# Improving Relevance in a Content Pipeline via Syntactic Generalization

Boris Galitsky

Knowledge-Trail, Inc. Jose CA USA, bgalitsky@hotmail.com

**Abstract.** This is a report from the field on a linguistic-based relevance technology based on learning of parse trees for processing, classification and delivery of a stream of texts. We describe the content pipeline for eBay entertainment domain which employs this technology, and show that text processing relevance is the main bottleneck for its performance. A number of components of the content pipeline such as content mining, aggregation, deduplication, opinion mining, integrity enforcing need to rely on domain-independent efficient text classification, entity extraction and relevance assessment operations.

Text relevance assessment is based on the operation of syntactic generalization (SG) which finds a maximum common sub-tree for a pair of parse trees for sentences. Relevance of two portions of texts is then defined as a cardinality of this sub-tree. SG is intended to substitute keyword-based analysis for more accurate assessment of relevance which takes phrase-level and sentence-level information into account. In the partial case where short expression are commonly used terms such as Facebook likes, SG ascends to the level of categories and a reasoning technique is required to map these categories in the course of relevance assessment.

A number of content pipeline components employ web mining which needs SG to compare web search results. We describe how SG works in a number of components in the content pipeline including personalization and recommendation, and provide the evaluation results for eBay deployment. Content pipeline support is implemented as an open source contribution OpenNLP.Similarity and is available at https://code.google.com/p/relevance-based-on-parse-trees/.

**Keywords:** content pipeline, relevance of text classification, machine learning of syntactic parse trees, personalized recommendation

## 1. Introduction

Building a relevant and efficient content pipeline is a key to successful consume Internet application. In recent years, a number of scalable, adjustable, and intelligent platforms have been built for content processing, providing an effective information access for users. These platforms are capable of collecting relevant information, filtering out low quality unreliable pieces of content, de-duplication, aggregating it from multiple sources, achieving integrity between various data components, and generating search, recommendation and personalization results. Moreover, these platforms support search engine optimization, mobile application, share content with partner network, and other functionality. In this report from the field, we share the experience working in the domain of eBay.com entertainment, and providing entertainment content for indexing to Bing.com.

The main limitation of the current content pipeline is a quality of content obtained from various sources on the web, and its relevance to user intent. In the domains such as entertainment, a significant portion of content cannot be structured or formalized, so needs to be handled as text. If the relevance recall is low, users' access to products and services is limited; they don't get search, recommendation and personalization results they should be getting. If, on the contrary, precision is low, users get lower quality results than potentially available. Also, the users would lose trust in a web portal such as eBay.com [28]. In this study we demonstrate how a use of linguistic technology such as machine learning syntactic parse trees for relevance assessment can benefit content pipeline. To systematically learn linguistic structures, we introduce a *syntactic generalization* (SG) operation which finds a common maximum sub-trees for parse trees of sentences. For paragraphs of text, we extend this operation to such structures as *parse thicket*, expressing information on how sentences are connected in a paragraph, in addition to parse trees of this paragraph.

The input of the content pipeline presented here is a stream of international entertainment events, ranging from concerts to shows, from sports to industry meet-ups, which are coming at a speed of up to 6000 per day. An event data, obtained from various sources, frequently inconsistent, need to come through the content pipeline as fast as possible and appear in the index for search and recommendation to become available to users. Ticket availability stream at the scale of eBay is massive data as well, but will not be considered in this study. Event data from different sources varies a lot in structure and the kind of information available, and mostly occur in the form of unstructured text. Opinion data on events, performers and venues are even more diverse in terms of structure and linguistic phrasing. Also, data distribution between the sources, and data nature varies with seasons, travels of performers and unexpected circumstances such as political situations in various countries. The content pipeline should be able to adapt to abrupt variations in volume and linguistic properties of incoming content stream [26]. Solving classification problems is an important task of the content pipeline, and these problems need to be solved in the dynamically changing environments [10,37], especially adapting to particular user interests. Required text mining technology needs to be expressive, sensitive and domain-independent at the same time. Using keyword-level classifier for our domains would lead to a huge classification dimension and possibly over-fitting, so we need more accurate representation of linguistic data such as parse trees and machine learning means for them [28].

Typically, relevance in content pipelines is achieved via statistics of keywords. As fast and efficient processing becomes more available in industry, parsing and learning of its results becomes plausible. Platforms like Hadoop and their implementations such as Cascading (Cascading 2013) and Mahout (Mahout 2013) are capable of parsing and learning a large amount of textual data, but the relevance and semantic features are behind. So in this study we focus on making relevance efficient and will not go into low-level implementation details (see them in [28]). We will evaluate how an implementation of machine learning of parse trees can improve a number of text-based content pipeline tasks. Also, an open source implementation for relevance has been built, so it can be integrated and tested in other arbitrary content pipelines.

In contrast to content pipelines performing in restricted domains such as customer relationship management [24] or a vertical product domain, relevance cannot be based on domain knowledge in such a broad domain as eBay entertainment. It is not plausible to build ontology for all kinds of entertainments, maintain and update it to achieve relevance. Instead, we mostly rely on domain-independent linguistic information. We show that once domain-independent efficient text matching component is developed, taking advantage of the rich linguistic information available for learning of parse tree, *the same component* is used to solve a spectrum of problems. Although a number of distributed systems including the open-source ones have been built to address the scalability problem, the relevance for content processing and delivery is still a major bottleneck for effective content-based systems.

## 1.1 Syntactic Generalization for text relevance assessment

*Tree Kernel methods* (TK) for text learning, based on non-parametric density estimation techniques that compute a kernel function between data instances, are becoming popular. This data includes keywords as well as their syntactic parameters. A kernel function can be thought of as a text similarity measure: given a set of labeled instances, kernel methods determine the label of a novel instance by comparing it to the labeled training instances using this kernel function. Nearest neighbor classification and support-vector machines (SVMs) are two popular examples of kernel methods. Compared to kernel methods, syntactic generalization (SG) can be considered as structure-based and deterministic; linguistic features retain their structure and are not represented as values. We will be forming a set of maximal common sub-trees for a pair of parse tree for two sentences as a measure of similarity between them. It will be done using representation of constituency parse trees via chunking; each type of phrases (NP, VP PRP etc.) will be aligned and subject to generalization.

A number of authors including (Cumby and Roth 2003, Moschitti, 2008, Kong and Zhou 2011) proposed several kernel functions to model parse tree properties in kernel-based machines such as perceptrons or support vector machines. In this study, instead of tackling a high dimensional space of features formed from syntactic parse trees, we apply a more structural machine learning approach to learn syntactic parse trees themselves, measuring similarities via sub-parse trees and not distances in this space. The authors define different kinds of tree kernels as general approaches to feature engineering for semantic role labeling, and experiments with such kernels to investigate their contribution to individual stages of an SRL architecture both in isolation and in combination with other traditional manually coded features. The results for boundary recognition, classification, and re-ranking stages provide systematic evidence about the significant impact of tree kernels on the overall accuracy.

Tree kernel method assumes we are dealing with arbitrary trees. In this study we are interested in properties of linguistic parse trees, so the method of matching is specific to them. We use the tree rewrite rules specific to parse trees, significantly reducing the dimension of feature space we operate with. In our other studies Galitsky et al.,

2011) we used ontologies, further reducing the size of common subtrees. Table 6 performs the further comparative analysis of tree kernel and SG approaches:

Structural method allows combining learning and rule-based approaches to improve the accuracy, visibility and explainability of text classification. Explainability of machine learning results is a key feature in industrial environment. Quality assurance personnel should be able to verify the reason for every decision of automated system. Visibility shows all intermediate generalization results, which allows tracking of how class separation rules are built at each level (pair-wise generalization, generalization ^ sentence, generalization ^ generalization, (generalization ^ generalization) ^ generalization, etc.) Among the disadvantages of SVM (Suykens et al., 2003) are a lack of transparency of results: it is hard to represent the similarity as a simple parametric function, since the dimension of feature space is rather high. Overall, a tree kernel approach can be thought as statistical AI, and proposed approach follows along the line of logical AI traditionally applied in linguistics two-three decades ago. In this work we will apply machine learning of parse trees in the forms of SG to the content pipeline and address its engineering and architectural issues.

In our earlier studies [25, 26, 28] we approached the text learning problem as parse tree learning one based on syntactic generalization. The motivation was to explore how a richer set of linguistic features such as constituency parse trees can provide richer semantic information and therefore provide more accurate and efficient solution for text classification. We also applied graph learning to other domains [29] such as understanding complex dialogues with conflicts. We performed the comparative analysis for the accuracies of SG versus tree kernel (TK) methods in a number of applied NLP tasks (Galitsky et al 2015) including search, text classification and information retrieval and evaluation benchmark tasks including discourse analysis. In most cases the accuracy of SG and TK are similar, since the overall accuracy is determined by the richness of the feature space rather than the learning framework. In this study we implement both SG and TK and perform the joint evaluation of these techniques in industrial settings with the baseline approaches.

Although TK might give a couple of percent higher accuracy in some domains, it is much slower and consumes much higher memory. We draw the comparison between TK and SG approach in Table 1a. Based on these findings, we selected SG and not kernel-based approaches as suitable for industrial applications.

Table 1a: comparison of features of TK and SG approaches to computing text similarity

| Task or feature | TK | SG |
|---|---|---|
| Speed of ranking | Extremely computationally intensive, comparable with parsing | Finding maximal common sub-phrases is much faster than parsing |
| Phrase rewriting and normalization | Not applied and is expected to be handled by SVM | Rewriting patterns are obtained from literature. Rewriting/normalization significantly reduces the dimension of learning. |
| Handling semantic features | Semantic features are extracted and added to feature space for syntactic features. | Semantics is represented via logic forms. There is a mechanism to build logic forms from generalizations |
| Expressing similarity between phrases, sentences, paragraphs | Distance in feature space | Maximal common sub-object, retaining all common features: sub-phrase, sub- sentence, sub-paragraph |
| Ranking search results | By SVM classification score, classifying into two classes: correct and incorrect answers | |
| Integration with reasoning-based components | N/A | Results of SG can be fed to a default reasoning system, abduction/inductive reasoning system like Jasmine (Galitsky et al., 2007), domain-specific reasoning system |

| | | |
|---|---|---|
| | | like reasoning about actions |
| Combining search with taxonomy and other search features | Requires a separate taxonomy-based relevance component | SG operation is naturally combined with taxonomy tree matching operation (Galitsky 2015), other ranking mechanisms based on location, popularity, user-based recommendation |
| Using manually formed relevance rules | Requires a separate component, impossible to alter SVM feature space explicitly | Relevance rules in the form of generalizations can be added, significantly reducing dimension of feature space where learning occurs. |
| Combining with deep learning approach | N/A | Word2vec models are naturally used when generalizing two words. |

The computational linguistics community has assembled large data sets on a range of interesting computational linguistic problems. Some of these problems can be reduced to a standard classification task by appropriately constructing features; however, others require using and/or producing complex data structures such as complete parse trees and operations on them [26]. In our previous work [27, 61] we built a paragraph-level structure which is a sum of parse trees of sentences and called it *parse thicket*. In this paper we rely on the operation of generalization on the pair of parse trees (syntactic generalization, SG) for two sentences and demonstrate its role in sentence classification and other text similarity assessment tasks. Operation of generalization is defined starting from the level of lemmas to chunks/phrases and all the way to paragraphs/texts.

Most current learning research in NLP employs particular statistical techniques inspired by research in speech recognition, such as hidden Markov models (HMMs) and probabilistic context-free grammars (PCFGs). A variety of learning methods including decision tree and rule induction, neural networks, instance-based methods, Bayesian network learning, inductive logic programming, explanation-based learning, and genetic algorithms can also be applied to natural language problems and can have significant advantages in particular applications [41]. In addition to specific learning algorithms, a variety of general ideas from traditional machine learning such as active learning, boosting, reinforcement learning, constructive induction, learning with background knowledge, theory refinement, experimental evaluation methods, PAC learnability, etc., may also be usefully applied to natural language problems [12]. In this study we employ nearest neighbor type of learning, which is relatively simple, to focus our investigation on how expressive can similarity between texts be measured. The proposed text similarity assessment technique is intended to detect weak semantic signals, essential in a number of content pipeline tasks such as de-duplication and establishing links between data components (compare with [44]). Other more complex learning techniques can be applied, being more sensitive or more cautious, after we confirm that our measure of syntactic similarity between texts is adequate.

## 1.2 From search to personalized recommendations

One of the purposes of the content pipeline is to provide user recommendations. In this section we introduce a social profile-based personalized recommendation task. It relies on a special case of SG where instead of syntactic information we use ontology of categories. However we use the terms SG for all text relevance tasks in the content pipeline.

In the eBay entertainment domain, recommendations include performances and performers, movies and shows, as well as other *things to do*. Personalized recommendations are becoming more and more popular to enable people to efficiently get products and services. Internet entrepreneurs have started to believe that personalization is one of the next big steps towards a more semantic web. Everything users “like” on sites like Facebook gives others information about the things users are interested in. If one gets enough of that kind of data, as well as similar data from the people he is connected to, he can effectively judge a person's tastes and interests.

Social data-based personalization is an important feature of context-aware systems which can both sense and react based on their environments. Although a high number of successful user interfaces employ user behavior, interpersonal trust and attempt to recognize user intentions [4,22,53] a context-aware methodology of adjustment to prior knowledge about users is still to be developed.

A number of systems (Amazon, last.fm [38], Hunch [33]) have been released to personalize the Internet by getting to know users and then making smart recommendations about what they might like. There are systems functioning in horizontal domain, as well as specific areas like music and shopping. In the former case, the objective is to build a graph of users' tastes of the whole web, connecting web users with their affinities for all sorts of their everyday activities, from electronics to vacations.

Although users are in the process of starting to appreciate the value of personalization and learn to adjust their profiles for efficient online personalization, the relevance and timeliness of personalization is still quite low. In this study we address the root problem of personalization quality, propose a solution for Zvents.com of eBay, and evaluate it for a vertical recommendation domain.

