only provides the guarantee that each user has rated at least twenty movies, but no user has rated all the movies at once, so we have to take extra steps for the simulation.

For simulation purposes, we have to assume that a user who didn't rate a movie wouldn't have liked it. Considering how the orientation is done on items, the results on both recall and precision will be abysmal compared to normal datasets. It means we can only compare those results with the original WhatsUp on the same data set.

### 3.1.2 Survey

This dataset is a survey made on 130 users for 207 news items. This survey was broadcasted at a small scale, originating from the team members and sent to acquaintances. Each user had to say if he was interested in a news article or a website, based on its title. The advantage of this dataset is that it is complete (we know each user preference for each article, so we can get more realistic results from the simulation), but it is also very small, (compared to large-scale samples like MovieLens) and therefore potentially biaised and not necessarily accurate at a wider scale.

### 3.1.3 Dataset summary

Below is a quick summary of the datasets used for our experiments. The big MovieLens dataset (71567 users/10681 movies) was considered but its size made it unpractical for fast simulations, without much justification that the results would be different from the two smaller ones.

Table 3.1: Dataset summary

| Data Set Name | Number of Users | Number of Items | Complete ratings |
|---|---|---|---|
| Survey | 130 | 207 | Yes |
| MovieLens (small) | 943 | 1682 | No |
| MovieLens (medium) | 6040 | 3883 | No |

## 3.2 Experimental protocol

### 3.2.1 Evaluation metrics

To evaluate the quality of the results, we consider three main metrics, well-known [vR79] in the domain of information systems:

- Recall (completeness) =

$$\frac{|\{interested\ users\} \cap \{reached\ users\}|}{|\{interested\ users\}|}$$

- Precision (accuracy) =

$$\frac{|\{interested\ users\} \cap \{reached\ users\}|}{|\{reached\ users\}|}$$

- F1-score =

$$2 \cdot \frac{precision \cdot recall}{precision + recall}$$

### 3.2.2 Similarity metric

When using the default WhatsUp behavior, we used the WhatsUp similarity metric defined in the first chapter:

Taking a node $n$ with a profile $P_n$ and a neighbor $c$ with a profile $P_c$, $sub(P_n, P_c)$ is defined as the subset of the scores in $P_n$ corresponding to items in $P_c$. The spam limiter is achieved by dividing this metric with the number liked by $c$ in this subset. The metric is as follows:

$$Similarity(n, c) = \frac{sub(P_n, P_c) \cdot P_c}{\|sub(P_n, P_c)\|\|P_c\|}$$

### 3.2.3 Experimentation platform

The experiments were ran my personal machine, an Intel T7300 with 4GB or RAM with the small experiments, and after that on the Igrida [Igr] cluster at INRIA Rennes.

The simulation is using a java-based simulator developed by the team to reproduce the behavior of a cycle-based distributed system. It can use a variable number of threads depending of the machine it runs on and it is therefore suitable for the simulations on the cluster.

### 3.2.4 Experimental parameters

The simulator used as extensive options, and the following ones were used for the experiments:

- Cluster fanout: The number of times an item will be forwarded to the nodes in the neighborhood if the user liked it. This parameter is has a changing values in the simulations because its effect are impacted by the orientation used.
- Cluster size: Number of node to keep in memory for the neighborhood; this is set to $2 \cdot clusterfanout$ in the simulations, as defined in [BFG+13]
- Orientation on like: orientation is done in the neighborhood when the user likes a new item. This parameter is evaluated for both values on each setting.
- Orientation on dislike: orientation is done in the random view when the user dislikes a news item. This parameter is evaluated for both values on each setting.
- Random view fanout for like: By default, WhatsUp does not forward the liked news to the random view of the network. Since our intent is to investigate the effect of orientation, we do not modify this setting.
- Dislike TTL: The number of times an item can be disliked before it is dropped from the network.

| Parameter name | Parameter value |
|---|---|
| dislike TTL | 4 |
| random view fanout for like | 0 |

Table 3.2: Static parameters

## 3.3 Results for profile enhancement

The profile enhancement mechanism was only tested with the Survey and various other datasets that do not contain the information necessary for content-based orientation.

