Addressing Scalability Issues in Semantics-Driven Recommender Systems

Mounir M. Bendouch
Erasmus University Rotterdam
Rotterdam, the Netherlands
mbendouch@hotmail.com

Flavius Frasincar
Erasmus University Rotterdam
Rotterdam, the Netherlands
frasincar@ese.eur.nl

Tarmo Robal
Tallinn University of Technology
Tallinn, Estonia
tarmo.robal@ttu.ee

ABSTRACT

Content-based semantics-driven recommender systems are often used in the small-scale news recommendation domain. These recommender systems improve over TF-IDF by taking into account (domain) semantics through semantic lexicons or domain ontologies. Our work explores the application of such recommender systems to other domains, using the case of large-scale movie recommendations. We propose new methods to extract semantic features from various item descriptions, and for scaling up the semantics-driven approach with pre-computation of the cosine similarities and gradient learning of the model. The results of the study on a large-scale dataset of user ratings demonstrate that semantics-driven recommenders can be extended to more complex domains and outperform TF-IDF on *ROC*, *PR*, F_1, and Kappa metrics.

CCS CONCEPTS

• **Information systems → Recommender systems**; *Ontologies*; *Personalization.*

KEYWORDS

semantics-driven recommendation, scalability, ontology

ACM Reference Format:
Mounir M. Bendouch, Flavius Frasincar, and Tarmo Robal. 2021. Addressing Scalability Issues in Semantics-Driven Recommender Systems. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI-IAT '21), December 14–17, 2021, ESSENDON, VIC, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3486622.3493963

1 INTRODUCTION

Since the introduction of the World Wide Web vast amounts of information have become available with an accelerating increase [26], emerging in the prodigious scale of 44 trillion gigabytes in 2020 [22]. However, this abundance of choice of content (articles, movies, music, etc.) comes at the price of information overload, and finding the right content has become exceedingly time consuming.

Recommender systems (RS) [17, 18] have emerged as a solution to this problem, filling the need to filter and deliver relevant content to the user by sorting through large amounts of information and presenting the most interesting selection in the form of recommendations. This goes beyond plain information retrieval systems such as search engines because RS incorporate the user's preferences, interests, and needs captured in user profiles, e.g., by using domain models [19]. Two different approaches to RS [17] can be distinguished: *collaborative filtering*, which provides recommendations based on similarities between preferences of one user and preferences of others, and *content-based filtering*, which recommends items according to their content, differing in the data and underlying assumptions they use for their predictions. A combination of the two latter is known as *hybrid* RS [4]. Here, our focus will be on content-based RS operating on similarities between content items.

Content-based RS [15] vary in the features they consider and how they use these to calculate similarities among items. The features that are available to the RS depend on the item type and dataset. For instance, the director, cast, genre, plot, etc. are typically available to movie recommenders. Yet, text (e.g. descriptions, summaries, reviews, etc.) is a common form of information about the contents of various on-line items from which features can be extracted for measuring similarity. A widely used technique to estimate similarity between texts or documents is Term Frequency - Inverse Document Frequency (TF-IDF) [13], which constructs for each document a feature vector out of the frequency counts of terms in the document, and multiplies these frequencies by the inverse frequency that these terms occur in all documents. The resulting vectors can then be directly compared using measures such as cosine similarity [11].

Several RS building upon TF-IDF, like for example CF-IDF(+), SF-IDF(+), have been established, mainly for the recommendation of news articles [3, 5, 8, 11], using concepts from domain ontologies or synsets from semantic lexicons for features instead of terms. These methods have further been extended [6, 14, 23] by adding semantically related synsets or concepts to the vector, or incorporating named-entity similarities using Bing page counts. These RS make use of weight vectors to optimize the relative importance of the different features in the calculation of the similarities. Inspired by the promising results of these semantics-driven news articles RS, we want to explore the application of SF/CF-IDF(+) to large-scale recommendation and choose movies as the domain.

In this paper, we show that the previously mentioned semantics-driven RS used for small-scale news recommendations can be extended to more complex domains, and are able to make high-quality recommendations on an extremely large scale. We specifically focus on one of the domains that is largely different from news articles – movie recommendations – and that could largely benefit from scalable RS as information on thousands of movies and millions of user ratings is available. This leads us to the following research questions:

WI-IAT '21, December 14–17, 2021, ESSENDON, VIC, Australia
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9115-3/21/12…$15.00
https://doi.org/10.1145/3486622.3493963

RQ1: *Whether and how can semantics-driven recommenders be applied to a large-scale (movie) recommendation problem?*
RQ2: *How to scale the existing proven approach to large datasets?*

To answer *RQ1* and *RQ2*, we propose new methods to scale up the semantics-driven approach and its optimization procedure, extract semantic features from publicly available movie-level information, and find related concepts without the need for an external domain ontology. A challenge is the extraction of semantic features from information of different nature – data available for movies is encoded differently and the items are more complex than just collections of texts used for news recommendation. Thus, we extend semantics-driven RS to the problem of large-scale (movie) recommendations, which can pose substantial challenges to traditional recommender systems. The main contributions of this paper are:

- First, a method for extraction of semantic features from complex domain information for semantics-driven RS. We employ the example of movies for the case, where variable data exists in large scale.
- Second, we establish a method to efficiently devise a domain ontology for the selected complex dataset when an external ontology is not available.
- Third, we propose a method to scale up existing semantics-driven recommendation methods (proven for news recommendation) for large-scale variable data with pre-computation of the cosine similarities and gradient learning of the model, delivering an approach of scalable RS for complex domains. We show that semantics-driven RS can be scaled up to millions of user profiles and thousands of features by rewriting the similarity model as a function of the dot-products between the feature vectors of individual items, which can be pre-computed before optimization.

The remainder of this paper is organized as follows. Section 2 discusses related work on content-based RS. Section 3 discusses data used for the research, while Section 4 describes the proposed recommender. Sections 5 and 6 deliver method evaluation and conclusions respectively.

2 RELATED WORK

Let us start with a review of the semantics-driven recommenders. The RS we focus here were originally designed for news recommendation by extracting features from article text but can be used to predict similarity between any two texts. We will consider the following recommenders: TF-IDF, CF-IDF, SF-IDF, and their extensions CF-IDF+, SF-IDF+, and Bing-SF-IDF+.