## 2. A content pipeline and its relevance-related problems

The content pipeline to be introduced was designed and implemented in eBay Entertainment / Ticket sales domain. The relevance technology that supports this pipeline has been first deployed and evaluated at AllVoices.com. Then SG-supported relevance was used at a *things-to-do* recommendation portal Zvents.com, acquired by StubHub, the ticket sales site of eBay in 2011 to be an entertainment content provider. Although we evaluated relevance processing in the entertainment domain, the expectations are that relevance can be supported in any content domain in a similar manner.

The content pipeline includes data mining of web and social networks, content aggregation, reasoning, information extraction, question answering and advertising. The accuracy of search and recommendation is primarily determined by the quality of content, which in turn depends on the accuracy of operations with content by each pipeline component. The latter is essentially a relevance-based operation, therefore the higher its accuracy is, the better is the performance of the overall content portal.

Our presentation is focused on the support of content pipeline units by the SG and other *engines*, web mining, classification and rules. We enumerate the components of content pipeline, and its units with the focus on those where relevance assessment between various portions of texts is required. In Evaluation section we will do three kinds of assessments for the contribution of SG:

1. How stand-alone performance of content units is affected by SG;
2. How the performance of the overall search & recommendation system is affected by SG;
3. How search relevance itself is supported by SG.

In the production environment, SG component includes an implementation of linguistic syntactic generalization algorithm only. In our evaluation settings, we perform the comparison of SG versus tree kernel (TK) and also against the baseline algorithms generally accepted in industry, such as Lucene TF*IDF and Weka. The evaluation version of both SG and TK are available at https://github.com/bgalitsky/relevance-based-on-parse-trees.

### 2.1 Content pipeline architecture

The input for the content pipeline includes various sources of information that are intended to be provided for users. The output is the search and recommendation index which stores the refined information ready to be gives as a search or recommendation result. Also, the content pipeline provides the feed for other search engines that include the structured results in entertainment domain, such as Bing.com (Fig. 1).

We first enumerate the four components in the content pipeline, then their units, followed by enumerating problems being solved by an SG or other engines within each unit.

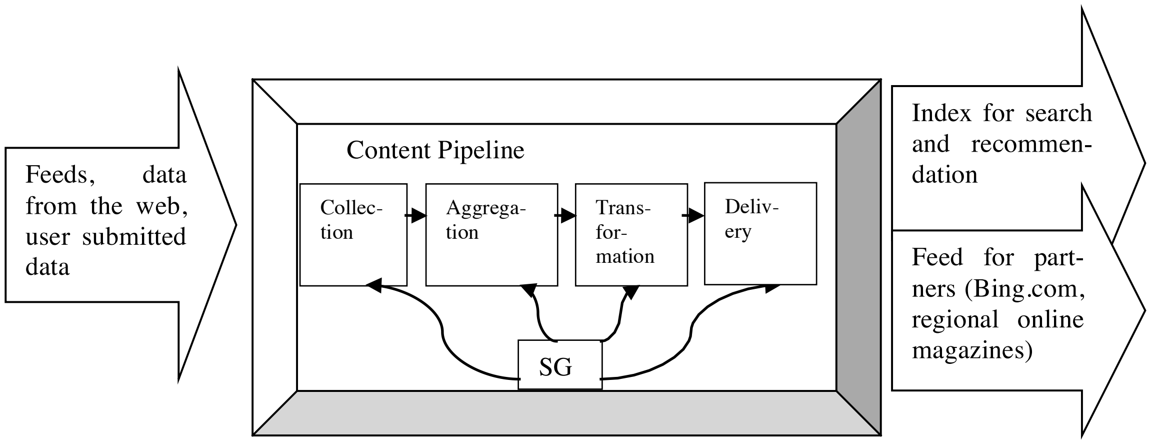

Feeds, data from the web, user submitted data
Content Pipeline
Collec-tion
Aggrega-tion
Trans-for-mation
Deliv-ery
SG
Index for search and recommen-dation
Feed for part-ners (Bing.com, regional online magazines)

Fig.1: Input, output and the components of content pipeline and its relevance support by SG

Content pipeline includes the four following component (Fig. 2):

1. *Content collection* from multiple sources (automated finding content on the web relevant to given topic, feed processing;
2. *Content aggregation*, cleaning, enrichment (deduplication, cleaning, entity extraction, forming links between different pieces of content with the same entity, auto-generation or discovering of missing pieces of content, compare with (Mavridis et al 2014));
3. *Content transformation* to a form for search and recommendation (coordinating factual and opinionated content, building index for search and recommendation;
4. *Content delivery* (search and recommendation for web and mobile, personalization, search engine marketing, sharing with partners).

The first component, *Content Collection*, uses various sources to collect pieces of content to assure as broad and deep coverage of a domain as possible. It combine feed from content partners, usually well structured, with content found on the web, usually unstructured but timely and opinionated, frequently of higher interest to users. Such content is collected from various sources on the web, from blogs and forums to public chats and social network sources. Taxonomy of entities is built here to support search [58].

The second component, *Content Aggregation* makes sure the pieces of content are transformed into a cohesive form, without redundancies and duplications. Irrelevant, ambiguous portions of content need to be filtered, as well as the one of authors with low reputations and from sources which are not reputable. Opinionated content should be separated from factual. Also, pieces of content about entities should be coordinated, for example a description of a concert, description of its performer, and a magazine article of this performer.

The third component, *Content Transformation* concerns building index for search and recommendation. To do that, we need to take into account the strength of association of information with entities, factual vs opinionated content for an entity such as a performer (event data vs user feedback), entities position it time and geo-location (where and when a concert is happening). Content inversion groups the portions of text by entertainment entities, better suitable for recommendations. This step performs ranking of pieces of content with respect to entities and their attributes (such as a performer and her music genre).

The fourth component, *Content Delivery* performs content distribution to uses and content partners. Users consume content in the active form (searching), passive form (recommendation), individually adjusted form (personalization), based on knowledge about this user, obtained, in particular, from social network sources. Partners consume the content for further aggregation. Content delivery occurs on the web and via mobile, and for the latter a web mining support for speech recognition is required, to filter out meaningless speech recognition results.

The content pipeline can be viewed as the one converting arbitrary unstructured content on the web to a well-structured, tagged, rated content with properly inter-connected pieces (referential integrity), ready to be consumed by users and content partners. For example, Bing consumes pipeline content as a set of structured text fields, tagged with types of user interest, with ratings in such dimensions as kids vs adult, professional vs amateur. However, at the same time, Bing is the source of this content: it is mined from the web using Bing search engine API. For Bing, the content pipeline converts arbitrary pieces of text distributed in the web (and indexed by Bing) into a structured form.

Each of the above components contains a number of processing units, and some of them need to rely on SG and other *engines*. We view the content pipeline from the standpoint of how relevance assessments are supported by SG engine (Fig.2). Once we have to establish a relation between two portions of texts, varying from a phrase to a paragraph, we use the SG component. For the de-duplication component, we need a very close distance between texts to confirm respective items as relevant. Conversely, for an article-item match component, which finds an internal item (such as an event) in an arbitrary article on the web, the distance between texts *for content relatedness* can be rather high. On the other hand, harvesting, automated finding content on the web, domain taxonomy, inversion of content, building search index and other components are supported by other engines.

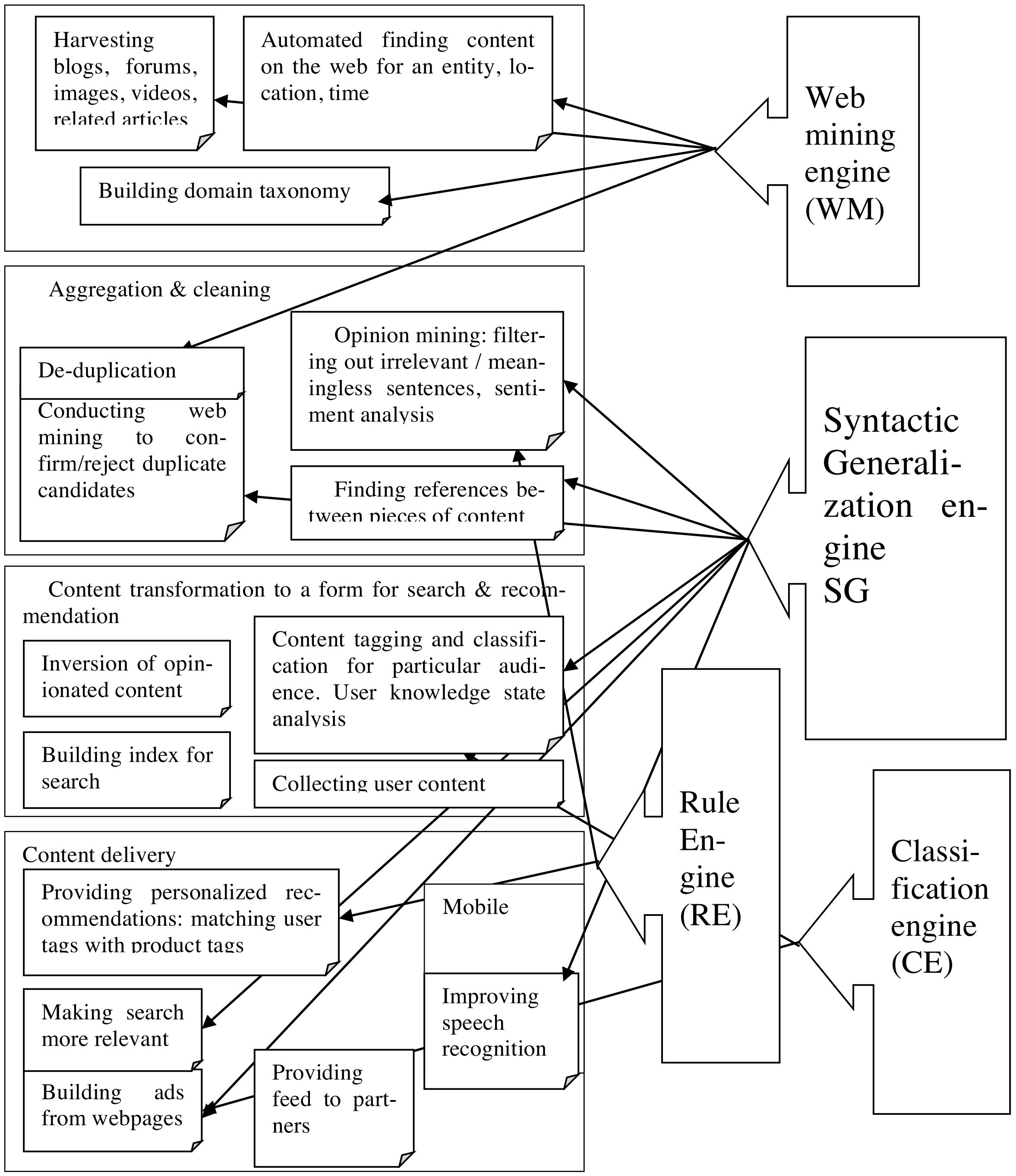

Harvesting blogs, forums, images, videos, related articles
Automated finding content on the web for an entity, location, time
Building domain taxonomy
Web mining engine (WM)
Aggregation & cleaning
De-duplication
Conducting web mining to confirm/reject duplicate candidates
Opinion mining: filtering out irrelevant / meaningless sentences, sentiment analysis
Finding references between pieces of content
Syntactic Generalization engine SG
Content transformation to a form for search & recommendation
Inversion of opinionated content
Building index for search
Content tagging and classification for particular audience. User knowledge state analysis
Collecting user content
Rule Engine (RE)
Classification engine (CE)
Content delivery
Providing personalized recommendations: matching user tags with product tags
Mobile
Making search more relevant
Improving speech recognition
Building ads from webpages
Providing feed to partners

Fig.2: Detailed architecture of the content pipeline. The content processing components are on left and the engines are on the right.

## 2.2 Content processing engines

We enumerate content processing engines, each of which supports a number of units in the content pipeline:

1) Syntactic Generalization (SG) engine which computes similarity between portions of text.
2) Rule Engine (RE) which deals with rule systems for information extraction, sentiment analysis, and personalization units when the rules are inconsistent and the results of rule application depend on the order the rules are applied.
3) Web Mining (WM) engine which uses search engine API to look for information on a particular topic, extracts information from the web on a given entity, and also compares webpages related to entities on the web to establish a correlation between them.
4) Text classification engine (CE), which takes a text and classifies it, given a set of texts for positive and negative training sets. In the opinion mining unit, it is necessary to relate a sentence into two classes, e.g. *informative vs uninformative* opinion. In ad generation unit, the classes of *suitable* vs. *unsuitable are* a basis for ad generation. In the Knowledge state of a user assessment unit, the classes are *highly knowledgeable or unknowledgeable users*. Other classification tasks include *relevant/irrelevant answer, and plausible/implausible speech recognition result*. In these tasks, decision about belonging to a class cannot be made given occurrence of the specific keywords or their frequency patters; instead, peculiar and implicit linguistic information needs to be taken into account. It is rather hard to formulate and even to imagine keyword-based classification rules for these problems, hence SG is used; however finding plentiful examples for respective classes is quite easy.

## 2.3 Content processing units

### 2.3.1 Harvesting unit

Providing detailed information on local events which are less popular on a global scale but fairly important to locals, is a key competitive advantage of the personalization technology being described. The system runs a web search for event theme keywords and location-based keywords (WM support), and then finds web search results which looks like events, according to certain rules (RE support). These rules include time and date constraints, price information, and some linguistic patterns specific to event descriptions (CE support). We have formed the training dataset, and machine learning helps to filter out search results (and web domains) which are not events. Using this technology, we can add local events not available via feeds or the event fetching procedure, where we extract events from the page with known format. This component is capable of harvesting tail events since we don't need to know specific web domains or web page formats.