### 3.3.1 Each-step profile enhancement

On figure 3.1, we can see that the enhancement mechanism on each step does not improve the values of the F1-score compared to the default WhatsUp behavior (no enhancement). In fact, the value is even lower than without enhancing the profiles. Figures 3.9a and 3.9b provided in the Appendix also show that there is a slight decrease in precision, and an improvement in recall, but not enough to obtain a better F1-score.

Our supposition is that adding the items at each cycle does not provide a substantive imrpovement over time, as the news items only retain a limited number of users in its profile. This means adding more nodes to this profile at each cycle will increase the rate at which users are removed from the profile, and lead to overall instability in the profile.

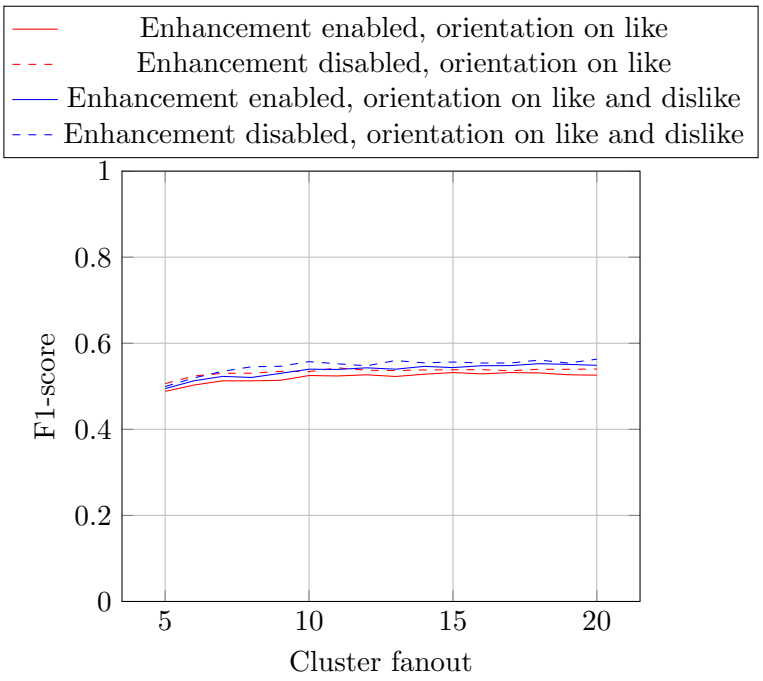



Figure 3.1: Impact of the F1-score with default behavior and the enhancement at each step

### 3.3.2 First-step profile enhancement

On Figure 3.2 we see that the F1-score is still lower, but does not differ as much from the baseline as the enhancement at each step. Figures 3.10a and 3.10b are also provided in the Appendix to show the effect on recall and precision.

We suppose that the gain of the first-step enhancement is not significant enough to make a visible trace on the results, as the item profile gets to that level of completeness after a few cycles.