The TF-IDF is of interest as SF-IDF and CF-IDF build on the mathematical concept provided by it. The Term Frequency - Inverse Document Frequency (TF-IDF) [20] recommender consists of two parts, where the TF indicates how often a term occurs in a given document (higher frequencies linked to higher relevancies), and the IDF captures the importance and uniqueness of a term in a collection of documents. Frequent terms are considered to be common and less important. The resulting feature vector represents terms with scores, which can be compared to user vectors using similarity functions, e.g., the cosine distance. The TF-IDF score is large for terms that occur frequently in single document but not often in all other documents. A certain specified threshold value decides whether an item and the user's interest are considered similar.

The Concept Frequency - Inverse Document Frequency (CF-IDF) [11] is a variant of TF-IDF, which uses concepts instead of terms. The text is processed by a natural language processing (NLP) engine that performs word sense disambiguation, part-of-speech (POS) tagging, and tokenization to transform the text into a collection of concept candidates. A domain ontology containing concepts and their relationships is checked for each candidate, and if a match is found, a count is added to that concept. Using concepts instead of terms represents the domain semantics better as only words relevant to the specific domain are considered, resulting in an observed performance improvement over TF-IDF in [11]. CF-IDF+ extends this method further by including directly related concepts in the domain ontology [8]. Each type of relationship (superclass, subclass, or instance) is given a weight to vary the overall importance of the found concepts and their related concepts. These three weights are then optimized by grid search.

The Synset Frequency - Inverse Document Frequency (SF-IDF) [5] is another variation of TF-IDF, which in addition to all terms looks at synonyms and ambiguous terms using a semantic lexicon (WordNet). Terms having the same meaning will be subsumed in one single concept, and therefore, word sense disambiguation (WSD) is needed. For terms with multiple meanings, corresponding word senses are to be counted separately. This generally results in a longer vector than CF-IDF because more matches are found as WordNet is much larger than a typical domain ontology. SF-IDF+ [14] outperforms SF-IDF by including synsets that are directly related over the 27 types of semantic relationships present in WordNet, where each type has a weight optimized by a genetic algorithm.

Bing-SF-IDF+ [6] extends SF-IDF+, which in addition to words in the semantic lexicon also considers the similarity between named entities frequently occurring on the Web through a separate similarity measure – Bing distance, based on the number of page counts originating from the Bing search engine. This measure is a function of three search result page counts: two counts for each entity separately and one for a combination of the two. An optimized weight is used to combine the Bing distance and the SF-IDF+ cosine similarity, leading to improved performance over SF-IDF+ [6].

Recommenders for (multi)media are of interest for researchers due to the large variable information available, and different approaches have been taken to provide recommendations. Multimedia recommendations over social media streams using a graphical model and signature-tree-based scheme over Youtube and MovieLens data is explored in [28]. The possibility to derive the context from social media for context-aware social media recommendations is studied in [27], where the work, as ours, also advantages from TF-IDF. An ontology based RS for online social network data is proposed in [1] with experiments carried out on MovieLens data for which new ontology is created, unlike in our work, this is done manually. Large-scale ontology is used to provide fiction content recommendations and mitigate the item cold-start problem in [21]. In [16] the applicability of Bidirectional Encoder Representations from Transformers (BERT) [10] in the context of conversational RS is explored, experimenting with movies, books and music recommendation and showing its suitability especially for content-based knowledge. In [25] on the other hand Word2Vec algorithm is used

Table 1: Research data descriptive statistics (movies).

Data type (*ML=MovieLens*)	N	Missing [%]	Mean	Min	Max
Title (ML)	27,278				
Genres (ML)*	27,278		1.99	1	10
Genres (OMDb)*	27,207	0.26	2.21	1	5
Directors (OMDb)*	27,003	1.01	1.11	1	41
Plot (OMDb)**	26,327	3.49	63.49	3	1,471
Writers (OMDb)*	25,831	5.30	2.41	1	35
Actors (OMDb)*	26,925	1.29	3.93	1	4

* Multi-class variable, statistics reported for number of classes.
** Full text, statistics reported for number of words.

to recommend movies based on metadata (e.g., directors, actors) information. A comprehensive overview of recommender systems for multimedia content is provided in [9], and a survey on the usage of knowledge graphs for RS in [12].

3 LARGE-SCALE RECOMMENDATION DATA

To study scalability of semantics-driven RS, we choose large-scale recommendation of movies, using the datasets available through the GroupLens Research Project[1]. We gather user ratings from the largest dataset available (MovieLens 20M[2]) describing 5-star user rating activity from an on-line movie recommendation service MovieLens[3]. It contains a total of 20,000,000 user ratings on a scale of 1 to 5 across 27,278 movies over a ten-year period from the ratings given by 138,493 users, who had all rated at least 20 movies.

Semantics-driven RS are content-based and thereby require item-level information as input for feature extraction. The MovieLens data contains the title, year of release, genre labels, and identification numbers for The Internet Movie Database (IMDB)[4] for each movie. In addition we use the Open Movie Database (OMDb)[5] over IMDB ids to query movie plots, available for 96.51% of the movies in the MovieLens data, containing 63 words on average (substantially shorter than typical news articles). The plots (storyline descriptions without revealing any 'spoilers', similar in their intent to a movie trailer) are the only texts in our data from which semantic features can be extracted. We discard the 3.49% of movies for which no plot is available. Out of the combined data with many movie-level variables we retain only those containing substantial semantic information valuable for semantics-driven recommendations – the names of persons involved in the movie (the actor(s), director(s), and writer(s)), the genre(s), and the plot. Table 1 describes the different variables, their quantity, and characteristics (e.g. mean number of words in plot, % of missing variables, max number of genres, etc.) we use in this research.

Variables such as genres and persons provide additional (domain-specific) information, and do not have to be extracted from text. The relationships between actors, directors, and other persons could be extracted from the available information and used to construct a domain ontology. Each movie is additionally labelled with one or more genres obtained from both MovieLens and OMDb. As these genre labels vary between the two sources both are retained to ensure no valuable information is lost. We discard (affects only 0.83% of collected user ratings) any movie without at least one director, actor, writer, and genre, leaving us with the final dataset of 25,138 movies for this research.

We notice that named entities in the plots are generally fictional characters, which would rarely provide substantial information in Bing distance evaluations, while persons and titles could be used as named entities to calculate Bing distance metrics. Considering that these entities are covered by the domain ontology in this research, and that a large number of pair-wise search queries over these would not be feasible, we decide to exclude Bing distances from our recommender system, and in further focus on semantics from terms, concepts, and synsets, in line with TF-IDF [11], CF-IDF(+) [8, 11], and SF-IDF(+) [6] recommender systems.

4 RECOMMENDATION METHODOLOGY

We start with extraction of semantic features from the plots and how to find related concepts without the need for an external domain ontology, followed by the definition of the similarity model, where we show how a small modification enables massive scalability. The approach builds on the existing TF/CF/SF-IDF(+) recommenders.