Since the events are obtained by the content acquisition system from multiple sources, they are frequently presented to users apart from their context. When content comes from feed or manually entered, frequently background knowledge on events, venues and performers is missing or limited. It is sometimes hard for a user to judge how interesting and appealing a given event is, first just by looking at title, short description, and a single image of a performer (if available). To enhance the user experience, we harvest additional visual and opinion data to provide more information about similar events, performers and locations. We mine for images and videos from major search engines and apply additional relevance verification, making sure entities from image captions are the same as performers, locations and other entities of events. Five-ten thumbnail images assist as well in the go / no-go decision of a user. Verification of entities, obtained via WM, is based on SG engine. What people write about their experience with similar events in past, how they like the performance is an important data which supports personalized recommendations. We use major blog search engines and also verify relevance of posting to the event by SG.

### 2.3.2 Content mining unit

Content mining units relies on W*eb Mining* engine which takes an entity and searches for its instances on the web, for given location and time. For example, entity='*fairs & festivals*' and location = '*Boston*'. Web mining component runs search engine API, such as Bing, Yahoo, Google or Yandex to obtain search results for *(fair OR festival) AND Boston*. Then web mining unit needs to classify search result into events and not events, and for events verify that they are indeed *fairs* or *festivals, and* that they indeed occur in Boston.

It is of utmost importance that the content includes the latest and most accurate information on performers and venues. For the latter, we use web mining and search result filtering to confirm that currently stored information for a venue is correct. We filter out search engine-optimized results trying to find an original source of venue data, avoiding aggregator sites. A number of machine-learned rules are in action to verify venue address from the most authentic source found.

For performers, we verify how then are represented in a number of entertainment-related websites, assess the consistency of information among them and compute the estimate of popularity.

### 2.3.3 Taxonomy unit

Taxonomies are used to filter out irrelevant search results by verifying that important keywords from a query are also in a search result. The taxonomy construction process starts from the seed entities and mines available source domains for new entities associated with these seed entities. New entities are formed by applying the machine learning of syntactic parse trees (their generalizations) to the search results for existing entities to form commonalities between them. These commonality expressions then form parameters of existing entities, and are turned into new entities at the next learning iteration. SG is used to form these commonality expressions.

To use the taxonomy to filter out irrelevant answers, we search for a taxonomy path (down to a leaf node, if possible) that is closest to the given question in terms of the number of entities from this question. Then, this path and leaf node most accurately specifies the meaning of the question, and constrains which entities *must* occur and which *should* occur in the answer, to be considered relevant. If the n-th node entity from the question occurs in the answer, then all $k < n$ entities should occur in it as well. More details are available in [60].

### 2.3.4 Opinion mining unit

The purpose of opinion mining is to support product recommendation. When a user searches for a product, we provide the information about this product in the form of opinions from other users. If a user is unsure which product to choose, we provide recommendation based on experience of other users, similar to the given user (if such user data is available). Hence we need to extract sentences containing opinions about products, which can form an appropriate recommendation basis. There are the following steps in forming the review quotes for recommendation:

1) For a given sentence, confirm that it contains an opinion about a product, and therefore is appropriate for recommendation. Sentences which contain sentiments about entities other than product usability should be filtered out.
2) For a given opinionated sentence, determine the knowledge state of a user to be provided with recommendation. If the user is very knowledgeable about a product, this sentence needs to be more specific. If the user is a beginner, this sentence should be of rather general nature. Hence each recommendation sentence needs to be classified into a class with respect to a user knowledge state.
3) Sentiment analysis, where polarity and topicality needs to be determined. Using topicality (a product name, its feature or a category of features) is required to assign a given sentence to a product being recommended. A polarity is important to back up recommendation: a given product is recommended because associated sentiment is positive and other products, whose sentiments are negative, is *not* being recommended.

Traditionally, opinion mining problems is formulated as finding and grouping a set of sentences expressing sentiments about given features of products, extracted from customer reviews of products. A number of comparison shopping sites (Buzzilions.com 2014) are showing such features and the 'strength' of opinions about them as a number of occurrences of such features. However, to increase user confidence and trust in extracted opinion date, it is advisable to link aggregated sentiments for a feature to original quotes from customer reviews; this significantly backs up review-based recommendations by a shopping portal.

Among all sentences mentioning the feature of interest, some of them are indeed irrelevant to this feature, does not really express customer opinion about this particular features (and not about something else). For example, 'I don't like touch pads ' in reviews on Dell Latitude notebooks does not mean that this touchpad of these notebook series is not good, instead, we have a general customer opinion on a feature which is not expected to be interesting to another user. One can see that this problem for an opinion sentence has to be resolved for building highly trusted opinion mining applications.

We believe this classification problem is rather hard one and require a sensitive treatment of sentence structure, because a difference between meaningful and meaningless sentence with respect to expressed opinion is frequently subtle. A short sentence can be meaningless, its extension become meaningful, but its further extension can become meaningless again.

The problem of classification of knowledge states of a user who is a recipient of recommendation is a more conventional classification problem, where we determine what kind of response a user is expecting:

- A general recommendation,
- an advice on a series of products, a brand, or a particular product,
- a response and feedback on information shared, and others.

For each knowledge state (such as *a new user, a user seeking recommendations, an expert user sharing recommendations, a novice user sharing recommendation*) we have a training set of sentences, each of which is assigned to this state by a human expert. For example (knowledge states are italicized),

"I keep in mind no brand in particular but I have read that Canon makes good cameras" □ *user with one brand in mind,* "I have read a lot of reviews but still have some questions on what camera is right for me" □ *experienced buyer*. We expect the proper knowledge state to be determined by syntactically closest representative sentence.

Transitioning from keywords match to SG is expected to significantly improve the accuracy of knowledge state classification, since these states can be inferred from the syntactic structure of sentences rather than explicitly mentioned most of times. Hence the results of SGs of the sentences form the training set for each knowledge state will serve as classification templates rather than common keywords among these sentences.

### 2.3.5 De-duplication unit

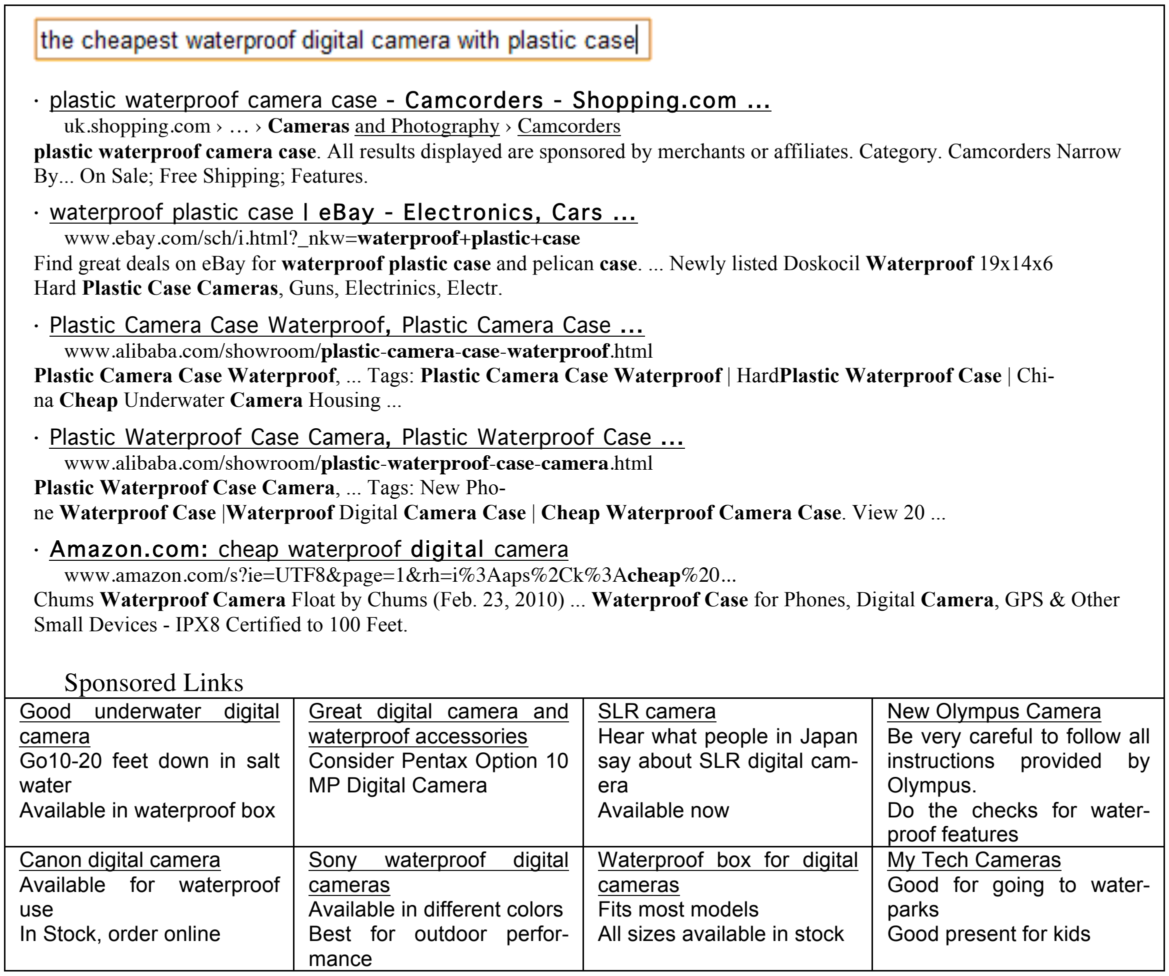
De-duplication is an essential component of any content pipeline. All entities like performers, events and venues are subject to de-duplication. We use a rather sophisticated rule system with multiple layers of exceptions, implemented via nonmonotonic reasoning –based architecture of RE, machine learning and web mining. WM takes advantage of the experience web search engines have accumulated on how people search refer to performers, events and venues in various ways, based on which search results they actually used.

When we have a pair of similar titles for an data item such as an event, a performer, a venue, we want to determine if they are the same entities or different. We search them on the web using search engine API such as Bing, and compare how similar are search results. This transition from the level of short titles, where it is hard to determine the identity, to the web search results, where we can compare one set of these against the other, is a key to high recall de-duplication. If search results do not overlap much, it means that items are not duplicates, and if they do, the items are duplicates. SG can be applied to the titles and descriptions of the candidate duplicate items, but the results are much more accurate if SG is applied to the sets of search result snippets.

We also look up the entities being de-duped at various entertainment domain-specific sites and social network sites to make sure most possible phrasings to express each entity is covered. We compute what we call "similarity in web search space" among the search results for a pair of entities to reject/confirm that they are identical.

For venues, the system extracts the relationship of being a sub-venue (like a room in a building) or a sibling between venues, which is important to provide a precise location for events and clarify how multiple events occurring in a given location are related to each other. A number of rules which are based on taxonomy of terms in venue titles, as well as address normalization rules, are applied, and being constantly machine learned to improve the accuracy and integrity of content [23].

The value of this unit is high because besides providing index for Zvents.com and eBay.com entertainment search, it also provides the stream of events for Bing.com entertainment search.



the cheapest waterproof digital camera with plastic case

· plastic waterproof camera case - Camcorders - Shopping.com ...
uk.shopping.com › … › **Cameras** and Photography › Camcorders
**plastic waterproof camera case**. All results displayed are sponsored by merchants or affiliates. Category. Camcorders Narrow By... On Sale; Free Shipping; Features.

· waterproof plastic case | eBay - Electronics, Cars ...
www.ebay.com/sch/i.html?_nkw=**waterproof+plastic+case**
Find great deals on eBay for **waterproof plastic case** and pelican **case**. ... Newly listed Doskocil **Waterproof** 19x14x6 Hard **Plastic Case Cameras**, Guns, Electrinics, Electr.

· Plastic Camera Case Waterproof, Plastic Camera Case ...
www.alibaba.com/showroom/**plastic-camera-case-waterproof**.html
**Plastic Camera Case Waterproof**, ... Tags: **Plastic Camera Case Waterproof** | Hard**Plastic Waterproof Case** | China **Cheap** Underwater **Camera** Housing ...

· Plastic Waterproof Case Camera, Plastic Waterproof Case ...
www.alibaba.com/showroom/**plastic-waterproof-case-camera**.html
**Plastic Waterproof Case Camera**, ... Tags: New Phone **Waterproof Case** |**Waterproof** Digital **Camera Case** | **Cheap Waterproof Camera Case**. View 20 ...

· Amazon.com: cheap waterproof digital camera
www.amazon.com/s?ie=UTF8&page=1&rh=i%3Aaps%2Ck%3A**cheap**%20...
Chums **Waterproof Camera** Float by Chums (Feb. 23, 2010) ... **Waterproof Case** for Phones, Digital **Camera**, GPS & Other Small Devices - IPX8 Certified to 100 Feet.

Sponsored Links

| | | | |
|---|---|---|---|
| Good underwater digital camera<br>Go10-20 feet down in salt water<br>Available in waterproof box | Great digital camera and waterproof accessories<br>Consider Pentax Option 10 MP Digital Camera | SLR camera<br>Hear what people in Japan say about SLR digital camera<br>Available now | New Olympus Camera<br>Be very careful to follow all instructions provided by Olympus.<br>Do the checks for waterproof features |
| Canon digital camera<br>Available for waterproof use<br>In Stock, order online | Sony waterproof digital cameras<br>Available in different colors<br>Best for outdoor performance | Waterproof box for digital cameras<br>Fits most models<br>All sizes available in stock | My Tech Cameras<br>Good for going to waterparks<br>Good present for kids |

Fig. 3: SEM unit for automated generation of ads from a webpage

### 2.3.6 Search Engine Marketing unit

We build 3-line advertisements (ads) in a specific format to mimic ads for search engine paid advertisement. The ad is automatically built from an arbitrary product page, published with the content mined by the above units. Features of products or services as described on the landing page are extracted along with 'advertising' language such as positive sentiment and calls to action. The formulation of ad relies on appropriate product/service descriptions from the landing page (Fig. 3).

Its practical value is to assist search engine marketing (SEM) personnel in writing ads. Given the content of a website and its selected landing page, the system needs to select sentences which are most suitable to form an ad (Fig. 3). The regular search results are on the top, and auto generated ads from these pages are on the bottom.