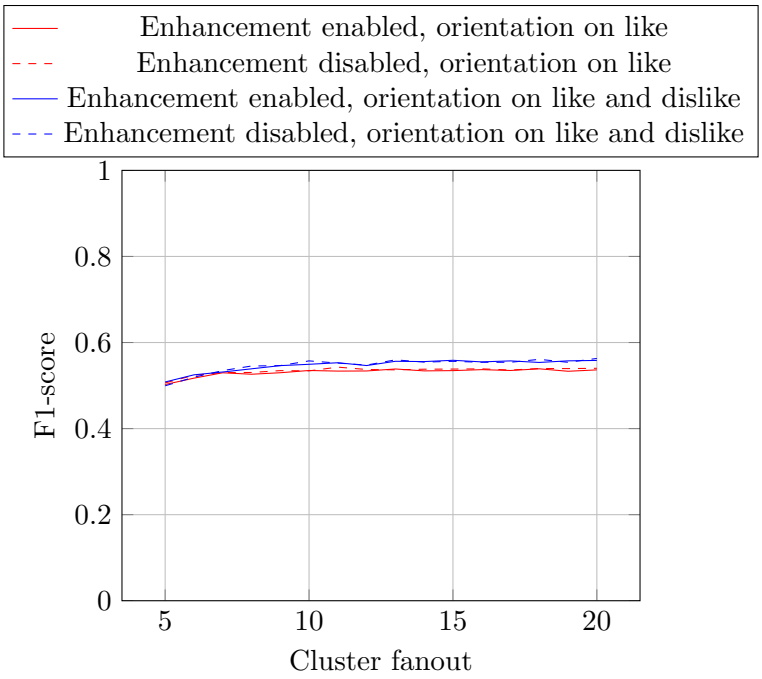



Figure 3.2: Impact the F1-score of using the profile enhancement at the first step

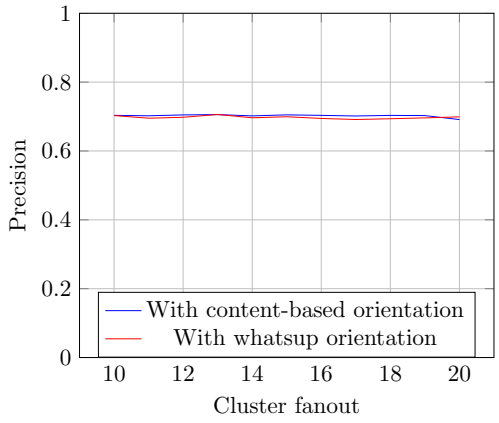
## 3.4 Results for content-based orientation

### 3.4.1 On MovieLens

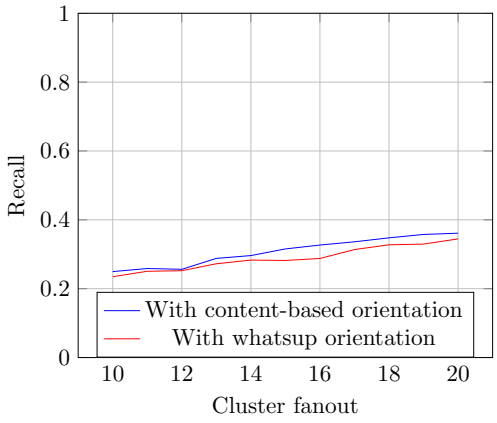
For the MovieLens dataset, we only considered the tags of the movies (there is only around twenty keywords available no matter what dataset is used), as only the tags were readily available, and retrieving more information from online databases (such as the synopsis) might be error-prone.

#### Small dataset: 100k ratings

Figure 3.3a and 3.3b show that the content-based approach works marginally better than the original profile-based similarity both in recall and precision. With 3.4 we can see that the small increase on recall improves the F1-score of the content-based orientation version, and gives slightly better results than the whatsup orientation.




(a) Precision with orientation on both like and dislike




(b) Recall with orientation on like and dislike

Figure 3.3: Orientation and recall comparing Whatsup similarity and content-based

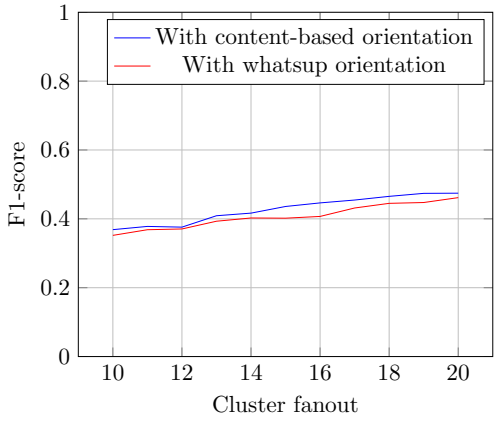
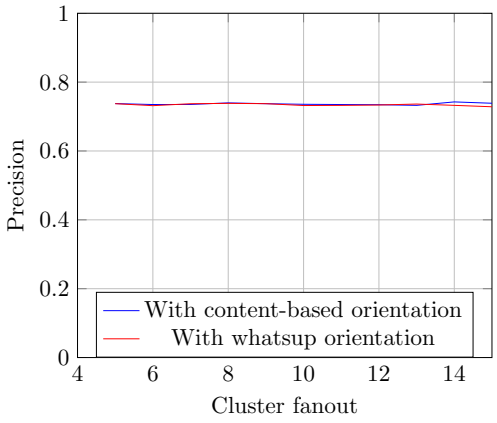



