Reveal: Fine-grained Recommendations in Online Social Networks

Markos Aivazoglou[1], Orestis Roussos[1], Sotiris Ioannidis[1], Dimitris Spiliotopoulos[2], and Jason Polakis[3]
[1]FORTH, email: {markos, roussos, sotiris}@ics.forth.gr
[2]University of Houston, email: dspiliotopoulos@uh.edu
[3]University of Illinois at Chicago, email: polakis@uic.edu

***Abstract*—Content selection in social networks is driven by numerous extraneous factors that can result in the loss of content of interest. In this paper we present Reveal, a fine-grained recommender system for social networks, designed to recommend media content posted by the user's friends. The intuition is to leverage the abundance of pre-existing information and identify overlapping user interests in specific sub-categories. While our system is intended as a component of the social network, we develop a proof-of-concept implementation for Facebook and experimentally evaluate the effectiveness of our approach.**

I. Introduction

A critical aspect of the popularity of social networks, which also drives user engagement, is the sharing of content among friends. However, the massive amount of content published can result in users missing content of interest, due to the overall noise. Furthermore, the content shown in Facebook's News Feed is heavily influenced by the user's relationship to the friend that published it [1]; thus, "weak" social ties can lead to the loss of content of interest.

In this paper we propose Reveal, a recommender system that exists within the social network that categorizes content and offers suggestions based on the interest similarities between the user and the friend that published the content. Specifically, Reveal processes all content published by the user's friends, identifies content from different categories (e.g., music), and then collects information to assign that content to a more specific sub-category (e.g., music (sub)genres). Subsequently, the system analyzes accompanying text to identify the sentiment regarding that content and infer if a positive or negative opinion was expressed. Based on the user's *interest profile*, and the *similarity score* with that friend, Reveal determines if the content should be suggested to the user or not.

Previous work has focused on like objects for recommending content [2]. Facebook has also implemented a recommender system that suggests content liked by friends. Our system follows a different approach; it aims to filter through the massive amount of content that is posted by a user's friends, and select posts most likely to be of interest to the user. Another key concept behind our system that differentiates it from other recommender systems, is that it enables users to leverage their existing knowledge regarding the overlapping interests they have with their online friends, in a fine-grained manner. While certain users may have very similar tastes in a specific subcategory, they might have completely different interests in other categories. It is important to note that our system is not intended as a replacement for existing content selection algorithms for users' News Feeds; instead we aim to identify the most interesting items (e.g., Youtube videos) posted by a user's friends, which may be otherwise lost amidst the massive amount of generated content. These selected posts are to be presented in a separate recommendations section, each dedicated to a specific category, thus expanding the existing functionality of the social network, and improving the user experience.

II. Methodology

Our system consists of three main components:

Backend. The main component comprises the *backend*, and functionality can be broken down to three tasks:

Bootstrapping, where every like under categories of interest in the users' and their friends' profiles are analyzed, for extracting entities and conducting a fine-grained categorization. This data is then leveraged for creating users' *interest profiles*.

Similarity calculation and score tweaking, in which Reveal compares the users' *interest profile* to that of their friends and assigns a *similarity score* to each one. At this stage, the users can manually tweak any *similarity scores* to their liking, thus modifying the weights for specific sub-categories or friends. Essentially, this step builds the required social knowledge that enables the system to later accurately identify content that matches the users' interests.

Post analysis and list generation, for processing the text accompanying content published by the user's friends. This step extracts semantic information from posts and infers whether the poster has expressed a positive or negative opinion about the media content. This allows Reveal to assert whether it overlaps with the user's categories of interest, to calculate its significance and, ultimately, to decide whether it should be recommended to the user. To reduce the latency introduced by accessing large collections of online data, we created an offline knowledge base containing entities of interest, also acting as a data cache to avoid excessive network traffic and operational overhead delays. Our entity extraction mechanism also leverages online resources for data not found locally.

Housekeeping. This component consists of two distinct modules with separate functionalities, which are executed periodically. One module refreshes the recommendation lists for each user so as to contain fresh content. The other module, which is executed less frequently, polls users' profiles to

TABLE I. List of notations used throughout paper.