This is a semantic information extraction problem where rules need to be formed automatically (a similar class of problem was formulated in [50]). To form criteria for an expression to be a candidate for an advert line, we apply SG to the sentences of the collected training sets, and then form templates from the generalization results, which are expected to be much more accurate and less sensitive to a broad range of possible phrasings than just sets of keywords under traditional keyword-based IE approach. Also, using a database of existing adverts on Google, the system is capable of finding the closest ad and adjusting it to the given web page.

We combine two following technique implemented as parallel sub-units:

1) Syntactic Information extraction problem is formulated as extracting and modifying three expressions from a web pages such that these expressions:
   - serve as a page abstract, are as close as possible to the content of the page. Every line of an ad contains one of the important messages from the webpage, and may include the name of brands, companies and products from this page.
   - obey the conventional ad style: certain meaning is shared between them, one of these expression should be imperative, all of them should be positive and mention important well understood parameters.

2) Template-based approach; it finds a series of existing adverts mined on the web, which are semantically close to the given webpage, and modifies them to form original advert with the lines matching the content of this webpage. This component assures that resultant advert is a typical ad in terms of phrasing.

For example, for a fragment of a webpage like

At Smartbank **we believe in great loan deals**, **that's why we offer 5.9% APR typical on our loans of $27,500 to $35,000**.. It's also why we pledge to pay the difference if you're offered a better deal elsewhere.
What you get with a personal loan from Smartbank:
* An instant decision if you're an Online Banking customer and **get your money in 3 hours**, if accepted†
* Our price guarantee: if you're offered a better deal elsewhere we'll pledge to pay you the difference between loan repayments***
* **Apply to borrow up to $35,000**
* No fees for arrangement or set up
* Fixed monthly payments, so you know where you are
* Optional tailored Payment Protection Insurance.

We generate two ads:

Great Loan Deals
5.9% APR typical on loans of
$27,500 to $35,000. Apply now!

Apply for a Smartbank loan
We offer 5.9% APR typical
Get your money in 3 hours

### 2.3.7 Speech recognition semantics unit

This unit assures we have a reliable speech recognition, for example when searching for events via a mobile app. A typical speech recognition SDK such as the one from Google gives a number of candidates for a given utterance, so that some of them are meaningful phrases and some of them are not.

*"remember to buy milk tomorrow from trader joes",*
*"remember to buy milk tomorrow from 3 to jones"*

One can see that the former is meaningful, and the latter is meaningless (although similar in terms of how it is pronounced). A mistake caused by trying to interpret a meaningless request by a query understanding system such as *Siri* for *iPhone* can be costly. For event search by voice, this component significantly improved the accuracy (not evaluated in this study).

WM engine supported by SG comes into play: if for a given phrase its web search results are similar to this phrase (someone has said something similar somewhere), we can assume the phrase is meaningful. If we cannot find anything on the web expressed in a similar way, then we can assume that this expression is meaningless. For the more accurate analysis, this algorithm is applied to sub-phrases as well. The key here is to assess the similarity between the recognized candidate phrase and the web search results, which is performed by SG.

### 2.3.8 Search unit

SG engine improves search relevance by measuring similarity between query and sentences in search results (or snapshots) by computing SG. Such syntactic similarity is important when a search query contains keywords which form a phrase, domain-specific expression, or an idiom, such as “shot to shot time”, “high number of shots in a short amount of time”. Usually, a search engine is unable to store all of these expressions because they are not necessarily sufficiently frequent, however make sense only if occur within a certain natural language expression. Further details are provided in [25, 61].

### 2.3.9 Personalization unit

Personalization of search results to take into account user interest and user profile becoming a must in todays content delivery systems, In particular, a number of recommendation systems uses social profile to tailor search results to needs of a given user. Nowadays, when integration and access control with social sites like Facebook has been mostly solved, the main reason for low relevance is the existence of inconsistent mappings between the categories of interests as specified in social sites like Facebook and LinkedIn, and the internal categories of content providers such as eBay product categories. In fact, there is strong disagreement between how the set of user interests are stored in social sites and how such interests are associated with categories of product in a vertical domain of a recommendation system. In particular, Facebook stores user interests at individual level (user likes) and category level (categories of user likes) for the wide range of interests. Since our recommendation engine is focused on 'things to do', most of the existing Facebook categories are irrelevant, but those which are relevant are too coarse and provide limited and uneven coverage of our domain of events. Hence we need a systematic way to map horizontal domain categories and individual "likes" into the product attributes and categories in a vertical domain. In this study, we use Defeasible Logic Programming (García and Simari 2004), an argumentative framework based on logic programming to define such mapping where categories expressed in the same worlds frequently have different meanings and are therefore inconsistent.

The main purpose of personalized recommendation delivery in dynamic domain as attending events includes:

- A user specifies her interests only once (in her Facebook profile) but thoroughly so that personalized recommendation can be produced in a wide variety of domains, from things to do to consumer products.
- Selecting an event for a given date, a user does not have to manually run queries for all kinds of events she is interested in; instead, she logs in with her personal profile and sees what is happening according to her interests.
- Personalization is expected to impress customers with unique treatment of interests of themselves and their friends supporting such social features as trust.

In terms of search implementation, this can be done in two steps:

1) Keywords are formed from query in a conventional manner, and search hits are obtained by TF*IDF also taking into account popularity of hits, page rank and others.
2) The above hits are filtered with respect to syntactic similarity of the snapshots of search hits with search query. Parse tree generalization comes into play here.

Hence we obtain the results of the conventional search and calculate the score of the generalization results for the query and each sentence and each search hit snapshot. Search results are then re-sorted and only the ones syntactically close to search query are assumes to be relevant and returned to a user.

# 3. Generalization of Texts

To measure of similarity of abstract entities expressed by logic formulas, a least-general generalization was proposed for a number of machine learning approaches, including explanation based learning and inductive logic programming. Least general generalization was originally introduced by [45]. It is the opposite of most general unification [47] therefore it is also called *anti-unification*. For two words of the same POS, their generalization is the same word with POS. If lemmas are different but POS is the same, POS stays in the result. If lemmas are the same but POS is different, lemma stays in the result.

Let us represent a meaning of two natural language expressions by logic formulas and then construct unification and anti-unification of these formulas. Some words (entities) are mapped into predicates, some are mapped into their arguments, and some other words do not explicitly occur in logic form representation but indicate the above instantiation of predicates with arguments. How to express a commonality between the expressions?

- *camera with digital zoom*
- *camera with zoom for beginners*

To express the meanings we use logic predicates *camera(name_of_feature, type_of_users)* (in real life, we would have much higher number of arguments), and zoom(*type_of_zoom)*. The above NL expressions will be represented as:

*camera(zoom(digital), AnyUser)*
*camera(zoom(AnyZoom), beginner),*

where variables (uninstantiated values, not specified in NL expressions) are capitalized. Given the above pair of formulas, unification computes their most general specialization *camera(zoom(digital), beginner),* and anti-unification computes their most specific generalization, *camera(zoom(AnyZoom), AnyUser).*

At syntactic level, we have generalization of two noun phrases as:

*{NN-camera, PRP-with, [digital], NN-zoom [for beginners]}.*

We eliminate expressions in square brackets since they occur in one expression and do not occur in another. As a result, we obtain

*{NN-camera, PRP-with, NN-zoom]},* which is a syntactic analog as the semantic generalization above.

Notice that a typical scalar product of feature vectors in a vector space model would deal with frequencies of these words, but cannot express such features as co-occurrence of words in phrases, which is frequently important to express a meaning of a sentence and avoid ambiguity.

Since the constituent trees keep the sentence order intact, building structures upward for phrases, we select constituent tree to introduce our phrase-based generalization algorithm. The dependency tree has the word nodes at different levels and each word modifies another word or the root. Because it does not introduce phrase structures, the dependency tree has few nodes than the constituent tree and is less suitable for generalization. Constituent tree explicitly contains word alignment-related information required for generalization at the level of phrases. We use (OpenNLP 2013) system to derive constituent trees for generalization (chunker and parser). Dependency-tree based, or graph-based similarity measurement algorithms [11, 25] are expected to perform as well as the one we focus on in this paper.

The purpose of an abstract generalization is to find commonality between portions of text at various semantic levels. Generalization operation occurs on the levels of *Text*/*Paragraph*/*Sentence*/*Individual word*.
At each level except the lowest one, individual words, the result of generalization of two expressions is a *set* of expressions. In such set, for each pair of expressions so that one is less general than other, the latter is eliminated. Generalization of two sets of expressions is a set of sets which are the results of pair-wise generalization of these expressions.

For a given pair of words, only a single generalization exists: if words are the same in the same form, the result is a node with this word in this form. We refer to generalization of words occurring in syntactic tree as *word node*. If word forms are different (e.g. one is single and other is plural), then only the lemma of word stays. If the words are different but only parts of speech are the same, the resultant node contains part of speech information only and no lemma. If parts of speech are different, generalization node is empty.

For a pair of phrases, generalization includes all *maximum* ordered sets of generalization nodes for words in phrases so that the order of words is retained. In the following example

*To buy digital camera today, on Monday*
*Digital camera was a good buy today, first Monday of the month*

Generalization is {*<JJ-digital, NN-camera> ,<NN- today, ADV,Monday>*} , where the generalization for noun phrases is followed by the generalization by adverbial phrase. Verb *buy* is excluded from both generalizations because it occurs in a different order in the above phrases. *Buy - digital - camera* is not a generalization phrase because *buy* occurs in different sequence with the other generalization nodes.

To optimize the calculation of generalization score, we conducted a computational study to determine the POS weights to deliver the most accurate similarity measure between sentences possible [28]. The problem was formulated as finding optimal weights for nouns, adjectives, verbs and their forms (such as gerund and past tense) such that the resultant search relevance is maximum. Search relevance was measured as a deviation in the order of search results from the best one for a given query (delivered by Google); current search order was determined based on the score of generalization for the given set of POS weights (having other generalization parameters fixed). As a result of this optimization performed in [25], we obtained $W_{NN} = 1.0$, $W_{JJ} = 0.32$, $W_{RB} = 0.71$, $W_{CD} = 0.64$, $W_{VB} = 0.83$, $W_{PRP} = 0.35$ excluding common frequent verbs like *get/ take/set/put* for which $W_{VBcommon}= 0.57$. We also set that

$W_{<POS,*>}$ =0.2 (different words but the same POS), and $W_{<*,word>}$ =0.3 (the same word but occurs as different POSs in two sentences).

Generalization score (or similarity between sentences $sent_1$, $sent_2$) then can be expressed as sum through phrases of the weighted sum through words

*$word_{sent1}$* and *$word_{sent2}$*

*$score(sent_1, sent_2)$ =*

$$\sum_{\{NP, VP, \ldots\}} \sum W_{POS}\ word\_generalization(word_{sent1}\ word_{sent2}).$$

(Maximal) generalization can then be defined as the one with the highest score. This way we define a generalization for phrases, sentences and paragraphs.

Result of generalization can be further generalized with other parse trees or generalization. For a set of sentences, totality of generalizations forms a lattice: order on generalizations is set by the subsumption relation and generalization score. We enforce the associativity of generalization of parse trees by means of computation: it has to be verified and resultant list extended each time new sentence is added. Notice that such associativity is not implied by our definition of generalization.

## 3.1 Simplified example of generalization of sentences

We present an example of generalization operation of two sentences. Intermediate sub-trees are shown as lists for brevity. Generalization of distinct values is denoted by '*'. Let us consider three following sentences:

*I am curious how to use the digital zoom of this camera for filming insects.*

*How can I get short focus zoom lens for digital camera?*

*Can I get auto focus lens for digital camera?*

One can see that the second and third trees are rather similar, so it is straight-forward to build their common sub-tree as an (interrupted) path of the tree:

{ *MD-can, PRP-I, VB-get, NN-focus, NN-lens, IN-for JJ-digital NN-camera* }. At the phrase level, we obtain:

```
Noun phrases: [ [NN-focus NN-* ],  [JJ-digital NN-camera ]]
Verb phrases: [ [VB-get NN-focus NN-* NN-lens IN-for JJ-digital NN-camera ]]
```

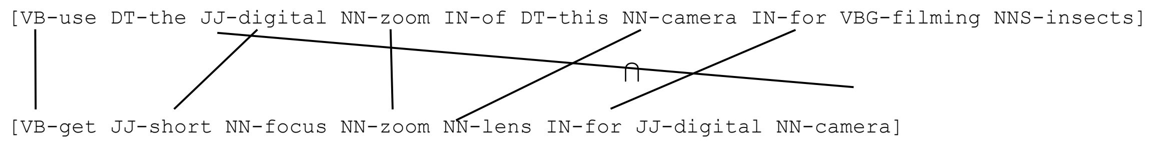
One can see that common words remain in the maximum common sub-tree, except 'can' which is unique for the second sentence, and modifiers for 'lens' which are different in these two sentences (shown as *NN-focus NN-* NN-lens*) . When sentences are not as similar as sentences 2 and 3, and we proceed to their generalization on phrase-by-phrase basis. Below we express the syntactic parse tree via chunking [1], using the format <position (POS – phrase)>.