Figure 3.4: F1-score with orientation on both like and dislike

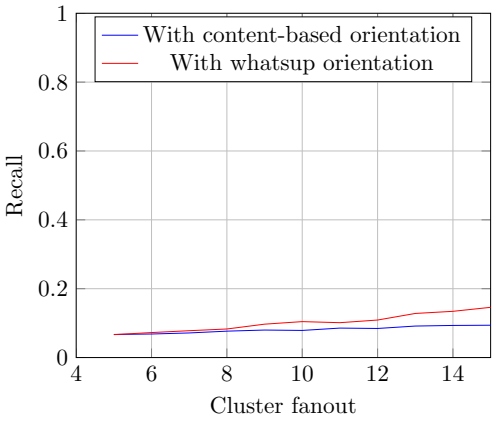
### Medium dataset: 1M ratings

On figures 3.5a and 3.5b, showing the recall and precision of both the content-based orientation and the profile-based one on the MovieLens data set containing one million ratings. The recall has this time a lower value with the content-based orientation, and this shows on Figure 3.6 which displays the F1-score.

The results are poor, just as with the smaller MovieLens data set, which is explained by the sparsity of the profiles. Our assumption is that the content-based orientation mechanism is hurt a lot more by the sparsity of the results because WhatsUp still uses the profile-based similarity for clustering. This means that the users will be clustered according to this metric, and have a better chance of being recommended items through this metric.




(a) Precision with orientation on like and dislike




(b) Recall with orientation on like and dislike

Figure 3.5: Orientation and recall comparing Whatsup similarity and content-based

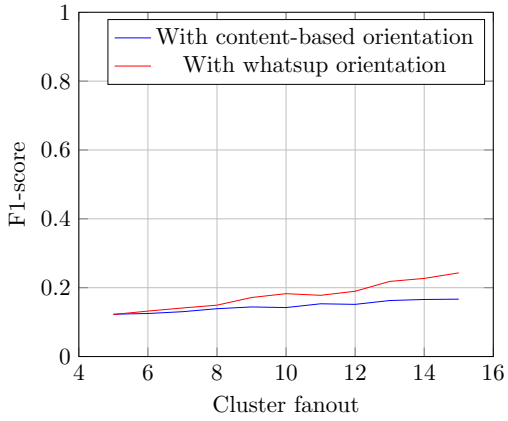



Figure 3.6: F1-score for orientation on like and dislike

### 3.4.2 On the Survey

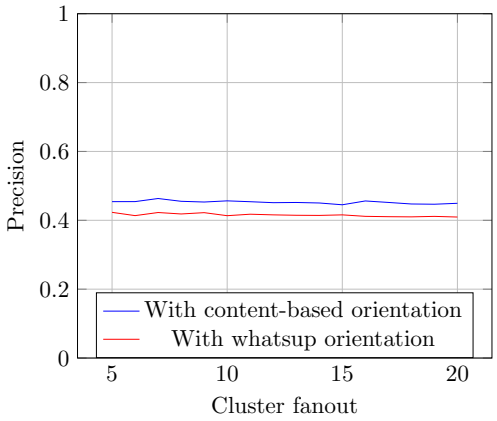
For the survey, the experiments were ran on the three possible sets of data available:

- The tags (keywords)
- The article title

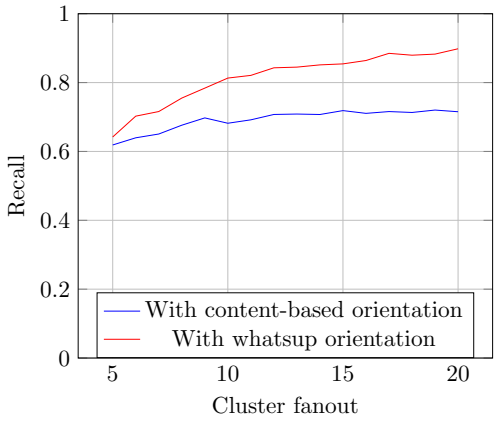
Doing content-based recommendation on the article title was considered, but it would have required writing a custom parser for every different website which had a news in the data set. This is, with the subject of WhatsUp (diversified news) an issue difficult to deal with, as we cannot make suppositions about the websites an user will submit to the system. This might however be doable with a restricted scope (for example, doing recommendation on news or items of a specific website).

### On the title

Figure 3.7 shows the precision and recall obtained by doing content-based orientation using the title of each item to perform the recommendation. We can see here that there is a net increase in precision compared to the original similarity metric, which is accompanied with a sharp decrease in recall. The F1-score plots (figures 3.11a and 3.11b) given in the appendix show that those two changes negated each other and that the overall curves stayed the same. Basu et al. [BHC+98] suggest that precision is more important than recall to the user experience, as long as the recall stays above a certain threshold.




(a) Precision with orientation on like and dislike

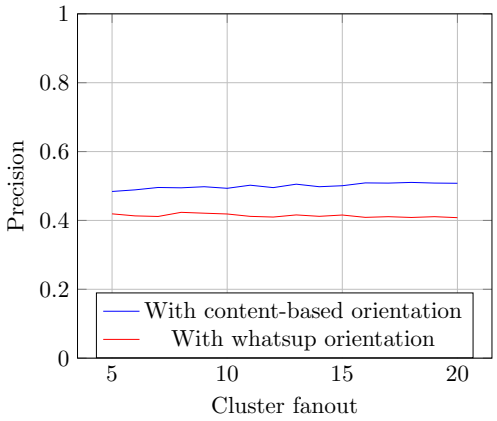



(b) Recall with orientation on like and dislike

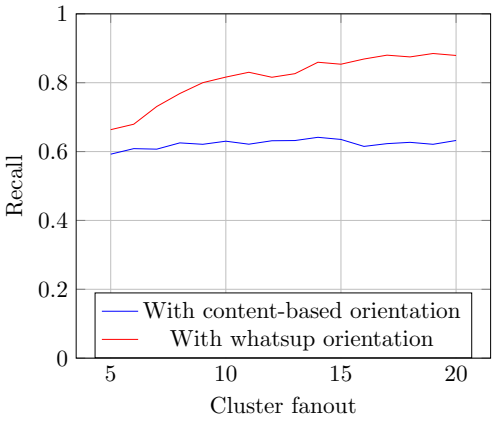
Figure 3.7: Orientation and recall comparing Whatsup similarity and content-based

### On the keywords

The divergence for recall and precision between the content-based approach and the baseline is confirmed with the use of the keywords for recommendation, as shown in Figure 3.8a and 3.8b. The increase in precision is even more visible in this case, but while Figure 3.7b seemed to indicate an increase in recall according to the cluster fanout when using the news title, it appears this is not the case here, as the recall mostly stays at the same value. We can also see that the F1-score (Figures 3.12a and 3.12b in the Appendix) stays stable, with the content-based solution getting a slight edge over the baseline.




(a) Precision with orientation on like and dislike




(b) Recall with orientation on like and dislike

Figure 3.8: Orientation and recall comparing Whatsup similarity and content-based

# Conclusion and Future Work

During this internship, we have evaluated two possible ways of improving the orientation mechanism in a push-based distributed recommendation system.

With the content-based approach, we managed to improve the precision of the system by a noticeable margin, while keeping an equal or better F1-score. This results are encouraging though mitigated by the low performance of the algorithm on a well-accepted data set, and we therefore need to expand the work to another, more complete data set.