	Variable	Description
GS	GenreScore	Genres in user's/friend's set of likes
GSL	GenreScoreList	List of genre scores for a user/friend
OGS	OveralGenreScore	Sum of genre scores in GSL
LS	LikeScore	GS avg for genres corresponding to like
LSL	LikeScoreList	List of LS entries for a user/friend
OLS	OveralLikeScore	Sum of like scores in LS
GSS	GenreSimilarityScore	Genre similarity between user&friend
LSS	LikeSimilarityScore	Like similarity between user&friend
FSS	FriendSimilarityScore	GSS/LSS similarity between user&friend
PS	PostScore	Rank of a friend's post
PG	PostGenres	Genres extracted from a post

account for any changes made to their likes and interests. This is done for recalculating necessary scores and weights based on new information.

Frontend. This is the interactive component of our app, where users can view the lists of recommended content, or tweak the weight of their friends to better reflect the desired similarity score for specific (sub)genres.

A. Bootstrapping

This component is responsible for the entire *bootstrapping* phase, where the required data about the active user and his or her friends (hereby referred to as a *clique*), is collected and processed. We create *interest profiles* from *likes* found on their personal Facebook profiles, namely, under the Music and Movies endpoints [3]. Furthermore, in order to provide fine-grained recommendations, accurate and detailed *genre* identification for each *like* is crucial. Consulting our Knowledge Base, we extract entities and information about their genres[1] and categories. Table I explains our notations.

Genre and like scoring. Facebook allows users to express their interests through *likes* associated with specific objects (pages, posts, etc). However, not all *likes* are equally representative of the user's preferences, and identifying which should have more contribution during the content selection process is critical for providing accurate recommendations. By analyzing all of the user's *likes*, we calculate the significance of all identified genres for the user. Each *like* represents an entity which corresponds to a set of associated *genres*, as specified by our Knowledge Base. We create *GSL* for every user in the *clique*, a key-value table where each *genre* is the key and the number of occurrences throughout all *likes*, is the value.

To normalize the genre similarity between a user and his or her friends at a later stage, we also calculate an overall score of genres for each of them as shown in Equation 1.

$$OGS = \sum_{i \in GSL} GS(i) \tag{1}$$

where $GS(i)$ is the genre score for a specific entry in *GSL*.

In addition, we create a *LSL* for every *clique* member's *likes*, to weight them using the *GS* values in *GSL* calculated above. *LSL* is an average calculation of the *GS* corresponding to the genres found both in the *like* item and in *GSL*. This metric allows us to weigh `likes` according to the *genre* preferences of the user. The *LS* scores in *LSL* are used for calculating similarity with other users. We create an overall `Like` score,

$$OLS = \sum_{j \in LSL} LS(j) \tag{2}$$

used later for normalization when calculating *like* similarity.

As a fallback option, if there is not enough information for a specific user, we crawl through their post history and attempt to identify and extract entities for creating their *interest profile*.

B. Sentiment Analysis

Posts are first processed by our sentiment analysis module, and those with negative polarity are discarded. If there is no accompanying text we assume that a positive polarity is implied, the intuition being that users are implicitly recommending them.

Analyser. A crucial aspect of recommendations, especially when natural language is involved, is detecting the reviewer's point-of-view on the topics and subjects they post; for instance, the text that accompanies links shared on Facebook. Sentiment analysis defines the method of discerning the positive feeling (attractiveness) or negative feeling (aversiveness) in text, and has been extensively explored in various contexts [4], [5], [6], [7]. `Reveal` needs the ability to discern the poster's opinion, for providing accurate recommendations, and leverages a modified version of the SO-CAL approach introduced by Taboada et al. [8]. SO-CAL extracts sentiment polarity and strength from text, and consists of the proposed algorithm and a set of dictionaries categorized by their part-of-speech. There are 6 different dictionaries in the set containing 1542 Nouns, 1142 Verbs, 2824 Adjectives, 876 adverbs, and 217 Valence shifters. We opted for SO-CAL due to its ranked dictionaries with words scored with sentiment intensity (valence) for fine-grained sentiment tagging, and also the scoring heuristics which performed better than other dictionary-based approaches we tested. Additionally, we made some modifications, such as changing the SO value of certain words, simplifying the negation lexicon, and adding an emoticon lexicon (110 entries) that we created manually using information from Wikipedia.