**Grouped phrases 1** `[[NP [DT-the JJ-digital NN-zoom IN-of DT-this NN-camera ], NP [DT-the JJ-digital NN-zoom ], NP [DT-this NN-camera ], NP [VBG-filming NNS-insects ]], [VP [VBP-am ADJP-curious WHADVP-how TO-to VB-use DT-the JJ-digital NN-zoom IN-of DT-this NN-camera IN-for VBG-filming NNS-insects ], VP [TO-to VB-use DT-the JJ-digital NN-zoom IN-of DT-this NN-camera IN-for VBG-filming NNS-insects ], VP [VB-use DT-the JJ-digital NN-zoom IN-of DT-this NN-camera IN-for VBG-filming NNS-insects ]]`

**Grouped phrases 2** `[[NP [JJ-short NN-focus NN-zoom NN-lens ], NP [JJ-digital NN-camera ]], [VP [VB-get JJ-short NN-focus NN-zoom NN-lens IN-for JJ-digital NN-camera ]] ]`

## 3.2 Sample generalization between phrases

At the phrase level, generalization starts with finding an alignement between two phrases, where we attempt to set a correspondence between as many words as possible between two phrases. We assure that the alignment operation retains phrase integrity: in particular, two phrases can be aligned only if the correspondence between their head nouns is established. There is a similar integrity constraint for aligning verb, prepositional and other types of phrases.



```
[VB-use DT-the JJ-digital NN-zoom IN-of DT-this NN-camera IN-for VBG-filming NNS-insects]

[VB-get JJ-short NN-focus NN-zoom NN-lens IN-for JJ-digital NN-camera]
=
[VB-* JJ-* NN-zoom NN-* IN-for NN-*]
```

Here we show the mapping between either words or respective POS to explain how generalization occurs for each pair of phrases for each phrase type. Six mapping links between phrases correspond to six members of generalization result links. The resultant generalization is shown in bold in the example below for verb phrases VP.

**Generalization result:**

NP [ [JJ-* NN-zoom NN-* ], [JJ-digital NN-camera ]]
VP [ [VBP-* ADJP-* NN-zoom NN-camera ], **[VB-* JJ-* NN-zoom NN-* IN-for NN-* ]**
PP [ [IN-* NN-camera ], [IN-for NN-* ]]

score(NP) = $(W_{<POS,*>} + W_{NN} + W_{<POS,*>}) + (W_{NN} + W_{NN}) = 3.4$,
score(VP) = $(2 * W_{<POS,*>} + 2 * W_{NN}) + (4 W_{<POS,*>} + W_{NN} + W_{PRP}) = 4.55$, and
score(PRP) = $(W_{<POS,*>} + W_{NN}) + (W_{PRP} + W_{NN}) = 2.55$, hence score = 10.5.

One can see that that such common concept as '*digital camera*' are automatically generalized from the examples, as well as the verb phrase "*be some-kind-of zoom camera*" which expresses the common meaning for the above sentences. Notice the occurrence of expression [digital-camera] in the first sentence: although *digital* does not refer to *camera* directly, we merge two noun group and *digital* becomes one of the adjective of this resultant noun group with its head *camera*. It is matched against the noun phrase reformulated in a similar way (but with preposition *for*) from the second sentence with the same head noun *camera*.

## 3.3 Tree Kernel approach for text similarity

Instead of comparing the parse trees of a question and the parse trees of an answer directly, we can try to convert discourse features into a form for statistical machine learning and use it to rank answers. The idea of measuring the similarity between the question-answer pairs for question answering instead of the question-answer similarity turned out to be fruitful (Moschitti and Quarteroni, 2011). The classifier for correct vs incorrect answers processes two pairs at a time, $<q_1,a_1>$ and $<q_2,a_2>$, and compares $q_1$ with $q_2$ and $a_1$ with $a_2$, producing a combined similarity score. Such a comparison allows it to determine whether an unknown question/answer pair contains a correct answer or not by assessing its distance from another question/answer pair with a known label. In particular, an unlabeled pair $<q_2,a_2>$ will be processed so that rather than ''guessing" correctness based on words or structures shared by $q_2$ and $a_2$, both $q_2$ and $a_2$ will be compared with their corresponding components $q_1$ and $a_1$ of the labeled pair $<q_2, a_{2>}$ on the grounds of such words or structures. Because this approach targets a domain-independent classification of an answer, only the structural cohesiveness between a question and answer , and not 'meanings' of answers is leveraged.

We follow along the lines of this approach and consider an arbitrary sequence of sentences instead of question-sentence and answer-sentence pairs for text classification. Our positive training paragraphs are "plausible" sequences of sentences for our class, and our negative training paragraphs are "implausible" sequences, irrespective of the domain-specific keywords in these sentences. In our opinion, for candidate answer selection tasks, such structural information is important but insufficient. At the same time, for the text classification tasks just structure analysis can suffice for proper classification.

We now proceed to define a kernel method for individual sentences and then explain how it is extended to the case of parse thickets. Kernel methods are a large class of learning algorithms based on inner product vector spaces, such as SVM. Convolution kernels as a measure of similarity between trees compute the common sub-trees between two trees $T_1$ and $T_2$ (Altınel et al 2015). The convolution kernel does not have to compute the whole space of tree fragments. Let the set $\tau = \{t_1, t_2, \ldots, t_{2|\tau|}\}$ be the set of sub-trees of an extended parse tree and $\chi_i(n)$ be an indicator function that is equal to 1 if the sub-tree $t_i$ is rooted at a node $n$ and is equal to 0 otherwise. A tree kernel function over trees $T_1$ and $T_2$ is

$$\mathrm{TK}(T_1, T_2) = \sum_{n_1 \in N_{T1}} \sum_{n_2 \in N_{T2}} \Delta(n_1, n_2) \qquad (3)$$

where $N_{T1}$ and $N_{T2}$ are the sets of $T_1$ 's and $T_2$ 's nodes, respectively and

$$\Delta(n_1, n_2) = \sum_{i=1}^{|\tau|} \chi_i(n_1)\chi_i(n_2) \quad (4)$$

(4) calculates the number of common fragments with the roots in $n_1$ and $n_2$ nodes.

To define the tree kernel for a parse thicket as an ordered set of parse trees with connected nodes, we extend the notion of a parse tree for a sentence to incorporating a portion of the tree for another sentence that is linked to the given sentence via a discourse arc. For every arc that connects two parse trees, we derive the extension of these trees, extending branches according to the arc (Fig. 4a). In this approach, for a given parse tree, we will obtain a set of its extension, so the elements of the kernel will be computed for many extensions, instead of just a single tree. The problem here is that we need to find common sub-trees for a much higher number of trees than the number of sentences in the text; however, by subsumption (sub-tree relation) the number of common sub-trees will be substantially reduced.

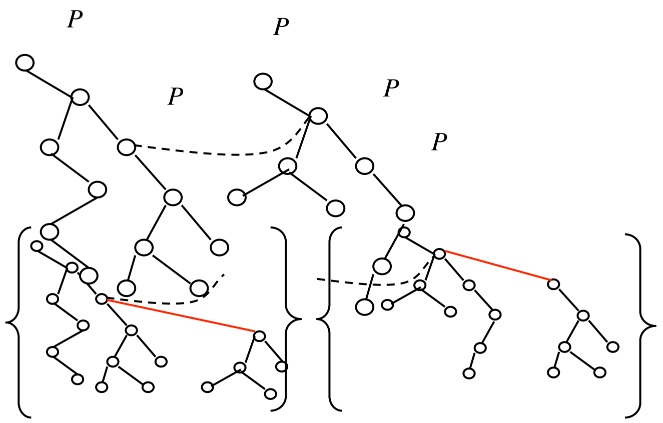


Fig. 4a: An arc which connects two parse trees for two sentences in a text (on the top) and the derived set of extended trees (on the bottom).

The algorithm for building an extended tree for a set of parse trees $T$ is as follows:

**Input:**

1) Set of parse trees $T$.

2) Set of relations $R$, which includes relations $R_{ijk}$ between the nodes of $T_i$ and $T_j$: $T_i \in T, T_j \in T, R_{ijk} \in R$. We use index $k$ to range over multiple relations between the nodes of parse tree for a pair of sentences.

**Output:** the exhaustive set of extended trees $E$.

```
Set $E = \varnothing$;
For each tree $i$=1:|$T$|
  For each relation $R_{ijk}$, $k$= 1: |R|
    Obtain $T_j$
    Form the pair of extended trees $T_i * T_j$;
    Verify that each of the extended trees do not have a super-tree in $E$
     If verified, add to $E$;
Return $E$.
```

Notice that the resultant trees are not the proper parse trees for a sentence, but nevertheless form an adequate feature space for tree kernel learning.

## 3.4 Phrase-level generalization

Although the generalization is defined as the set of maximal common sub-graphs, its computation is based on matching phrases. To generalize a pair of sentences, we perform chunking and extract all noun, verb, prepositional and other types of phrases from each sentence. Then we perform generalization for each type of phrases, attempting to find a maximal common sub-phrase for each pair of phrases of the same type. The resultant phrase-level generalization can then be interpreted as a set of paths in resultant common sub-trees (Galitsky et al 2012).

Generalization of parse thickets, being a maximal common sub-graph (sub-parse thicket) can be computed at the level of phrases as well as a structure containing a maximal common sub-phrases. However, the notion of phrases is extended now: thicket phrases can contain regular phrases from different sentences. The way these phrases are extracted and formed depends on the source of non-syntactic link between words in different sentences: thicket phrases are formed in a different way for communicative actions and RST relations. Notice that the set of regular phrases for a parse thicket is a sub-set of the set of thicket phrases (all phrases extracted for generalization). Because of this richer set of phrases for generalization, the parse thicket generalization is richer than the pair-wise thicket generalization, and can better tackle variety in phrasings and writing styles, as well as distribution of information through sentences.

## 3.5 Computing maximal common subgraphs

To find maximal subPT we use a reduction of the maximal common subgraph problem to the maximal clique problem [27] by constructing the edge product. The main difference with the traditional edge product is that instead of requiring the same label for two edges to be combined, we require non-empty generalization results for these edges. Though computationally not optimal, this approach is convenient for prototyping.
To solve the maximal subPT problem, we transform it to the maximal clique problem [27] . This can be done by constructing an edge product PT. Let $G_1 = (V_1, E_1, \alpha_1, L_1)$ and $G_2 = (V_2, E_2, \alpha_{21}, L_2)$ be PTs with nodes V and edges E, where $\alpha : V \to L$ is a function assigning labels to the vertices, and L is a finite non-empty set of labels for the edges and vertices. The edge product PT $H_e = G_1 \circ G_2$ includes the vertex set $V_H = E_1 \circ E_2$ in which all edge pairs $(e_i, e_j)$ with $1 \le i \le |E_1|$ and $1 \le j \le |E_2|$ have to produce non-empty generalization in their edge labels. Also, these edge pairs must produce non-empty generalizations in their corresponding end vertex labels. Let $e_i = (u_1, v_1, l_1)$ and $e_j = (u_2, v_2, l_2)$. The labels coincide if $l_1 \cap l_2 \neq \varnothing$ and $\alpha_1(v_1) \cap \alpha_2(v_2) \neq \varnothing$ and $\alpha_1(u_1) \cap \alpha_2(u_2) \neq \varnothing$.

There is an edge between vertices $e_H, f_H \in V_H$ where $e_H = (e_1, e_2)$ and $f_H = (f_1, f_2)$ if two edge pairs are compatible, meaning that they are distinct: $e_1 \neq f_1$ and $e_2 \neq f_2$. Also, either condition for these edges of resultant PT should hold:

- $e_1, f_1$ in $G_1$ are connected via a vertex of the label which produces non-empty generalization with the label of the vertex shared by $e_2, f_2$ in $G_2 : \{vertices\ for\ e_1, f_1\} \cap \{vertices\ for\ e_2, f_2\} \neq \varnothing$. We label and call them ***c-edges***, or
- $e_1, f_1$ and $e_2, f_2$ are not adjacent in $G_1$ and in $G_2$, respectively. We label and call them ***d-edges***.

To get a common subPT in $G_1$ and $G_2$ each edge pair in $G_1$ and $G_2$ (vertex pair in $H_e$) has to have generalizable label with all other edge pairs in $G_1$ and $G_2$ (vertex pair in $H_e$), which are forming a common subgraph. In this way a clique in $H_e$ corresponds to common subPTs $G_1$ and $G_2$.

A subset of vertices of $G$ is said to be a clique if all its nodes are mutually adjacent. A maximal clique is one which is not contained in any larger clique, while a maximum clique is a clique having largest cardinality.

After finding all **maximal** cliques for all pairs (representing question and answer for instance) we look through all results and range them due to cliques cardinality. In this case the more edge pairs the result of generalization of two paragraphs contains the more relevant it is.

# 4. Generalization of expressions of interest

In SG algorithm, default operation is a matching of parse trees for two texts. The only exception is when one of these texts is a canonic social profile expression, such as Facebook likes, and another is a regular text to be matched with, such as a candidate event name. Once we have a canonic social profile expression, we don't need to act on the syntactic level since we know what kind of entity it is. Instead, we need to do a category mapping between Facebook likes and names of entities, such as event names.

## 4.1 Personalization algorithm as intersection of *likes*

We can define vertical personalization as finding a set of recommendations which are the overlap of two sets:

- *InterestSet*: all possible subjects of user interests (all events) we believe a user would be interested in according to what she specified;

- *LocationDateSet*: all events available at a specified location at a specific time.

In this setting, we can define a new set *Recommendation* = *InterestSet* ∩ *LocationDateSet*. Since *InterestSet* is specified as two sets of <*Likes, Categories*>, as long as *LocationDateSet* can be tagged with the same tags and categories, the problem is solved. If overlap of *likes* is too small (unimportant), events with *categories of likes* will be formed as the desired intersection. Note that <*Likes, Categories*> is frequently redundant: *Likes* derive *Categories* unambiguously but not the other way around.

Without personalization, using a conventional search engine, a user would have to explicitly search for each of her <*Likes, Categories*>, or form a respective OR query, to find this intersection. This is happening today when a user is searching the web for' things to do' this weekend. However, not all Facebook likes are equally meaningful. Some of the likes were specified because the user is confident in her interests, whereas another set of likes is encouraged by various Facebook apps and therefore not as indicative of real user interest, and might be too unimportant. We use the following mechanism to differentiate between these two classes of likes (important/ unimportant):

1) Using friends: all *likes* shared by friends
2) Using dynamics of how *likes* appeared: initial set of likes are assumed to be important, clusters of likes encouraged by various Facebook apps are unimportant, likes of weakly represented categories are unimportant as well, whereas well-represented categories of likes are important.