### Future work

The issue with the two evaluated data sets is that they don't apply well to our issue: our desired setting for content-based recommendation is a set of distinct topics. The easiest way to represent such topic discrepancy is to have articles in a different language. Moreover, for realistic results, our simulation requires that each simulated user can give an opinion on each news, unlike in the MovieLens data set. For this reason, we have begun the creation of a new data set, using the same methodology as the first survey. We will hopefully be able to present results later using this data set.

### Acknowledgements

I would like to thank my supervisor Davide Frey for his guidance during this internship, and Arnaud Jégou, whose assistance helped me on many technical subjects. I also want to extend my thanks to the whole ASAP team who made my internship enjoyable.

# Bibliography


[ABG+07] Jae-wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, and Sue Yeon Syn. Open user profiles for adaptive news systems: help or harm? In *Proceedings of the 16th international conference on World Wide Web*, pages 11–20. ACM, 2007.

[AVS04] Kamal Ali and Wijnand Van Stam. Tivo: making show recommendations using a distributed collaborative filtering architecture. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 394–401. ACM, 2004.

[BFG+13] Antoine Boutet, Davide Frey, Rachid Guerraoui, Arnaud Jegou, and Anne-Marie Kermarrec. Whatsup: A decentralized instant news recommender. In *IEEE 27th International Symposium on Parallel & Distributed Processing*. IEEE, IEEE, 2013.

[BHC+98] Chumki Basu, Haym Hirsh, William Cohen, et al. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720, 1998.

[BHK98] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[BS97] Marko Balabanović and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, March 1997.

[Can02] John Canny. Collaborative filtering with privacy. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 45–57. IEEE, 2002.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European transactions on Telecommunications*, 8(5):481–490, 1997.

[CH97] David Maxwell Chickering and David Heckerman. Efficient approximations for the marginal likelihood of bayesian networks with hidden variables. *Machine Learning*, 29(2-3):181–212, 1997.

[CHW01] Sonny Han Seng Chee, Jiawei Han, and Ke Wang. Rectree: An efficient collaborative filtering method. In *Data Warehousing and Knowledge Discovery*, pages 141–151. Springer, 2001.

[DGH+87] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, New York, NY, USA, 1987. ACM Press.

[DM10] Moez Draief and Laurent Massouli. *Epidemics and rumours in complex networks*. Cambridge University Press, 2010.

[FLNP00] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620, 2000.

[Gro] GroupLens. Movielens. `http://grouplens.org/datasets/movielens/`.

[HKBR99] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[HW79] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[HXYS04] Peng Han, Bo Xie, Fan Yang, and Ruimin Shen. A scalable p2p recommender system based on distributed collaborative filtering. *Expert systems with applications*, 27(2):203–210, 2004.

[Igr] Igrida. `http://igrida.gforge.inria.fr/`.

[JLGB13] D. Jannach, L. Lerche, F. Gedikli, and G. Bonnin. What recommenders recommend - an analysis of accuracy, popularity, and sales diversity effects. In *Proc. 21st International Conference on User Modeling, Adaptation and Personalization (UMAP 2013)*, Rome, Italy, 2013.

[JVG+07] Márk Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. Gossip-based peer sampling. *ACM Trans. Comput. Syst.*, 25(3), August 2007.

[KSJ09] Ioannis Konstas, Vassilios Stathopoulos, and Joemon M Jose. On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 195–202. ACM, 2009.

[L+95] Henry Lieberman et al. Letizia: An agent that assists web browsing. *IJCAI (1)*, 1995:924–929, 1995.

[LGS11] Pasquale Lops, Marco Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[MKR04] Bradley N Miller, Joseph A Konstan, and John Riedl. Pocketlens: Toward a personal recommender system. *ACM Transactions on Information Systems (TOIS)*, 22(3):437–476, 2004.

[Mou97] Alexandros Moukas. Amalthaea information discovery and filtering using a multiagent evolving ecosystem. *Applied Artificial Intelligence*, 11(5):437–457, 1997.

[MW99] Dunja Mladenic and Browsing Assistant Personal Webwatcher. Machine learning used by personal webwatcher. 1999.

[PB97] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.