4.1 Feature Extraction

Some concepts (e.g., persons, genre labels) are readily obtainable for each movie. In addition, we extract both terms and synsets from the plots using natural language processing (NLP) techniques that can filter out noise from the plots and exploit known regularities in natural language. Using the NLTK[6] package in Python 2.7 for these NLP techniques, each plot is split up into a set of sentences and processed separately. Further, tokenization is applied to split sentences up into a list of words (tokens) using known properties of words, such as they usually occur in the English dictionary, and separated by spaces or commas. Using part-of-speech (POS) tagging, we add to each word the POS (e.g., noun, verb, adjective). Stop words, containing negligible semantic information, are removed. We then apply the Porter [24] stemming algorithm to each word to reduce the words to their roots and extract the terms.

To extract synsets, we apply the Adapted Lesk [2] word sense disambiguation (WSD) algorithm to each word. WSD addresses the problem of identifying the sense of a word – the meaning in its context. Only senses that have the same POS tag as the word from the text are considered. If no sense can be found, all senses with any POS are considered. The synset containing the identified sense of the word is extracted.

4.2 Domain Ontology

Domain ontologies are considered resources that are external to the dataset from which the concepts are derived, and subsequently have to either be obtained or manually constructed for the purpose of the RS. We now propose a general method as an alternative to external domain ontologies, solely based on the dataset itself, allowing to

[1]https://grouplens.org/
[2]http://grouplens.org/datasets/movielens/20m/
[3]https://movielens.org/
[4]http://www.imdb.com/
[5]http://www.omdbapi.com/
[6]http://www.nltk.org/

find concepts related through a common item by a series of matrix multiplications of binary matrices.

Our dataset has 25,138 movies, covering 12,231 directors, 45,393 actors, 27,415 writers, all 19 MovieLens genres (*mlg*), and 27 OMDb genres (*omg*). Based on the average number of these concepts per movie (Table 1) we have an estimated 292,857 bidirectional movie-concept relations that implicitly form a virtual domain ontology, which can be used to find related concepts.

Let the set of existing concept classes $classes = \{director, actor, writer, mlg, omg\}$, its size k and the number of concepts (instances) in class $c \in classes$ as n_c. Let M_c be an $z \times n_c$ binary matrix of the occurrences of concepts of class c in each of the z items, where the (i, j)-th element is denoted as $M_{c(i,j)}$, and occurrences encoded such that $M_{c(i,j)} = 1$ if concept j is present in item i, and $M_{c(i,j)} = 0$ otherwise. Thus, the sum of row i is the number of concepts found in item i, and the sum of column j is the number of items in which concept i is found. All these sums are at least 1 since all concepts occur at least once and every movie has at least one concept of each class. An example of a M_c in our case is $M_{director}$, a binary $25,138 \times 12,231$ matrix that encodes the directors found in each of the movies.

We further denote with $concept_{c(j)}$ the j-th concept of class c and the i-th item with $item_i$. We show that the k matrices M_c can be used to find related concepts for an item through relations of the form (Eq. 1):

$$item_u \xrightarrow{contains} concept_{a(i)} \xrightarrow{occurs\ in} item_v \xrightarrow{contains} concept_{b(j)}$$
$$\forall a, b \in classes \quad \forall u, v \in [1, z] \quad \forall i \in [1, n_a] \quad \forall j \in [1, n_b] \tag{1}$$

Suppose $item_u$ contains a concept of class c. If another $item_v$ contains that same concept, we say that $item_v$ is related to $item_u$ through relation c. The number of possible relations is therefore equal to the number of classes k. For example, if an actor from movie u also plays in movie v, these movies are related through the relation $actor$. Relations are bidirectional and a movie is always related to itself. This existence of a relation c between $item_u$ and $item_v$ is equivalent to the existence of a $j \in [0, n_c]$ for which $M_{c(u,j)} M_{c(v,j)} = 1$. This is the case if and only if (iff) $\sum_{j=1}^{n_c} M_{c(u,j)} M_{c(v,j)} \geqslant 1$. The expression $\sum_{j=1}^{n_c} M_{c(u,j)} M_{c(v,j)}$ is also the definition of the dot-product between the u-th and v-th rows of M_c. If we calculate the $z \times z$ symmetric matrix $R_c = M_c M_c^\top$, then $R_{c_{(u,v)}}$ is the dot-product of the u-th row of M_c and the v-th column of $M_c^\top$. The v-th column of $M_c^\top$ is also the v-th row of M_c. Therefore $R_{c_{(u,v)}} \geqslant 1$ iff $item_u$ and $item_v$ are related through relation c.

Consider that $M_{b(v,j)} = 1$ iff $item_v$ contains $concept_{b(j)}$, and $R_{a_{(u,v)}} \geqslant 1$ iff the $item_v$ is related to $item_u$ through a. Hence $item_u$ has related $concept_{b(j)}$ through relation a with the $item_v$ iff $M_{b(v,j)} R_{a_{(u,v)}} \geqslant 1$, and $item_u$ has related $concept_{b(j)}$ through a with *any* item iff $\sum_{v=1}^{z} R_{a_{(u,v)}} M_{b(v,j)} \geqslant 1$. By the definition of matrix multiplication, it is the (u, j)-th element of the $z \times n_c$ matrix $(R_a M_b)$. We denote this matrix $M_{b \xleftarrow{a}}$ and call it the matrix of related concepts of class b through relation a. It encodes the related concepts of all items since $M_{b \xleftarrow{a} (u,j)} \geqslant 1$ whenever $item_u$ contains related concept j of class b ($concept_{b(j)}$). As we are only interested in occurrences of related concepts (not counts), we make $M_{b \xleftarrow{a}}$ binary by clipping its values to 1, and can directly obtain related concepts from $M_{b \xleftarrow{a}} = R_a M_b = M_a M_a^\top M_b \quad \forall a, b \in classes$. This can be expressed as a function of two arbitrary matrices with same number of rows: $getRelated(A, B) = AA^\top B$. Using this function in a nested way (e.g., $getRelated(getRelated(M_a, M_b), M_c)$, denoted as $M_{c \xleftarrow{ba}}$), we can obtain related concepts through any path length.

This shows that we can find related concepts through relations of the form expressed in Eq. 1 using only matrix multiplications of the k matrices M_c. With this we propose an alternative to external ontologies that can be used to find related concepts using only item-concept occurrences from the dataset itself.