Text pre-processing. We filter out Facebook tags (with @) and non-English or non-printable characters. We use the NLTK toolkit for part-of-speech (POS) tagging and the Pattern module [9] for text processing (splitting and tokenization).

Semantic strength tagging using dictionaries. SO-CAL uses dictionaries with words grouped by their Part-of-Speech, ranked with valence strength (in the range $[-5,0) \cup (0,5]$). Tagging is necessary as a word may be defined with different POS, which results in different valence strength. Additionally, as the use of emoticons is widespread in social media, we rank and handle them accordingly.

Valence shifters. Valence shifters are words that carry different semantic values than the words described so far, and their POS-tag does not necessarily affect their use. They are called shifters as they change the strength or effect of a lexical item when they are nearby. Their area of effect is limited, which is defined by various grammatical properties. Each valence shifter is assigned an SO value, although it is applied differently on the score calculation. Specifically, it

[1]For simplicity, *genre* will denote fine-grained sub-genre information.

works as an additive multiplier on the initial SO value of the lexical item it shifts. The default multiplier for words is 1.

Negation. We apply *shift negation*, where a term's SO value is shifted towards the opposite polarity by a fixed amount.

Scoring. Valence shifters and negators are applied as modifiers to the SO value. To calculate the final SO value, we recursively apply any modifier value found searching backwards, until a determiner (e.g., a *comma* or sentence connective) is found. This calculation is applied to every lexical term and the sum of every sentence's SO value is aggregated, producing the total SO value of a given text.

Irrealis blocking [8], [10], is used to describe situations where something has not happened yet as the speaker is talking and, thus, the result or action is uncertain. This contains subjunctive, conditional and imperative moods which can be detected by a pattern module. Some of them, such as the imperative mood, may be semantically significant in our case, as it can be used to express sentiment upon a subject. That means, words in the effective radius of an irrealis marker (e.g., modals and quoted sentences), have a nullified SO value. We also ignore sentences that are categorized as questions.

C. Entity Extraction

To gather information about the friends' posts, we have to analyze their entire activity. Our goal is to define all the entities in a friend's post and keep those that belong to categories of interest. `Reveal` utilizes 3 endpoints (Facebook *edges*) from the Facebook Graph API to obtain the needed data, *Links* which are posts containing a URL, and two kinds of *Actions* which are posts with user generated social stories [11]. Specifically, we use the *listening* and *watching* social stories.

We also leverage the Freebase [12] collection for creating a knowledge base of entities. Google's Knowledge Graph Search API was partly powered by Freebase, which was recently deprecated and replaced by the Graph API. We extracted every entity under musical artists/bands and movies coupled with detailed genre information, through their API. Specifically, we obtained 293,506 unique entities, 221,091 movie entities and 72,415 musical artist/bands entities paired with genres, as `JSON` objects. Each entity in the knowledge base carries a unique `Topic ID` as provided by Freebase, that we use for entity matching with Youtube videos. We developed a dictionary-based entity extractor, that utilizes resources from our knowledge base. In addition, to swiftly map a video to its respective topic (entity) we developed an indexer that maps each `Topic ID` to its entity and every corresponding `Video ID` found in Youtube at that time. This resource serves as a data cache that enables fast named entity recognition, without minimal reliance on online APIs, and also reduces excessive network traffic and delays due to operational overhead.

Links. Facebook *Links* are posts that contain a link/URL. To extract entities, we obtain each URL by calling the appropriate Graph API endpoint, filter out non-Youtube URLs, and extract the Video ID. We match potential entities using the aforementioned `Video ID`-to-`Topic ID` mechanism.

Our entities in the knowledge base are indexed with their unique `Topic ID`, enabling us to directly extract any of them that is related to a given Youtube video.

Actions. Entity recognition in these posts is straightforward, as the name or title of the item is found within the post's data. Since Facebook prohibits fetching comments accompanying Actions, again we assume they are positive recommendations. However, as `Reveal` is designed to be deployed by Facebook, in practice it would also have access to the comments.