Once we have <*Likes, Categories*>of *InterestSet,* we first try to find important likes, then unimportant likes, and finally different likes but from *Categories* in *LocationDateSet*.

The remaining problem is to map two set of categories for *Likes, CategoriesSrc* for source and *CategoriesDest* for destination. For this problem we will apply argumentation techniques for dealing with potentially inconsistent and contradictory information.

## 4.2 Mapping categories of interest / taxonomies

What is available for Facebook likes for the domain of entertaining events are as follows:

*CategorySrc* = {*Retail, Consumer_products, Sports_athletics, Sports_teams, Athlete, Television, Comedian, Clubs, Food_beverage , Musicians, Health_beauty, Actor, Writer, Restaurants, Fashion, Comedian, Musician/band, Games, Musicians, Movie, Tv show, Television, Album, Actor/director, Film, Bars, Education, Nonprofit, Song*}.

As the reader can see these categories are overlapping, and belong to various level of generalization and granularity. These categories have to be mapped into types of events, venues such as restaurants and theaters, and particular music genres:

*CategoryDest* = {*Arts & Crafts, Community, Business & Tech, Dance, Fairs & Festivals, Food & Dining, Music,*
*Performing Arts, Sports & Outdoors, Visual Arts*} *(higher-level categories)* ∪
{ *Fairs & Festivals* /{*sport festivals*} excluding other kinds of festivals} ∪ {sub-categories including *Jazz, R&B* and *Soul, Rock, Techno & Dance, Country, Classical, Folk & Traditional* ….}

Mapping between categories can be described as

*Sports_athletics → Sports & Outdoors*/{*soccer, hiking …*}
excluding {*camping, bird-watching*} ∪ *Dance*/{*gymnastics*} excluding other kinds of *dance* ∪
*Fairs & Festivals* / {*sport festivals*} excluding other kinds of *festivals*}

As an essentially deterministic categorization, we would avoid applying statistical and fuzzy mapping here; instead we prefer a systematic way to handle inconsistencies between source and target categorizations. Deterministic mapping better fits current software development methodology, making this mapping fully controllable and therefore well-suited for commercial environments (compare with approaches to reasoning related to argumentation in [9, 46].

The rules (clauses) for the target category above would look like: *sports_outdoors :- sports_athletics OR (outdoors, ┐camping, ┐bird-watching*) OR (*dance, gymnastics*) OR (*fairs_festivals & sport_festival*). We now proceed to the systematic treatment of inconsistencies among such rules using Defeasible Logic Programming, an argumentative framework based on logic programming (García and Simari 2004).

## 4.3 Defeasible logic programming-based rule engine

To deal with inconsistent rules, the rule engine need to implement a mechanism of rule justification, rejecting certain rules in the situation when other rules have fired. We select Defeasible Logic Programming (DeLP) [30] approach and present an overview of the main concepts associated with it. One of the key applications of the rule engine is category mapping, and we will show an example of how categories can be mapped in order to determine recommendation categories given social profile categories.

A *Defeasible logic program* is a set of facts, strict rules Π of the form (A:-B) , and a set of defeasible rules Δ of the form A-<B, whose intended meaning is "if B is the case, then usually A is also the case". A *Category mapping* DeLP program includes facts which are formed from *likes,* and strict and defeasible clauses where the heads and bodies corresponds to the sets Category1 and Category2. A given DeLP includes a part from a social profile that contains facts (likes), and a fixed set of mapping rules which include positive and negative occurrences of categories.

Let P=(Π, Δ) be a DeLP program and L a ground literal. A *defeasible derivation* of L from P consists of a finite sequence $L_1, L_2, \ldots, L_n = L$ of ground literals, such that each literal $L_i$ is in the sequence because:

(a) $L_i$ is a fact in Π, or

(b) there exists a rule $R_i$ in P (strict or defeasible) with head $L_i$ and body $B_1, B_2, \ldots, B_k$ and every literal of the body is an element $L_j$ of the sequence appearing before $L_j$ ($j < i$ ).

Let h be a literal, and P=(Π, Δ) a delp program. We say that <A, h> is an *argument* for h, if A is a set of defeasible rules of Δ, such that:

1. there exists a defeasible derivation for h from =(Π ∪ A)
2. the set (Π ∪ A) is non-contradictory, and
3. A is minimal: there is no proper subset $A_0$ of A such that $A_0$ satisfies conditions (1) and (2).

Hence an argument <A, h> is a minimal non-contradictory set of defeasible rules, obtained from a defeasible derivation for a given literal h associated with a program P.

We say that $<A_1, h_1>$ *attacks* $<A_2, h_2>$ iff there exists a sub-argument <A, h> of $<A_2, h_2>$ ($A \subseteq A_1$) such that h and $h_1$ are inconsistent (i.e. Π ∪ {h, $h_1$} derives complementary literals). Our analysis will be focused on those attacks corresponding to *defeaters*. When comparing attacking arguments, we will assume a partial preference ordering on arguments (given e.g. by specificity as in [30]).

Specificity, for example, is a syntactic criterion preferring those arguments that are more direct (ie. requiring less inference steps) or more informed (those based on more premises are preferred over those based on less information). This preference criterion is modular in DeLP, and in fact other criteria are possible (see e.g. [3]), where numerical values are propagated via modus ponens and used for comparing arguments).

We will say that <$A_1$, $h_1$> *defeats* <$A_2$, $h_2$> if <$A_1$, $h_1$> attacks <$A_2$, $h_2$> at a sub-argument <A, h> and <$A_1$, $h_1$> is strictly preferred (or not comparable to) <A, h>. In the first case we will refer to <$A_1$, $h_1$> as a *proper defeater*, whereas in the second case it will be a *blocking defeater*. Defeaters are arguments which can be in their turn attacked by other arguments, as is the case in a human dialogue. An *argumentation line* is a sequence of arguments where each element in a sequence defeats its predecessor. In the case of DeLP, there are a number of *acceptability* requirements for argumentation lines [30] in order to avoid fallacies (such as circular reasoning by repeating the same argument twice).

Based on the previous notions, DeLP queries are solved in terms of a dialectical tree, which subsumes all possible argumentation lines for a given query. The definition of dialectical tree provides us with an algorithmic view for discovering implicit self-attack relations in users' claims. Let <$A_0$, $h_0$> be an argument from a program P. A *dialectical tree* for <$A_0$, $h_0$> is defined as follows:

1. The root of the tree is labeled with <$A_0$, $h_0$>

2. Let N be a non-root vertex of the tree labeled <$A_n$, $h_n$> and

$\Lambda$= [<$A_0$, $h_0$>, <$A_1$, $h_1$>, …, <$A_n$, $h_n$>] the sequence of labels of the path from the root to N. Let [<$B_0$, $q_0$>, <$B_1$, $q_1$>, …, <$B_k$, $q_k$>] all attack <$A_n$, $h_n$>. For each attacker <$B_i$, $q_i$> with acceptable argumentation line [$\Lambda$,<$B_i$, $q_i$>], we have an arc between N and its *child* $N_i$ .

A marking on the dialectical tree can be then performed as follows:

1. All leaves are to be marked as U-nodes (undefeated nodes).
2. Any inner node is to be marked as U-node whenever all its associated children nodes are marked as D-nodes.
3. Any inner node is to be marked as D-node whenever at least one of its associated children nodes is marked as U-node.

After performing this marking, if the root node of the tree is marked as a U-node, the original argument at issue (and its conclusion) can be deemed as *justified* or *warranted*.

Let us now build an example of a dialectical tree of category mapping. Imagine we have a following set of mapping clauses and available categories obtained from likes:

*Defeasible Rules*
*sports_outdoors -< sports_athletics*
*sports_athletics -< exercise_facility.*
┐ *sports_athletics -< exercise_facility, yoga.*
┐ *sports_athletics -< chess.*
┐ *community -< food_dining, music. (*commercial, not a community event)
*music -< rock..*
┐ *sports_athletics -< exercise_facility, community* (where people do stuff different from sport)
*community -< food_dining.*
┐ *community -< business_tech.*
┐ *music -< visual_arts.*

*Facts (*facts are obtain from explicit likes*)*
*exercise_facility.*
*yoga.*
*chess.*
*rock.*
*business_tech.*
*food_dining.*
*visual_arts.*

Table 1: An example of a Defeasible Logic Program for modeling category mapping

In this category mapping to DeLP, the literal *sports_outdoors* is supported by <*A, sports_outdoors*>=
<{ (*sports_outdoors -< sports_athletics*), (*sports_athletics -< exercise_facility*)}, *sports_outdoors*> and there exist three defeaters for it with three respective argumentation lines: <$B_1$, ┐*sports_athletics*> = <{(┐*sports_athletics -< exercise_facility, yoga*)}, *sports_athletics*>.
<$B_2$,┐*sports_athletics*> = <{(┐ *sports_athletics -< exercise_facility, community*), (*community -< food_dining*) }, *sports_athletics*>.
<$B_3$, ┐*sports_athletics*> = <{(┐*sports_athletics -< chess* )}, *sports_athletics*>. The first two are proper defeaters and the last one is a blocking defeater. Observe that the first argument structure has the counter-argument, <{*sports_athletics -< exercise_facility*}, *sports_athletics)*, but it is not a defeater because the former is more specific.
Thus, no defeaters exist and the argumentation line ends there.
$B_3$ above has a blocking defeater <{(*sports_athletics -< exercise_facility*)}, *sports_athletics*> which is a disagreement sub-argument of <*A, sports_outdoors*> and it cannot be introduced since it gives rise to an unacceptable argumentation line. $B_2$ has two defeaters which can be introduced: <$C_1$, ┐*community* >, *where* $C_1$ = {(┐*community -< food_dining, music*),(*music -< rock*)}, a proper defeater, and <$C_2$, ┐*community* >, *where* $C_2$={(┐*community -< business_tech*)} is a blocking defeater. Hence one of these lines is further split into two; $C_1$ has a blocking defeater that can be introduced in the line <$D_1$, ┐*music* > , *where* $D_1$= <{(┐*music -< visual_arts*)}. $D_1$ and $C_2$ have a blocking defeater, but they cannot be introduced, because they make the argumentation line not acceptable. Hence the target category *sports_outdoor* cannot be accepted for the given user, as the argument supporting the literal *sports_outdoor* is not warranted. The dialectical tree for *A* is shown in Fig. 4.

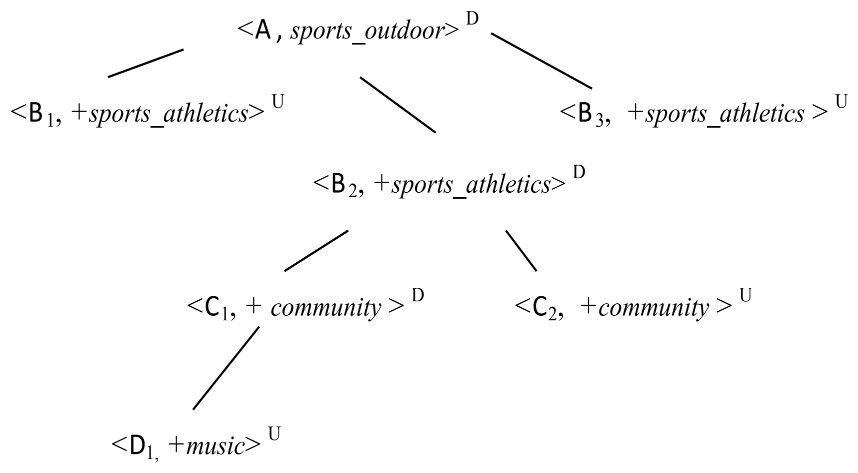


Fig.4: Dialectical tree for category *sports_outdoor* using the DeLP Category Mapping (Table 1)

Having shown how to build dialectic tree, we now ready to outline the algorithm for category mapping:

*1)* Get the list of likes from the social profile, and their list of categories *CategoriesSrc*;
*2)* Filter out unimportant categories and likes following criteria outlined above;
*3)* Add resultant set of *facts* to the fixed set of defeasible rules for category mappings;
*4)* Build a dialectic tree for each expected target category and accept/reject it based of defeasibility criterion;
*5)* Form the list of resultant target categories *CategoriesDest.*

We manually constructed 34 classical rules and 55 defeasible rules to cover the mapping of entertainment categories. A test-driven development methodology was used: first the test cases for rule mapping were constructed, followed by implementation of mapping rules which were immediately tested against these test cases.

## 5. Content pipeline algorithms

In this section we provide the algorithm details in a more formal way. Taxonomy building, de-duplication, sentiment analysis, SEM information extraction and SG engine are the areas which most heavily rely on algorithm implementation.

### 5.1 Taxonomy construction algorithm

We outline the iterative algorithm, which takes a taxonomy with its terminal nodes and attempts to extend the terminal nodes via web mining to acquire a new set of terminal nodes. At the iteration *k,* we acquire a set of nodes by extending the current terminal node $t_i$ with $t_{ik1}$ , $t_{ik2}$ …. This algorithm is based on SG which gives the least general set of texts in this form. We outline the iterative step:

Input: Taxonomy $T_k$ with terminal nodes $\{t_1, t_2 \dots t_n\}$
A threshold for the number of occurrences to provide sufficient evidence for inclusion into $T_k$: $th(k, T)$.
Output: extended taxonomy $T_{k+1}$ with terminal nodes $\{t_{1k1}, t_{1k2, \dots}, t_{2k1}, t_{2k2}, \dots t_{nk1,} t_{nk2}\}$

```
For each terminal node $t_i$
    1) Form a search query as a path from the root to $t_i$ , q = { $t_{root}$ , …, $t_i$};
    2) Run a web search for $q$ and obtain a set of answers (snippets) $A_q$.
    3) Compute SG the answers $A_q$ :
       $\Lambda (A_q) = a_1 \wedge a_2 , a_1 \wedge a_3, \dots, a_1 \wedge a_m, \dots, \dots, a_{m-1} \wedge a_m,$
```

4) Sort all of the elements (words, phrases) of $\Lambda$ ($A_q$) in descending order of the number of occurrences in $\Lambda$ ($A_q$). Retain only the elements of $\Lambda$ ($A_q$) with the number of occurrences above a threshold *th(k, T)*. We call this set $\Lambda^{high}$ ($A_q$).
5) Subtract the labels from all of the existing taxonomy nodes from $\Lambda^{high}$ ($A_q$):
$\Lambda^{new}$ ($A_q$) = $\Lambda^{high}$ ($A_q$) / $T_{k.}$
6) For each element of $\Lambda^{high}$ ($A_q$), create a taxonomy node $t_{ihk,}$ where $h \in \Lambda^{high}$ ($A_q$), and k is the current iteration number, and add the taxonomy edge
($t_{i,}$ , $t_{ihk,}$) to $T_{k.}$

---

## 5.2 De-duplication algorithms

We now describe the algorithms of de-duplication based on web mining. The idea is to assess a similarity of web search results for two entities, to decide whether they are the same entities. Since the search engines have accumulated experience on what people search for and which search results people click on, we leverage it to verify for two entities, if their corresponding search results are rather similar, then these entities are the same.