[Pea01] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[PHLG00] David M Pennock, Eric Horvitz, Steve Lawrence, and C Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 473–480. Morgan Kaufmann Publishers Inc., 2000.

[PNSK+06] Tan Pang-Ning, Michael Steinbach, Vipin Kumar, et al. Introduction to data mining. In *Library of Congress*, 2006.

[RFI02] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *arXiv preprint cs/0209028*, 2002.

[RIS+94] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[Roc71] Joseph John Rocchio. Relevance feedback in information retrieval. 1971.

[Sal89] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of*. Addison-Wesley, 1989.

[SFHS07] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[SK09] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[SMK+01] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 149–160. ACM, 2001.

[SPUP02a] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[SPUP02b] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[Tri10] Tribler. Tribler. `http://www.tribler.org`, 2010.

[UF98] Lyle H Ungar and Dean P Foster. Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems*, number 1, 1998.

[Use] Usenet. `https://en.wikipedia.org/wiki/Usenet`.

[vR79] CJ van Rijsbergen. Information retrieval (2nd edit.) butterworths. *London, UK*, 1979.

[VVS05] Spyros Voulgaris and Maarten Van Steen. Epidemic-style management of semantic overlays for content-based searching. In *Euro-Par 2005 Parallel Processing*, pages 1143–1152. Springer, 2005.

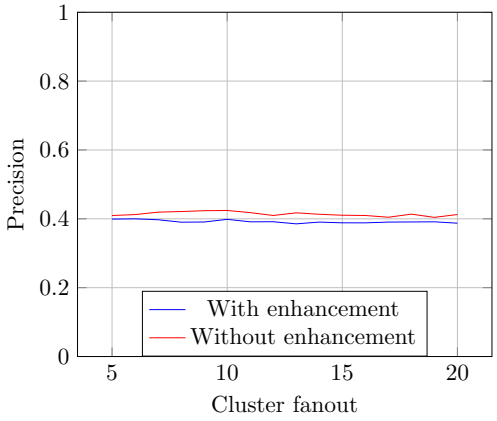
[Wur99] Michael Wurst. Word vector tool. `http://sourceforge.net/projects/wvtool/`, 1999.

[XLY+05] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121. ACM, 2005.

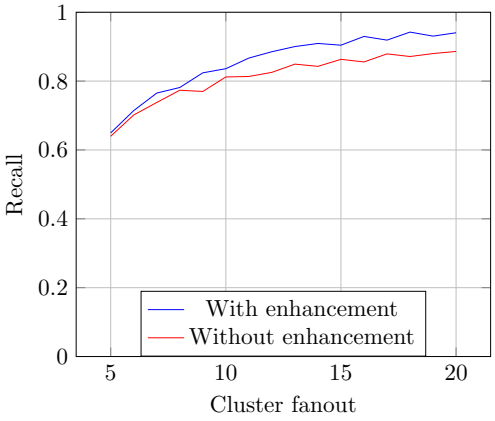
[ZFF11] Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2011.