D. Quantifying Similarity

A challenging aspect of the content recommendation process, is selecting the similarity formula that will quantify how interesting a specific post will be to the user. In previous work [2] the proposed system used an interactive graph, where item and friend weights were assigned manually and required user interaction even in the initial stage. Scores were also in a capped scale (1 to 5 inclusive), thus, not being as fine-grained as needed. To that end, we opted to employ the Jaccard coefficient, and created a formula that is well suited for automatic similarity calculation in sets with weighted items and of arbitrary size. Specifically, we devised a formula that contains certain modifications to the Jaccard coefficient, as we describe next. To calculate the similarity among the users and their friends in a fine-grained manner, we leverage the *genre* and *like* scores calculated in the bootstrap phase; our in-depth profiling of each user's interests, enables us to develop an accurate mechanism for "scoring" friends and posts.

To calculate genre similarity score between a user and a friend, we aggregate the sum of scores of each overlapping *genre* and then divide by the sum of their *OGS* to normalize.

$$GSS(U,F) = \sum_{i \in GSL(U) \cup GSL(F)} \frac{GS_U(i) + GS_F(i)}{OGS_U + OGS_F} \quad (3)$$

The same principle applies for the *LSL* table.

$$LSS(U,F) = \sum_{i \in LSL(U) \cup LSL(F)} \frac{LS_U(i) + LS_F(i)}{OLS_U + OLS_F} \quad (4)$$

Using (3) and (4), we calculate the FSS that the (active) user has with a friend in either category, which represents the actual overlap between them and is used for post score calculation in a later stage. Based on our initial observations, we found that applying weight to the equation would increase the accuracy and the precision of our results. Therefore, we experimented by applying different weights while running predefined use cases, which lead us to our approach. We find that a weighted average is suitable for treating content classification as a defining factor, and manual testing showed us that giving genre overlap double the weight boosted accuracy in a more fine-grained fashion.

$$FSS = \frac{2 * GSS + LSS}{3} \quad (5)$$

Finally, to populate the recommendation lists we gather all the posts contained in the user's friends' profiles and apply the formula shown in (Equation 6). Each *PS* result denotes how interested a user would be in that post, based on that user's *Genre Scores* that overlap with the post's *genres* and how similar preferences the user and the poster have.

$$PS = \frac{\sum_{i \in GSL_U(PG)} GS_U(i)}{|GSL_U(PG)|} \cdot FSS_{poster} \quad (6)$$

The resulting item score is not capped, as the values of genre and like scores cannot be predicted, and also allows a discreet ranking among items.

E. Experimental Evaluation

Named entity recognition. A critical phase of the recommendation process is entity recognition, which allows `Reveal` to identify the entities that the specific post is related to and define the *genres* that are associated with it. To that end, we measure the effectiveness of our module that maps the unique video IDs to unique topic, and thus entity, IDs. We create a dataset by crawling 20 popular music/movie Facebook Group pages. To obtain accurate results we filter out posts that do not contain a valid Youtube URL; our final dataset contains 5,310 links to Youtube. Our module which is able to identify 4,743 valid title-entities, achieving a coverage of 89.32%, demonstrating the robustness of our approach.

Sentiment analysis threshold. Next, we evaluate our sentiment analysis module, and its effectiveness in providing accurate results regarding the sentiment of a given post's content. We obtained datasets published in previous studies that contained English Tweets and Facebook comments, rendering them a suitable sample of the type of content we expect our system to handle. Specifically, we used the following labelled datasets as *ground truth* for evaluating our approach:

- Twitter Dataset [13] (5,513 tweets)
- Facebook Comments Dataset [14] (1,000 comments)

We pre-processed the datasets to remove text with a truly neutral sentiment, i.e., with a score of 0, as such text will not offer any valuable semantic information about the published content. Then, to regulate the semantic noise i.e., false positives/negatives, we experimented with different sentiment score thresholds, which specified whether a post is classified as positive or negative. The outcome of our algorithm for each text was compared to the polarity label in the aforementioned corpora, to calculate the accuracy. `Reveal` achieved the best results for a threshold of 0.7, by correctly labeling 79% and 76% of the positive and negative samples respectively.