**Matching algorithm 1**: finding common expression and confirming that it constitutes an entity

---

Input: A pair of strings
Output: decision on whether this pair is the same entity => merge them, or different entities

---

1) Get a set of candidate pairs;
2) Compare entity names: select overlap keywords;
 a) See if common keywords are common words, or entity-specific words
 b) Retain overlap words which are entity specific, subtract stop words for given duplicate-item type ({event, venue, performer}).
 c) Normalize remaining words.
3) Filter out cases of too short list of overlap words, or those including all common English words.
4) Verify that 2c) is an entity by searching for it in web space
 a) Collect all title of search results and see how the candidate searched entity occur in title
 b) Filter out search results so that the candidate searched entity occurs in them in a ‘distorted’ way: there is no alignment between searched entity and obtained title
 c) Filter out cases where words other than {nouns, adjectives, gerunds} occur in the sub-title which corresponds to searched entity.
5) Count the number of all accepted search results titles for a formed entity and compare with threshold for the minimum number of such titles. If it is above the threshold, confirm that the overlap words constitute an entity
6) If overlap words constitute an entity confirm duplication, otherwise confirm that candidate pair are different entities;

*Matching algorithm 2*: comparing possibly identical entities in web search space
1) Get a set of candidate pairs.
2) Form search expressions for the pair of entities to verify if they are identical in search space.
a) For events, just take their names
b) For venues, add address and city
c) For performers, possibly add tags and/or genres if available.
3) Filter our common words from both entity names which do not bring additional constraint, such as “Monday poetry night at Blue Lagoon” □ “poetry night at Blue Lagoon”
4) Apply additional set of normalization rules for both entities in the current pair
5) Run search for each entity in the pair and compare search results.
6) For venues, the comparison score includes the similarity in URL, address and city.
7) For a set of search results for first entity, find the closest search result for the second entity, and verify if results is identical:
a) By syntactic match

b) By string edit distance

c) Do it for title and also snippet

d) If similarity is above the threshold, accept a unit of similarity in search space

8) Sum up the total number of units of similarity in search space and compare with threshold. If above the threshold, confirm the similarity between two entities, otherwise reject.

## 5.3 Sentiment analysis algorithm

To provide a product recommendation, we extract expressions of *user needs* about products and services. User needs are extensions of what is referred to as topicality in sentiment analysis and are extracted as *attachments to a sentiment expressions*. To extract them, we need to use syntactic tree, where both vertices (lemmas) and edges (syntactic links) are labeled. In a sentence, we first identify sentiment as a node (single word like 'good'), or subtree ('did not work for me') and then proceed to the *sub-tree which is dependent* (linked to) the main node in sentiment sub-tree. Over the years, we accumulated our own domain-independent vocabulary of English sentiments, coded as parsing sub-trees to be identified at parsing trees.

Let us consider the domain of digital cameras, and focus on a particular class of usability needs associated with taking pictures at night. We use a pair of tags: *night* + specific *night-related* need sub-category:

*night – picture (general, overall – taking pictures at night)*
*night>cloud (how to film clouds at night),*
*night>cold (how to film at night in cold conditions*
*night>recommend (which measures are recommended at night, general issues)*
*night>dark (filming in dark conditions)*
*night>set (what and how needs to be set)*
*night>inconsistent (for some cameras, setting seemed inconsistent to some users)*
*night>shot (peculiarities about night shot)*
*night>tripod (use of tripod at night)*
*night>mode(switch to specific filming modes for night shots)*

As one can see, the meanings for needs of filming at night vary in generality and semantic roles, and phrasings include nouns, adjectives and verbs. So the criteria of being a user need indeed have to be formulated in terms of a sub-tree, satisfying certain syntactic (tree) conditions (see [28] for more details). For a horizontal (unlimited) domain (like electronics, which is rather wide), all terms from need expressions cannot be defined via a taxonomy. Therefore, semantics of a need expression has to be inferred from the syntactic one.

Our assumption is that if there is at least one author who attaches sentiment to an expression (which we know now as an expression for need), then other people might have the same need, so it is worth storing and analyzing. In terms of syntactic tree, if a lemma for sentiment is dependent of a term T and does not have its own dependent vertices, the need expression is a sub-tree dependent on T.

Examples of extraction of two need expressions are shown at Fig. 5. For the sentiment 'great', we have a sub-tree 'in-daylight-bright' which is a need expression (use of digital cameras can be 'great', or 'not so good' in 'bright daylight'. For the sentiment '*not*…good', we have a need 'indoor-in-setting-dim. In the latter case sentiment is expressed by 'don't expect it to get good', where the main vertex is 'be', and the need expression is branching from the vertex 'get'.

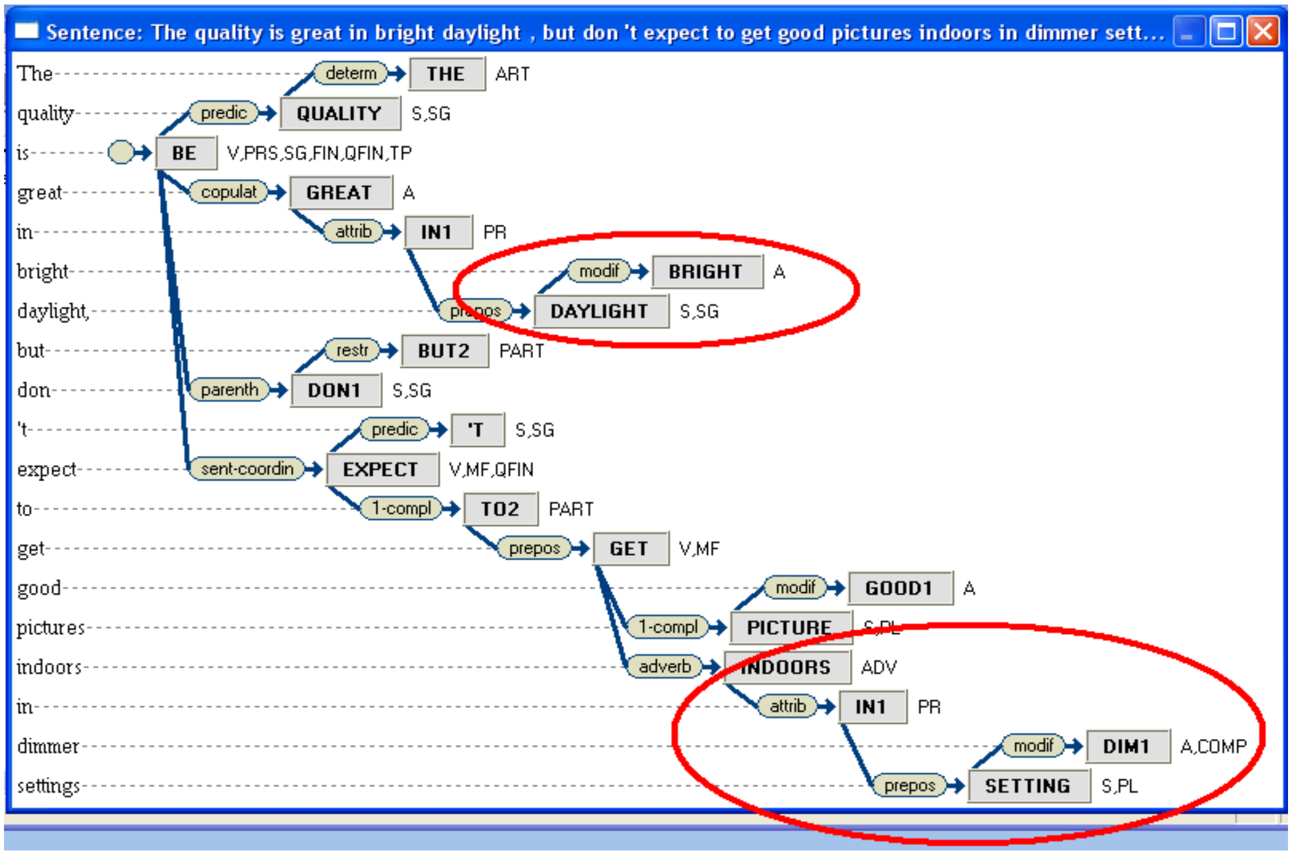
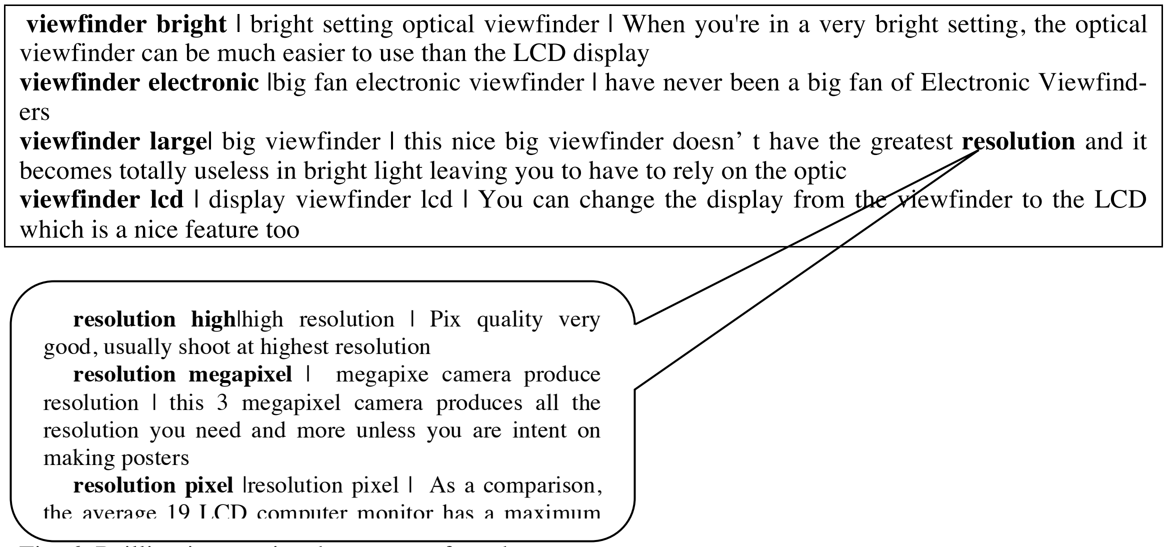

Sentence: The quality is great in bright daylight , but don 't expect to get good pictures indoors in dimmer sett...

Fig. 5: Syntactic parse tree for sentences about digital camera with two pairs of sentiment-need expressions (circumscribed).

After need expressions are extracted, they need to be normalized and grouped. Normalization transforms need expressions into sequences of words in normal form, without prepositions and articles. After that, need expressions are grouped by the main noun of expression (the closest noun to the trunk of the need expression as a sub-tree).

Let us consider an example of a group with noun *viewfinder*(Fig. 6), with the second word in grouped expression, all keywords in need expression, and original sentence. We have four need sub-categories {*bright, electronic, large, lcd*} for the need category *viewfinder*. These subcategories categorize view-finder from very different aspects. Notice that both syntactic relations between viewfinder and second word varies, as well as semantic relations, however we ignore that for the sake of forming categories and sub-categories.

Four sentences above come from different sources, the common thing between them is the product and a category of user needs about viewfinder in connection to this product.



**viewfinder bright** | bright setting optical viewfinder | When you're in a very bright setting, the optical viewfinder can be much easier to use than the LCD display
**viewfinder electronic** |big fan electronic viewfinder | have never been a big fan of Electronic Viewfind-ers
**viewfinder large**| big viewfinder | this nice big viewfinder doesn' t have the greatest **resolution** and it becomes totally useless in bright light leaving you to have to rely on the optic
**viewfinder lcd** | display viewfinder lcd | You can change the display from the viewfinder to the LCD which is a nice feature too

**resolution high**|high resolution | Pix quality very good, usually shoot at highest resolution
**resolution megapixel** | megapixe camera produce resolution | this 3 megapixel camera produces all the resolution you need and more unless you are intent on making posters
**resolution pixel** |resolution pixel | As a comparison, the average 19 LCD computer monitor has a maximum

Fig. 6: Drilling in associated category of needs

Whereas category noun is identified by a rule, a sub-category word is obtained by clustering category into clusters; sub-category word should not be a category word and should occur in more than one need expressions within a category. For more accurate identification of sub-category word more advanced methods could be used, combining machine learning and statistical analysis; it could produce higher per-centage of word pairs where meaning can be obtained just from this pair.

*Inversion of content* is a transformation of corpus of text to a set of interactive textual components where each component includes all content about given need for a given product. These components are hyperlinked to drill in and out of need categories associated with each other.

## 5.4 Taxonomy-supported search algorithm

Taxonomy-supported search algorithm takes a query *q*, runs a search (outside of this algorithm), ob-tains a set of candidate answers *a* and finds the best acceptable answer according to the definitions that we introduced in the previous section.