Table 2: Characterization of feature types used.

i	Feature type t_i	Source	Dataset	n_i*	m_i**
1	Directors	Variable	OMDb	12,231	4
2	Actors	Variable	OMDb	45,393	4
3	Writers	Variable	OMDb	27,415	4
4	MovieLens genres	Variable	MovieLens	19	1
5	OMDb genres	Variable	OMDb	27	1
6	Terms	Plot	OMDb	48,083	1
7	Synsets	Plot	OMDb	69,977	19

* # of Features of feature type t_i (i.e., length of feature vector).
** # of Relations between features of type t_i.

4.3 Similarity Model Scaling

The traditional method of scaling of TF-IDF is the Inverse Document Frequency (IDF), and we apply the same scaling to terms and synsets from the plots, in line with SF-IDF(+) [5, 14]. In contrast to CF-IDF+ [8], the scaling of concepts is slightly different as these are not extracted from texts but from variables, and we only extract occurrences of concepts with frequencies in $\{0, 1\}$. We do not apply IDF scaling, as the interpretation of the feature values deviates too much from their original meaning of relative frequencies in texts in TF/SF/CF-IDF+.

After the features are extracted from the descriptions and appropriately scaled, we prepare them for use in the similarity model. For each movie, we have a feature vector for each type of feature (Table 2). We widen the parametric freedom compared to CF-IDF+[8] by placing concepts of each class into separate vectors, so the concepts (features) of each class are considered a different type of feature, allowing to learn their relative importance.

Let us denote the number of feature types as $k = 7$ and the set of feature types as t, where t_i is the i-th type (e.g., $t_1 = director$). We represent the feature values of type t_i for item $g \in [1, z]$ in a matrix $V_i^g \in \mathbb{R}^{m_i \times n_i}$ with z the total number of items, m_i the number of relations, and n_i the number of features. To simplify the notation, our definition of the set of relations r_i includes directly found features as a relation, so the first relation is $r_{i_1} = direct$ for every feature type. Since we retrieve 18 relations from WordNet and only for the plot synsets, it follows that $m_7 = 1 + 18 = 19$ (Table 2). For finding related concepts we limit ourselves to single-step paths of directors, actors, and writers. This means $m_1 = m_2 = m_3 = 1 + 3 = 4$

and for the genres $m_4 = m_5 = 1$. As there are no relations among terms, we have $m_6 = 1$.

The feature matrices V_i^g are constructed such that the rows contain feature values from the same relation, and the columns contain feature values of the same feature. So for the g-th item the scalar element $V^g_{i_{(a,b)}}$ is the value of the b-th feature of type t_i found through relation r_{i_a}. We define the block matrices $F_i = [(V_i^1)^\top\ (V_i^2)^\top ..\ (V_i^z)^\top]^\top\ \forall i \in [1,k]$ and calculate the matrix multiplications $X_i = F_i F_i^\top$. It follows that the block $X_{i_{(a,b)}}$ is equal to the matrix $V_i^a (V_i^b)^\top$.

In order to scale the semantics-driven approach to large datasets, we have to be able to optimize the recommender's parameters on a large number of user profiles. For each of the 138,493 users in our dataset, one or multiple combinations of liked items can be used to construct a profile, e.g., if we construct profiles out of a fixed number of p liked items, the information from a user who has n items in his training set can be used to construct $\binom{n}{p}$ user profiles; thus the number of observations available to optimize the parameters can be much larger than the number of users in the dataset. To be able to scale up semantics-driven recommendations to millions of user profiles and thousands of features, we rewrite the similarity model as a function of the dot-products between the feature vectors of individual items, which can be pre-computed before optimization.

The semantics-driven RS construct one vector of features (for CF-IDF+ frequency of concepts in the text and concepts related to them in domain ontology, and for SF-IDF+ frequency of synsets and relations from lexical ontology) from the user profile and one from the unseen item under consideration, and use cosine similarity to calculate the similarity between those two vectors. The general similarity (sim) that can describe any semantics-driven RS addresses similarity between the user profile and a certain unseen item as a weighted average of a set of k similarity measures, which we call part-similarities.

We denote the weights as a parameter vector $\vec{w}$ and the part-similarities as elements of vector $\vec{s}$, both of length k (the number of part-similarities), and restrict all $w_i \geqslant 0$ so sim cannot be negatively related to any s_i. We assume $0 \leqslant s_i \leqslant 1$, and can therefore achieve the desired $0 \leqslant sim \leqslant 1$ by enforcing $\sum_{i=1}^{k} w_i = 1$. Thus, for CF-IDF+ and SF-IDF+, we can simply define $k = 1$, $\vec{w} = 1$, and $\vec{s} = s_1$ with s_1 a cosine similarity. Further, Bing-SF-IDF+ can also be expressed alike with $k = 2$, $\vec{w} = [\alpha, 1-\alpha]$, s_1 a Bing-similarity and s_2 a cosine similarity. Since we do not use Bing-similarity for our model, any part-similarity s_i in our model is a cosine similarity, which we write as $cos(\vec{u}_i, \vec{v}_i)$, between some feature vector $\vec{u}_i$ of length n_i representing the user profile and another vector of $\vec{v}_i$ of length n_i representing the unseen item, where $\vec{u}_i, \vec{v}_i \geqslant 1$ to achieve $0 \leqslant s_i = cos(\vec{u}_i, \vec{v}_i) \leqslant 1$. This defines sim as (Eq.2):

$$sim = \sum_{i=1}^{k} w_i \cdot s_i = \vec{w} \cdot \vec{s} = \sum_{i=1}^{k} w_i \cdot cos(\vec{u}_i, \vec{v}_i) \tag{2}$$

A feature vector $\vec{v}_i \in \mathbb{R}^{n_i}$ for an unseen item can be seen as a function $f_i()$ of the feature matrix $V_i \in \mathbb{R}^{m_i \times n_i}$ of the item's related feature values and a trainable relation weights vector $\vec{q}_i \in \mathbb{R}^{m_i}$ that is shared across all items. Taking SF-IDF+ as an example (CF-IDF+ is analogous), there exists $k = 1$ part-similarity $s_1 = cos(\vec{u}_1, \vec{v}_1)$ (Eq. 2) and therefore one type t_1 of feature – synsets. The feature values in $\vec{v}_1$ are the SF-IDF+ values of the unseen item, so v_{1_j} is the SF-IDF+ value of the j-th synset and n_1 is the number of unique synsets. As we consider directly found features as a specific case of relation, we can define the first relation as *direct* and in the SF-IDF+ model we restrict $q_{1_1} = 1$. This makes the number of relations $m_1 = 28$ because the remaining relations are the 27 semantic relations found in WordNet. The weights q_{1_l} for $l \in [2, 28]$ are then the 27 optimizable weights restricted to $[0, 1]$. We set our restrictions less strict than those of the original SF-IDF+, to $q_{i_l} \geqslant 0$ and $\sum_{l=1}^{m_i} q_{i_l} = 1$, as we want to allow any weight to be the highest.