Relevance of content. We collect all the data from 38 test subjects that installed our app. We focus on the posts that belong to one of the following three Graph API Edges: `Links` (any post with an embedded Youtube video), `Watches`, and `Listens`. Table II breaks down the statistics for relevant content identified by our system's heuristics, extracted from 3,493 accounts that were connected to our participants. Interestingly, we identified many cases of `Links` with invalid URLs; this included old Youtube links that were no longer available, or that were malformed (most likely due to users copy-pasting only part of the link to their post). As a result, 59.98% (94,626 of the 157,762) of the link posts were broken and, thus, removed. Also posts under irrelevant post categories (i.e., statuses, photos etc), amounted to more than 50% of the initial dataset and filtered out as well. Of the remaining valid posts, 30.9% contained relevant content under the music and movies categories, indicating that there is an abundance of content being published that could lead to interesting content being overlooked by users or ignored by the current selection algorithm of Facebook's News Feed.

TABLE II. Posts collected from participants and their friends.

Type	Number of posts
Total	**521,685**
- Filtered out (irrelevant, broken links)	277,356
- Processed	244,329
Relevant	**75,694**
- Movies	28,994
- Music	46,700

Reveal vs Facebook. To explore whether users missed relevant content due to Facebook's personalization algorithm, we gathered every News Feed post from N=7 participants (age M=27.6, SD=3.36; 71.4% male), with number of friends M=547.4, SD=154, that gave us access to their data for a time period of 2 weeks (maximum allowed by Graph API). We compared them to the content posted by their friends, and also processed the data with `Reveal` to identify posts of interest. We then compared the two datasets and identified the common items. The users' News Feeds contained a total of 23,546 items, while our system selected 1,680 posts also present in the News Feed, but also revealed 3,127 additional posts that are suitable for recommendation as they match the users' *interest profiles*.

III. Conclusion

We explored the utility of a recommender system within a social network designed to select content published by the user's friends that matches a fine-grained interest profile that is generated from social data. Our prototype app allowed us to explore the practical aspects and intricacies of processing and extracting information from user-published content, and our subsequent evaluation asserted the effectiveness of our approach, and the suitability of social networks as a information-rich ecosystem for providing fine-grained recommendations.

References

[1] (2016) Facebook news feed. https://goo.gl/Dupvg8.
[2] B. Gretarsson, J. O'Donovan, S. Bostandjiev, C. Hall, and T. Holerer, "Smallworlds: Visualizing social recommendations." *Comput. Graph. Forum*, vol. 29, no. 3, 2010.
[3] (2016) Facebook - graph api edges. https://goo.gl/PdWuYm.
[4] S. Mohammad, "From once upon a time to happily ever after: Tracking emotions in novels and fairy tales," in *5th ACL-HLT*, 2011.
[5] F. Å. Nielsen, "A new ANEW: evaluation of a word list for sentiment analysis in microblogs," *CoRR*, vol. abs/1103.2903, 2011.
[6] S. Gouws, D. Metzler, C. Cai, and E. Hovy, "Contextual bearing on linguistic variation in social media," in *LSM workshop*, ser. LSM, 2011.
[7] B. Liu, M. Hu, and J. Cheng, "Opinion observer: Analyzing and comparing opinions on the web," in *14th WWW*, ser. WWW, 2005.
[8] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Comput. Linguist.*, vol. 37, 2011.
[9] T. De Smedt and W. Daelemans, "Pattern for python," *J. Mach. Learn. Res.*, vol. 13, no. 1, 2012.
[10] B. Liu, *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, 2015.
[11] (2016) Facebook - using actions. https://goo.gl/9schoS.
[12] (2016) Freebase. https://developers.google.com/freebase/.
[13] N. Sanders. (2011) Twitter sentiment corpus. https://goo.gl/QwEP68.
[14] K. Zhang, Y. Cheng, Y. Xie, D. Honbo, A. Agrawal, D. Palsetia, K. Lee, W.-k. Liao, and A. Choudhary, "Ses: Sentiment elicitation system for social media data," in *IEEE*, ser. ICDMW, 2011.