# Appendix

## Precision and recall for each-step enhancement



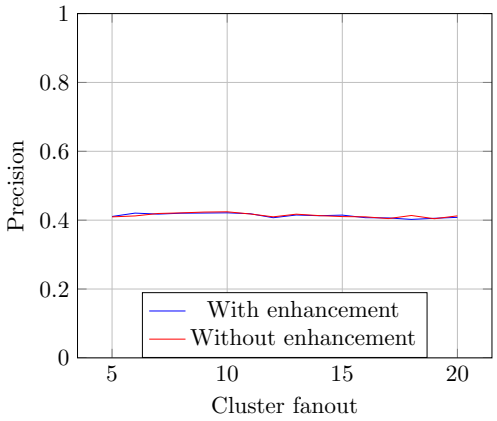

(a) Precision with orientation on like and dislike



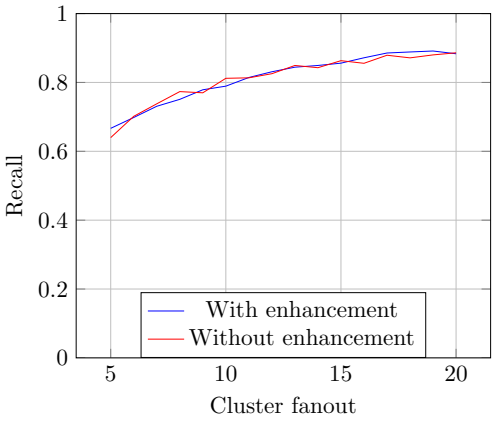

(b) Recall with orientation on like and dislike

Figure 3.9: Orientation and recall with and without each-step enhancement

## Precision and recall for first-step enhancement



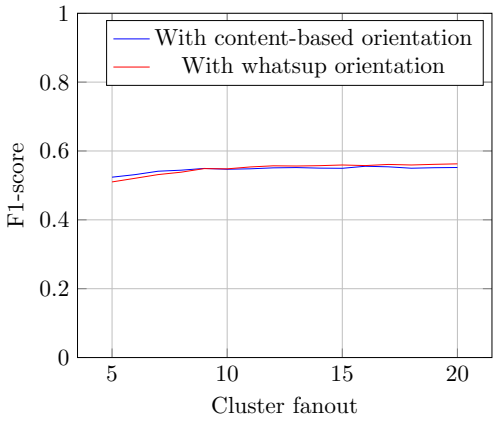

(a) Precision with orientation on like and dislike



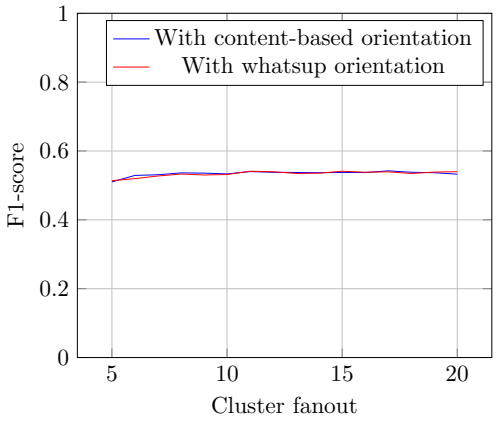

(b) Recall with orientation on like and dislike

Figure 3.10: Orientation and recall with and without first-step enhancement

## F1-score for orientation on the survey title



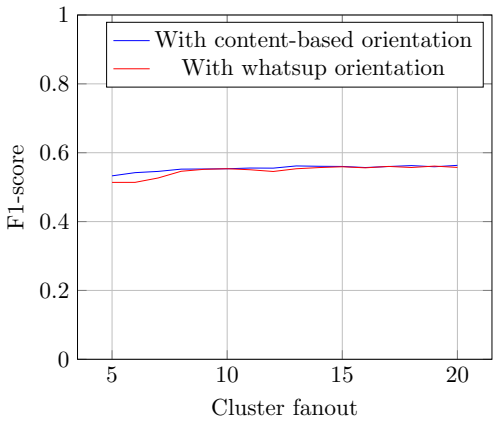

(a) Orientation on like and dislike



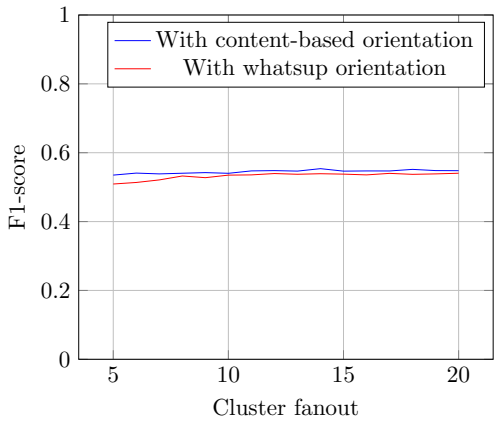

(b) Orientation on like

Figure 3.11: F1-score for different values of the orientation parameters

## F1-score for orientation on the survey keywords




(a) Orientation on like and dislike




(b) Orientation on like

Figure 3.12: F1-score for different values of the orientation parameters