We define V_1 and f_1 by noting that the SF-IDF+ value v_{1_j} is the maximum of the direct SF-IDF value and the SF-IDF values of synsets related to synset j multiplied by their corresponding relation weight from q_1. In other words, $V_{1_{(l,j)}}$ is the maximum of SF-IDF values of all synsets related to synset j by the l-th relation. Therefore the j-th column of V_1, denoted $V_{1_{(*,j)}}$ consists of these 28 SF-IDF values from related synsets, one for each type of relation. Note that the first value in the j-th column $V_{1_{(1,j)}}$ is the SF-IDF value of synset j itself. Now to replicate SF-IDF+ within our framework we also need to take a maximum over the 28 related SF-IDF values after they have been multiplied by their corresponding relation weights from $\vec{q}_1$. We therefore define f_1 as follows:

$$SF\text{-}IDF{+}_j = v_{1_j} = f_1(\vec{q}_1, V_{1_{(*,j)}}) = \max_{1 \leq l \leq 28} q_{1_l} V_{1_{(l,j)}} \quad \forall j \in [1, n_1] \tag{3}$$

If we would follow this CF-IDF+/SF-IDF+ method for all features in our research we could define f_i more generally for any feature type t_i as (Eq. 4):

$$v_{i_j} = f_i(\vec{q}_i, V_{i_{(*,j)}}) = \max_{1 \leq l \leq m_i} q_{i_l} V_{i_{(l,j)}} \quad \forall j \in [1, n_i] \tag{4}$$

Eq. 4 reveals the computational bottleneck in the SF/CF-IDF+ models: the item vector $\vec{v}_i$ used in the cosine similarity consists of maxima that can be known only by using the full matrix $V_i \in \mathbb{R}^{m_i \times n_i}$ and the parameter vector of weights $\vec{q}_i \in \mathbb{R}^{m_i}$. As we want to separate the dot-products from the parameters, we change f_i in Eq. 4 to calculate v_{i_j} by taking the sum instead of the maximum:

$$v_{i_j} = f_i(\vec{q}_i, V_{i_{(*,j)}}) = \sum_{l=1}^{m_i} q_{i_l} V_{i_{(l,j)}} = \vec{q}_i \cdot V_{i_{(*,j)}}^\top \quad \forall j \in [1, n_i] \tag{5}$$

We now see (Eq. 5) that the j-th element of v_i has become the matrix multiplication of the row-vector $\vec{q}_i$ and the transpose of column j of V_i. This means that if we represent the vector $\vec{q}_i$ as a $1 \times m_i$ matrix, we no longer need f_i, as we can perform the j multiplications that form the elements of $\vec{v}_i$ with one matrix multiplication, using $\vec{q}_i$ and the full $m_i \times n_i$ matrix V_i (Eq. 6):

$$v_i = \vec{q}_i V_i \tag{6}$$

Notice that the matrix V_i consists of (related) feature values for an unseen item, but can be calculated for any item, including those in the user profiles. We denote V_i^g as the item feature matrix V_i corresponding to the g-th item in a user profile, which is a set of p liked items. As the user profile's features are defined as sums of the p items' features and the weights $\vec{q}_i$ are the same for each item, the user profile feature vector $\vec{u}_i$ can be represented as (Eq. 7):

$$\vec{u}_i = \sum_{g=1}^{p} \vec{v}_i^g = \sum_{g=1}^{p} \vec{q}_i V_i^g = \vec{q}_i \sum_{g=1}^{p} V_i^g = \vec{q}_i U_i \tag{7}$$

Now, we can rewrite the *sim* of our model (Eq. 2) for feature type i as Eq. 8, considering that the dot-product $\vec{a}{\cdot}\vec{b}$ is equivalent to the matrix multiplication $\vec{a}\vec{b}^\top$ and the Euclidean distance $||\vec{a}||_2$ is equivalent to $\sqrt{\vec{a}{\cdot}\vec{a}}$:

$$sim = \sum_{i=1}^{k} w_i s_i = \sum_{i=1}^{k} w_i cos(\vec{q}_i U_i, \vec{q}_i V_i) = \sum_{i=1}^{k} w_i \frac{(\vec{q}_i U_i)\cdot(\vec{q}_i V_i)}{||\vec{q}_i U_i||_2 ||\vec{q}_i V_i||_2} =$$
$$= \sum_{i=1}^{k} w_i \frac{\vec{q}_i U_i (\vec{q}_i V_i)^\top}{\sqrt{\vec{q}_i U_i (\vec{q}_i U_i)^\top}\sqrt{\vec{q}_i V_i (\vec{q}_i V_i)^\top}} = \sum_{i=1}^{k} w_i \frac{\vec{q}_i (U_i V_i^\top) \vec{q}_i^{\,\top}}{\sqrt{\vec{q}_i (U_i U_i^\top) \vec{q}_i^{\,\top}}\sqrt{\vec{q}_i (V_i V_i^\top) \vec{q}_i^{\,\top}}} \tag{8}$$

Since we have defined $U_i = \sum_{g=1}^{p} V_i^g$, we can show that $(U_i V_i^\top)$ and $(U_i U_i^\top)$ can both be written as sums of matrix multiplications of the p feature matrices V_i^g of liked items and the feature matrix V_i of the unseen item:

$$U_i V_i^\top = (\sum_{g=1}^{p} V_i^g) V_i^\top = \sum_{g=1}^{p} (V_i^g V_i^\top) \tag{9}$$

$$\begin{aligned} U_i U_i^\top &= (\sum_{g=1}^{p} V_i^g)(\sum_{g=1}^{p} (V_i^g))^\top = (\sum_{a=1}^{p} V_i^a)(\sum_{b=1}^{p} (V_i^b)^\top) = \\ &= \sum_{a=1}^{p} (V_i^a \sum_{b=1}^{p} (V_i^b)^\top) = \sum_{a=1}^{p}\sum_{b=1}^{p} V_i^a (V_i^b)^\top \end{aligned} \tag{10}$$

Eq. 8 shows how *sim* is function of $(U_i V_i^\top), (U_i U_i^\top), (V_i V_i^\top) \in \mathbb{R}^{m_i \times m_i}$ and the parameter vectors $\vec{q}_i \in \mathbb{R}^{m_i}$, $\vec{w} \in \mathbb{R}^k$. We know (Eq. 9,10) that $U_i U_i^\top$ and $U_i V_i^\top$ are simply sums of $V_i^a (V_i^b)^\top$ for some $a, b \in [1, z]$ and as described, we can retrieve them by accessing $X_{i_{(a,b)}}$. This means we no longer need the large-dimensional V_i as input, and none of the n_i-dimensional dot-products $(\vec{u}_i \cdot \vec{v}_i)$ from the original model have to be computed while sampling observations or while training the weights. The data matrix $X_{i_{(a,b)}}$ has to be calculated only once for each feature type t_i $\forall i \in [1, k]$, in the preparation stage.

Having generalized the similarity model and increased its scalability by reducing dimensionality from n_i to m_i, we see that the restrictions $w_i \geqslant 0$ and $\sum_{i=1}^{k} w_i = 1$ to achieve $0 \leqslant sim \leqslant 1$ of previous recommenders lead to undesirable properties. Namely, user likes/dislikes can only be with some loss function over the deviation of *sim*, where the restriction makes independent mean-shifts impossible, neither can s_i be freely scaled. We avoid these by adjusting the specification and removing $\sum_{i=1}^{k} w_i = 1$, and adding a learnable bias β to the model. Now *sim* is still linear and increasing but less restricted. To then bound the model to $[0, 1]$ we apply a logistic function to the output of the model in Eq. 8:

$$sim = \frac{1}{1 + e^{-\beta - \sum_{i=1}^{k} w_i s_i}} \tag{11}$$

We now relax the previously imposed $s_i \geqslant 0$ restriction because $0 \leqslant sim \leqslant 1, \forall s_i \in \mathbb{R}$. As we no longer need $s_i \geqslant 0$ we can relax $\vec{u}_i, \vec{v}_i \geqslant 0$ for the feature vectors. We still keep $w_i \geqslant 0$ because we still desire $\frac{\partial sim}{\partial s_i} \geqslant 0$. Note that *sim* remains an increasing function of w_i and s_i after the transformation of Eq. 11.

In line with classic logistic regression, we can use cross-entropy, also called logloss, as a loss function over an item's observed like/dislike $y \in \{0, 1\}$ and the predicted similarity $sim \in [0, 1]$ between the item and the user-profile:

$$L = y \log(sim) + (1 - y) \log(1 - sim) \tag{12}$$

The similarity can therefore also be interpreted as the probability of a like given the input data $(U_i V_i^\top), (U_i U_i^\top), (V_i V_i^\top)$ $\forall i \in [1, k]$.

5 EXPERIMENTS AND RESULTS

We train the similarity model on pairs of user-profiles and corresponding unseen items, and use the trained similarity model to recommend items for which the predicted similarity is above a certain threshold value. Evaluation consists of calculating various classification metrics between the test users' actual likes/dislikes versus the recommendations. To optimize (train) the part-similarity weights $\vec{w}$ and the relationship weights $\vec{q}_i$ we apply stochastic gradient descent (SGD) on the gradient of the similarity model. The target similarity $y \in \{0, 1\}$ is defined as $y = I\{\ user\ likes\ item\ \}$. An item is considered liked by a user if the user rates it with a score ≥ 4.5 and disliked otherwise. This results in an average proportion of 19.12% liked items, and 20.9 liked items per user. We shuffle the order of users in our dataset and take the first 1,000 as the test (hold-out) set used for evaluation, the following 1,000 as the validation set to validate the similarity model for early stopping while training, and the remaining 136,493 as the training set to optimize the similarity model.

User-profiles are constructed by sampling $p = 5$ liked items from a user. For each observation (a pair of user-profile and unseen item) the feature matrices $U_i V_i^\top$, $U_i U_i^\top$, and $V_i V_i^\top$ are constructed from the X_i pre-computed data. The $V_i V_i^\top$ are retrieved as blocks of X_i, while $U_i V_i^\top$ and $U_i U_i^\top$ are constructed from sums of p blocks.

For the train and validation sets, the unseen items are defined as all items not in the user-profile, which is sampled from a random user. For each user-profile, we sample a liked item or a disliked item with equal probability such that we obtain balanced train and validation sets with $E(y) = 0.5$. We sample 100 batches of 1,024 validation observations and 1,374 training batches of 1,024 observations, for totals of 102,400 and 1,406,976 respectively. As the test set should reflect a realistic recommendation setting, we sample the $p = 5$ user-profile items by shuffling all rated items and then iteratively discarding the first item, adding it to the user-profile if it is liked. We stop as soon as we have obtained $p = 5$ liked items. All discarded liked and disliked items are then considered to be seen – thus, we simulate the situation in time when a recommender detects that a user has liked $p = 5$ items. We require the unseen items to contain at least one liked and one disliked item to able to measure performance, which leaves us with 809 eligible user-profiles from the 1,000 test users. We then construct observations for the user-profile with each unseen item. For each user, we save these in a separate batch. The test data is therefore composed of 809 batches of varying sizes, namely the number of unseen items.

The similarity model is trained with SGD on the 1,374 training batches. After each epoch the validation error is measured on the

Table 3: Models (model C holds 5 concept feature types – Directors, Actors, Writers, and genres from MovieLens and OMDb) and their optimization results, averages over 10 random restarts. n=102,400 validation and n=1,406,976 train observations.

Model	k^*	θ^{**}	Logloss***		Training time****		
			Valid.	Train	Epochs	Secs/Epoch	Minutes
C+S (C and S combined)	6	38	0.6812	0.6822	11.0	22.7	4.2
C (Modified CF-IDF+)	5	18	0.6815	0.6826	11.9	10.3	2.0
T (TF-IDF, benchmark)	1	2	0.6896	0.6900	10.0	6.4	1.1
S (SF-IDF+, synsets from plots)	1	21	0.6912	0.6914	11.0	14.7	2.7

* Number of feature types (part-similarities) ** Number of parameters.
*** Minimum over all epochs. **** Until early stopping.

100 validation batches. If it has improved compared to the last epoch, the current parameters are saved. Early stopping is activated as soon as the validation error has not improved over the last 5 epochs. The order of the training batches is shuffled at every epoch. We anneal the learning rate at every epoch by dividing the initial learning rate by the number of elapsed epochs. The stored parameters, those that led to the lowest validation error while training, are subsequently used for evaluation. The pseudo-code for this procedure is given in Algorithm 1.

The optimization procedure is implemented in Python 2.7 using Keras[7] and Theano[8] libraries. The gradients are calculated automatically by Theano using backpropagation. The calculations are run on a regular PC equipped with a NVIDIA GTX1060 GPU, enabling efficient parallel computations of the gradient updates in batches of 1,024 observations. The results for computational load of the optimization procedure (Table 3) show that given the separately pre-trained visual scaling, we can optimize the model with the scalable approach using pre-computed dot-products in 4-5 minutes. Training time (Table 3) is an important factor in the practical scalability of our approach.

We evaluate our method on the 809 test user-profiles sampled using the trained model to predict the similarity score for each unseen item in a batch. For each threshold $\tau \in \{\frac{i}{500}\ \forall i \in (0, 500)\}$ the unseen items for which $sim > \tau$ are recommended. We use TF-IDF (based on the extracted terms) as a benchmark and show the value of the proposed semantics-driven recommendation through comparison to other models. Our version of SF-IDF+ is based on synsets from plots, whereas the modified CF-IDF+ uses the features directly captured from variables (feature types 1 to 5 in Table 2). Table 3 describes the models used.

The comparison between the predicted scores and the actual likes forms the basis of performance measurement, expressed through area under curve (AUC) for the precision-recall (PR) and receiver operating characteristic (ROC) curve, F_1-measure, and Cohen's kappa [7] coefficient κ. Performance metrics are calculated for all 809 test users (Table 4). Even though we do not directly optimize for these metrics, a lower logloss results in higher test performance.

We obtain an unexpected weak result for SF-IDF+ (S) as it does not outperform TF-IDF on any metric, although the difference is not statistically significant. The WSD, additional information in the form of synsets, combined with more parametric freedom apparently do not translate into higher performance. The optimization procedure could be a factor, but the loss function works to increase test performance. Regardless, SF-IDF+ also shows a higher train and validation logloss (Table 3). Our further investigation points to low-quality WSD as the most likely cause of this under-performance. Although theoretically disambiguation leads to more accurate semantics, mistakes are also amplified.

Table 4 presents the analysis of performance metrics (averages are over 10 random runs) over all existing RS models on our solution for addressing scalability, and shows that concepts alone (C) are more informative than both synsets and terms as model C substantially improves over the baseline TF-IDF on all metrics. C+S outperforms the benchmark TF-IDF on all metrics – average

Algorithm 1 Optimization

```
1:  procedure OPTIMIZE($\alpha$, $trainBatches$, $valBatches$)
2:      Generate $\beta \sim$ Normal                          ▷ Random unrestricted
3:      Generate $w_i, \vec{q}_i \sim$ Exp $\forall i \in [1, k]$   ▷ Random non-negative
4:      $\vec{q}_i \leftarrow \vec{q}_i / \sum \vec{q}_i \ \ \forall i \in [1, k]$      ▷ Sum of $\vec{q}_i$ restriction
5:      $\theta \leftarrow \{\beta, \vec{w}, \vec{q}_1..\vec{q}_k\}$                  ▷ Parameters
6:      $epoch \leftarrow 1$                                   ▷ Epoch counter
7:      $stop \leftarrow 1$                                    ▷ Stopping counter
8:      while $stop \leqslant 5$ do                            ▷ Early stopping criterion
9:          Shuffle $trainBatches$
10:         $\alpha_t = \alpha / epoch$                        ▷ Learning rate annealing
11:         for $\mathcal{B} \in trainBatches$ do              ▷ Loop over batches
12:             $\theta \leftarrow \theta - \frac{\alpha_t}{1,024} \sum_{obs=1}^{1,024} \nabla L(\theta, \mathcal{B}_{obs})$   ▷ Gradient descent update
13:         end for
14:         $\ell \leftarrow L(\theta, valBatches)$            ▷ Get average validation loss
15:         if $epoch = 1$ or $\ell < \ell^*$ then
16:             $\ell^* \leftarrow \ell$                       ▷ Update best validation loss
17:             $\theta^* \leftarrow \theta$                   ▷ Update best parameters
18:             $stop \leftarrow 1$                            ▷ Reset stop counter
19:         else
20:             $stop \leftarrow stop + 1$                     ▷ Increment stop counter
21:         end if
22:         $epoch \leftarrow epoch + 1$                       ▷ Increment epoch counter
23:     end while
24:     return $\theta^*$                                      ▷ Return best parameters
```

[7]https://keras.io/
[8]https://pypi.org/project/Theano/

Table 4: Performance on test set, $n = 809$ users, averages over 10 random restarts. TF-IDF is the benchmark.

Models	*AUC*		F_1		κ	
	ROC	PR	$\min_r$	$\max_r$	$\min_r$	$\max_r$
C+S	0.570	0.361	0.419	0.509	0.083	0.251
C	0.567	0.358	0.419	0.507	0.081	0.249
T (TF-IDF)	0.535	0.324	0.413	0.479	0.041	0.200
S (SF-IDF+)	0.531	0.319	0.411	0.477	0.038	0.198

AUC(ROC) improves from 0.535 to 0.570, and AUC(PR) from 0.324 to 0.361, $\min_r(F_1)$ from 0.413 to 0.419, and $\max_r(F_1)$ from 0.479 to 0.509, Kappa metrics from 0.041 to 0.083 and from 0.200 to 0.251 for $\min_r(\kappa)$ and $\max_r(\kappa)$ respectively. To conclude, we see it is neither necessary to train the scaling together with the model as a whole, nor to directly optimize on the final performance metrics.

6 CONCLUSION

In this paper we proposed an extension to previous works on semantics-driven recommenders. We demonstrated that these systems are broadly applicable beyond news recommendations, for instance the complex domain of movies.

We found that rich semantic information can be extracted not just from articles but item descriptions in a much broader sense, and when a suitable domain ontology is unavailable our virtual ontology method can be applied directly requiring only the dataset itself. In situations where the proper scaling method for feature vectors is unknown, we showed that effective scales can be found through direct optimization of the logloss. Through a reformulation of how related features are combined, we were able to pre-compute the computationally expensive operations of the cosine similarities and reduced the dimensionality of the similarity model by several orders of magnitude.

The semantics-driven recommender we presented strongly outperformed the benchmark TF-IDF on *ROC*, *PR*, F_1, and κ, even though it was not directly optimized on these metrics but on a cross-entropy loss function that allowed for efficient gradient-based optimization. The proposed scaling-up of the semantics-driven approach has allowed us to optimize these models within minutes on consumer-grade commodity hardware. This research highlights that semantics-driven recommenders have many unexplored applications and can be utilized effectively with the proposed approach, opening the door to further extensions to other domains.

REFERENCES

[1] Mohamad Arafeh, Paolo Ceravolo, Azzam Mourad, Ernesto Damiani, and Emanuele Bellini. 2021. Ontology based recommender system using social network data. *Future Generation Computer Systems* 115 (2021), 769–779.

[2] Satanjeev Banerjee and Ted Pedersen. 2002. An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. In *Computational Linguistics and Intelligent Text Processing*, A. Gelbukh (Ed.). Springer, Berlin, Heidelberg, 136–145.

[3] Emma Brocken, Aron Hartveld, Emma de Koning, Thomas van Noort, Frederik Hogenboom, Flavius Frasincar, and Tarmo Robal. 2019. Bing-CF-IDF+: A Semantics-Driven News Recommender System. In *Advanced Information Systems Engineering*, P. Giorgini and B. Weber (Eds.). Springer, Cham, 32–47.

[4] Robin Burke. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction* 12, 4 (2002), 331–370.

[5] Michel Capelle, Flavius Frasincar, Marnix Moerland, and Frederik Hogenboom. 2012. Semantics-based News Recommendation. In *Proc. of the 2nd International Conference on Web Intelligence, Mining and Semantics* (Craiova, Romania) *(WIMS 2012)*. ACM, New York, NY, USA, Article 27, 9 pages.

[6] Michel Capelle, Marnix Moerland, Frederik Hogenboom, Flavius Frasincar, and Damir Vandic. 2015. Bing-SF-IDF+: A Hybrid Semantics-Driven News Recommender. In *Proc. of the 2015 ACM Symposium on Applied Computing* (Salamanca, Spain) *(SAC 2015)*. ACM, New York, NY, USA, 732–739.

[7] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46.

[8] Emma de Koning, Frederik Hogenboom, and Flavius Frasincar. 2018. News Recommendation with CF-IDF+. In *30th Intl Conference on Advanced Information Systems Engineering (CAiSE 2018) (LNCS, Vol. 10816)*. Springer, Cham, 170–184.

[9] Yashar Deldjoo, Markus Schedl, Paolo Cremonesi, and Gabriella Pasi. 2020. Recommender Systems Leveraging Multimedia Content. *ACM Comput. Surv.* 53, 5, Article 106 (Sept. 2020), 38 pages.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, MN, 4171–4186.

[11] Frank Goossen, Wouter IJntema, Flavius Frasincar, Frederik Hogenboom, and Uzay Kaymak. 2011. News Personalization Using the CF-IDF Semantic Recommender. In *Proceedings of the 1st International Conference on Web Intelligence, Mining and Semantics* (Sogndal, Norway) *(WIMS 2011)*. ACM, New York, NY, USA, Article 10, 12 pages.

[12] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A Survey on Knowledge Graph-Based Recommender Systems. *IEEE Transactions on Knowledge and Data Engineering* 01 (2020), 1–1.

[13] Karen Sparck Jones. 1972. A Statistical Interpretation of Term Specificity and its Application in Retrieval. *Journal of Documentation* 28, 1 (1972), 11–21.

[14] Marnix Moerland, Frederik Hogenboom, Michel Capelle, and Flavius Frasincar. 2013. Semantics-based News Recommendation with SF-IDF+. In *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics* (Madrid, Spain) *(WIMS 2013)*. ACM, New York, NY, USA, Article 22, 8 pages.

[15] Michael J. Pazzani and Daniel Billsus. 2007. *Content-Based Recommendation Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 325–341.

[16] Gustavo Penha and Claudia Hauff. 2020. What Does BERT Know about Books, Movies and Music? Probing BERT for Conversational Recommendation. In *14th ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 388–397.

[17] Amir Hossein Nabizadeh Rafsanjani, Naomie Salim, Atae Rezaei Aghdam, and Karamollah Bagheri Fard. 2013. Recommendation Systems: A Review. *International Journal of Computational Engineering Research* 3, 5 (2013), 47–52.

[18] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. *Recommender Systems Handbook*. Springer, Boston, MA, USA.

[19] Tarmo Robal, Hele-Mai Haav, and Ahto Kalja. 2007. Making Web Users' Domain Models Explicit by Applying Ontologies. In *Advances in Conceptual Modeling - Foundations and Applications, ER 2007 Workshops CMLSA, FP-UML, ONISW, QoIS, RIGiM, SeCoGIS (LNCS, Vol. 4802)*. Springer, Berlin, 170–179.

[20] Gerard Salton and Christopher Buckley. 1988. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management* 24, 5 (1988), 513–523.

[21] Paul Sheridan, Mikael Onsjö, Claudia Becerra, Sergio Jimenez, and George Dueñas. 2019. An Ontology-Based Recommender System with an Application to the Star Trek Television Franchise. *Future Internet* 11, 9 (2019), 182.

[22] Vernon Turner, John F. Gantz, David Reinsel, and Stephen Minton. 2014. The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. International Data Corporation, White Paper, IDC_1672.

[23] Lies Hooft van Huijsduijnen, Thom Hoogmoed, Geertje Keulers, Edmar Langendoen, Sanne Langendoen, Tim Vos, Frederik Hogenboom, Flavius Frasincar, and Tarmo Robal. 2020. Bing-CSF-IDF+: A Semantics-Driven Recommender System for News. In *New Trends in Databases and Information Systems*, J. Darmont, B. Novikov, and R. Wrembel (Eds.). Springer, Cham, 143–153.

[24] C.J. Van Rijsbergen, S.E. Robertson, and M.F. Porter. 1980. *New Models in Probabilistic Information Retrieval*. Computer Laboratory, University of Cambridge, Cambridge, England.

[25] Yeo Chan Yoon and Jun Woo Lee. 2018. Movie Recommendation Using Metadata Based Word2Vec Algorithm. In *2018 International Conference on Platform Technology and Service (PlatCon)*. IEEE, Jeju, Korea (South), 1–6.

[26] Guo-Qing Zhang, Guo-Qiang Zhang, Qing-Feng Yang, Su-Qi Cheng, and Tao Zhou. 2008. Evolution of the Internet and its Cores. *New Journal of Physics* 10, 12 (2008), 123027.

[27] Xiangmin Zhou, Dong Qin, Lei Chen, and Yanchun Zhang. 2019. Real-Time Context-Aware Social Media Recommendation. *The VLDB Journal* 28, 2 (April 2019), 197–219. https://doi.org/10.1007/s00778-018-0524-7

[28] Xiangmin Zhou, Dong Qin, Xiaolu Lu, Lei Chen, and Yanchun Zhang. 2019. Online Social Media Recommendation Over Streams. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, Piscataway, New Jersey, 938–949.