The input: query *q*

The output: the best answer $a_{best}$

```
1) For a query $q$, obtain a set of candidate answers $A$ by the available means (using keywords, using an internal index, or using an external index of search engine APIs);
2) Find a path in taxonomy $T_p$ that covers maximal terms in $q$, along with other paths that cover $q$, to form a set $P = \{ T_p \}$.
   Unless an acceptable answer is found:
3) Compute the set $T_p \cap q$.
   For each answer $a_i \in A$
         4) compute $a_i \cap ( T_p \cap q))$ and apply an acceptability criterion.
         5) compute the score for each $a_i$.
         6) compute the best answer $a_{best}$ and the set of acceptable answers $A_a$.
   If no acceptable answer is found, then return to 2 for the next path from P.

7) Return $a_{best}$ and the set of acceptable answers $A_a$ if available
```

## 5.5 Search engine marketing ad construction algorithm

Syntactic information extraction is a rule-based deterministic system implemented as a state machine. Rules include semantic template consisting of individual words and word forms including part of speech constraints. There are rules for beginning of extraction, termination of extraction, and acceptance of extraction. These semantic templates serve as evidence that an expression. For example, semantic template in the beginning of extraction rules verifies the evidence if expression of interest is about to follow. For example: "In our <business_name [noun phrase]> you should" is an instance of a beginning of extraction rule (verb phrase of recommended activity is to follow), "and you" is an instance of termination of extraction rule (next verb group is to follow, so the current one has ended) , and "-not"<is excluded> is an instance of acceptance rule (meaning of a negation can be ambiguous, so it is safer to reject current expression).

The procedure of semantic extraction implements the state machine, where each state is characterized by a current extraction part and position in text. Deterministic extraction rules take into account these state parameters and perform transition to the next state (iterating to the next word), changing verification of parameters for this word accordingly. For example, if noun phrase extraction is being selected, we search for the part of speech which is neither noun, adjective, adverb or number, checking occurrence of a web page entity. If an imperative expression is being extracted, we search for the end of verb phrase.

Main types of expressions are imperative verb phrases and noun phrases. In the context of ad generation we refer to the class of sentences encouraging potential customers to perform an action, be it physical or mental (knowledge or belief), as *imperative expressions*. The start of imperative expression is either an imperative verb ('sign up for …'), or an expression indicating that certain activity is expected to be performed by a user (*'we allow immediate cash withdrawal'*). In the latter case we reformulate the expression to derive an explicit imperative: ('get immediate cash withdrawal', 'take advantage of immediate cash withdrawal').

There are following constraints for a noun group which includes product entity:

1) Such entity should be derived from important webpage components such as title and/or keyword lists
2) Such entity should have a modifier or be combined with another noun group which express additional constraint for the main one including entity ('auto-focus digital camera with flat-screen LCD', 'Type3 USB slot for 2GB memory card' ).
3) This group should have more than 2 words, otherwise it is usually too brief to form a headline.

4) To have a higher weight, a noun phrase with entity should be under the focus of sentiment, webpage authors are expressing a positive sentiment about the product and/or its particular feature.

Forming an ad, we substitute an ad template with the lines of certain class:

Headline (class1): <positive sentiment><entity> OR just <Entity>
Line1: <semantic imperatives (class 2) and statement about features (class 3)> (taken from VP chunks using above rules)
Line2: <call to action, class3 or class4> (either imperative from page OR we reserved expression: "Buy Now! Only <extracted price>" | "In Stock now! Next day shipping" | "Buy Now! Worldwide delivery").

It is important to balance properly between the roles for line of each class, to maximize the number of advert derived given available set of lines.

## 5.6 An algorithm for generalization of two sentences

We outline the SG algorithm for two sentences. It deals with paths of syntactic trees rather than sub-trees, because it is tightly connected with language phrases. As a formal operation on abstract trees, generalization operation nevertheless yields semantic information about commonalities between sentences. Rather than extracting common keywords, generalization operation produces a syntactic expression that can be semantically interpreted as a common meaning shared by two sentences.

Input: Two parse trees for two sentences, or some SG results
Output: the maximum common sub-tree for the pairs of input trees

1) Obtain parsing tree for each sentence. For each word (tree node) we have lemma, part of speech and form of word information. This information is contained in the node label. We also have an arc to the other node.
2) Split sentences into sub-trees which are phrases for each type: verb, noun, prepositional and others; these sub-trees are overlapping. The sub-trees are coded so that information about occurrence in the full tree is retained.
3) All sub-trees are grouped by phrase types.
4) Extending the list of phrases by adding equivalence transformations such as noun phrase normalizations.
5) For the set of the pairs of sub-trees for both sentences for each phrase type.
6) For each pair in 5) yield an alignment [31], and then generalize each node for this alignment. For the obtained set of trees (generalization results), calculate the score.
7) For each pair of sub-trees for phrases, select the set of generalizations with the highest score (least general).
8) Form the sets of generalizations for each phrase types whose elements are sets of generalizations for this type.
9) Filtering the list of generalization results: for the list of generalization for each phrase type, exclude more general elements from lists of generalization for given pair of phrases.

## 6. Evaluation

In this section we will present the evaluation results for the units of the content pipeline which are supported by SG and other engines. Some of the units we described in Section 2 perform regular processing and do not require a special technology; these were tested as regular software units. We evaluate separately the content preparation units, and user experience-related units, proceeding from de-duplication to sentiment analysis and SEM, and then to personalization recommendation and search.

To run the SG and TK code we used for evaluation, the reader would need to build an open source project which the part of OpenNLP Similarity component available at
https://github.com/bgalitsky/relevance-based-on-parse-trees.
To perform SG based on Stanford NLP parsing and tagging results, one need to load
/src/main/java/opennlp/tools/parse_thicket/matching/Matcher.java,
and to apply it to OpenNLP results,
src/main/java/opennlp/tools/textsimilarity/chunker2matcher/ParserChunker2MatcherProcessor.java.
To run the TK, the reader needs
src/main/java/opennlp/tools/parse_thicket/kernel_interface/MultiSentenceKernelBasedSearchResultsProcessor.java

### 6.1 Performance analysis of the content pipeline components

We selected to evaluate the de-duplication unit because it its performance is a bottleneck for high quality content, and the opinion mining unit due to its importance for convincing a potential user to attend an event. We form a few datasets for each unit being evaluated, conduct independent evaluation for this dataset and then average the resultant (F-measure). Training and evaluation dataset of texts, as well as class assignments, was done by the quality assurance personnel.

One way to measure inter-annotator agreement between the quality assurance personnel is by using α measure of (Krippendorff, 2004), which treats the annotators as interchangeable and measures the difference between disagreement expected by chance vs observed disagreement:

$$\alpha = 1 - \frac{D_{observed}}{D_{chance}} = 1 - \frac{\sigma^2_{within}}{\sigma^2_{total}}$$

where $\sigma_{within}$ is standard deviation of the differences within the annotations for the same text, $\sigma_{total}$ is standard deviation of the overall difference between all annotations. The threshold of $\alpha \geq 0.80$ indicates reliable judgments, while $\alpha \geq 0.67$ is recommended as a limit to support conclusions about reliability of annotation, because the value of α below this limit makes conclusions drawn from such data not significant from the statistical viewpoint. In this section we present the evaluation settings where α exceeds 0.70.

Half of each set was used for training, and the other half for evaluation; the split was random but no cross-validation was conducted. Due to the nature of the problem positive sets are larger than negative sets for sensible/meaningless & ad line problems. We use WEKA C4.5 algorithm as a baseline approach, performing keyword-based classification.

#### 6.1.1 De-duplication: from string distance to SG-supported web mining

We compared the baseline approach of string distance with that of TK-supported and SG-supported web mining. We used the set of 1000 pairs of string for entities (performers and performances). We observed the improvement in de-duplication F-measure (is it the same entity or different entity?) proceeding from string-based algorithms to web mining with keyword match and then web mining with SG match (Section 5.2). Analysis is performed varying the number of keywords from 1 to 5 (Table 2).

The baseline computation relied on (Levenshtein 1956) and (Jaccard 1912) string distance computations.

Table 2: Evaluation results for de-duplication unit

| # of keywords in entity | Baseline (F-measure, %) | Searching for entities in web space and computing common keyword (F-measure, %) | Searching for entities in web space and computing TK between search results (F-measure, %) | Searching for entities in web space and computing SG between search results (F-measure, %) |
|---|---|---|---|---|
| 1 | 84.3 | 83.2 | 84.1 | 83.5 |
| 2 | 81.1 | 82.7 | 86.2 | 84.2 |
| 3 | 79.7 | 82.1 | 84.5 | 84.7 |
| 4 | 74.6 | 80.5 | 84.1 | 82.1 |
| 5 | 75.9 | 79.2 | 84.4 | 82.3 |

The de-duplication quality is computed as F-measure given the precision and recall. For the quality of content, false positives (entities are determined to be the same, but they are not) in de-duplications damage the content in a higher degree, since non-existent merged entities appear. False negatives (should be merged, but have not been merged) are not as bad but annoying to the users.

One can observe that the using web mining improves the F-measure of de-duplication by 4.5% on average for all entities, and SG gives further 2% when analyzing similarity of entities via web search results. Improvement of de-duplication by web mining is not helpful for single-keyword entities, but becomes noticeable for longer entity names. Contribution of web mining on its own is significantly stronger than that of SG for similarity assessment of search results of phrases. However, since we use SG to support other units anyway, we leverage it to gain extra 2% in F-measure. The SG-based web mining can be improved by 1.5% by the TK-based web mining.

### 6.1.2 Sentiment analysis for product recommendation

Sentiment analysis problem is traditionally formulated as finding a polarity of opinion for a text or short sentence like tweet. For the purpose of recommendation we focus on evaluation of the other accompanying problems: recognizing meaningful sentences to show to a user, and to recognize user knowledge state to provide a proper level of details.

For reviews, we classify each sentence with respect to sensible/meaningless classes by two approaches:

- A baseline WEKA C4.5, as a popular text classification approach
- SG - based approach.

We demonstrate that a traditional text classification approach poorly handles such a complex classification task, in particular due to slight differences between phrasings for these classes, and the property of non-monotonicity. Using SG instead of WEKA C4.5 brought us 16.1% increase in F-measure for the set of digital camera reviews (Table 3).

Table 3: Evaluation of sentiment analysis: meaningful vs meaningless sentences

| Domain | Data set size (# positive /#negative ex- | Baseline F-measure, obtained by WEKA | SG Precision relating to a class, % | SG Recall relating to a class, % | SG F-measure |
|---|---|---|---|---|---|

| | amples) | C4.5 | | | |
|---|---|---|---|---|---|
| digital camera reviews | 220/60 | 51.4% | 58.8% | 54.4% | 56.5% |
| wireless services reviews | 120/80 | 62.7% | 58.8% | 74.4% | 65.6% |
| laptop reviews | 300/100 | 65.0% | 62.4% | 78.4% | 69.5% |
| auto reviews | 250/100 | 69.2% | 74.2% | 80.4% | 77.2% |
| kitchen appliances reviews | 200/100 | 72.3% | 73.2% | 84.2% | 78.3% |

One can see that SG improves the classification F-measure by 8.2%. Notice that recognizing meaningless sentences and recognizing knowledge state of a user is different problem to a sentence-level sentiment polarity analysis, fairly popular problem nowadays, especially applied to twitter data (Go et al 2009; Pak and Paroubek 2010).

To recognize a knowledge state of a user to make recommendation more appropriate, we classified the texts users post as a question or as a sharing message in social network site such as Facebook. We manually did the assignment of user knowledge states, and use quality assurance personnel to evaluate the classification results. The same training dataset was used by Weka C4.5 as a baseline and by SG to assess a potential improvement.

Rows of Table 4 contain classification data for the reviews on different products, and variability in accuracies can be explained by various levels of diversity in phrasings. For example, the ways people express their feelings about cars is much more diverse than that of about kitchen appliances. Therefore, F-measure of the former task is lower than that of the latter. One can see that it is hard to form verbalized rules for the classes, and hypotheses are mostly domain-dependent; therefore, substantial coverage of varieties of phrasing is required.

Overall recognition F-measure of knowledge state classification is higher than for the other two domains because manually built templates for particular states cover a significant portion of cases. At the same time, recognition F-measure for a particular knowledge states significantly varies from state to state and was mostly determined by how well various phrasings are covered in the training dataset. We used the same set of reviews as we did for evaluation of meaningless sentences classification and manually selected sentences where the knowledge state was explicitly mentioned or can be unambiguously inferred. For evaluation dataset, we recognized which knowledge state exists in each of 200 sentences. Frequently, there are two or more of such states (without contradictions) per sentence: note that knowledge states overlap. Low classification F-measure occurs when classes are defined approximately and the boundary between them are fuzzy and beyond expressions in natural language. Therefore we observe that SG gives us some semantic cues that would be hard to obtain at the level of keywords or superficial parsing.

Table 4 Evaluation of sentiment analysis: recognizing a knowledge state of a user

| Knowledge state | Data set size (# positive /#negative examples) | Baseline F-measure, obtained by WEKA C4.5 | SG precision relating to a class, % | SG Recall relating to a class, % | SG F-measure, % |
|---|---|---|---|---|---|
| Beginner | 30/200 | 72.3% | 77.8% | 83.5% | 80.6% |
| User with average experience | 44/200 | 73.2% | 76.2% | 81.1% | 78.6% |
| Pro or semi-pro user | 25/200 | 70.0% | 78.7% | 84.9% | 81.7% |
| Potential buyer | 60/200 | 71.6% | 73.8% | 83.1% | 78.2% |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| Open-minded buyer | 55/200 | 69.4% | 71.8% | 79.6% | 75.5% |

On average, there is a 10.7% improvement of classification F-measure by SG. It can be interpreted as one extra correct (adjusted to user knowledge state) recommendation per 10 users.

For the sentiment polarity assessment, we used the sentiment detector [59] as a baseline. It was improved by enforcing the sentiment polarity templates for special cases of negations and combinations of negations. SG was used in a nearest neighbor setting to overwrite the decision of [59] when a given sentence gives a substantial overlap with one of the templates via SG (Table 5).

Table 5: Evaluation of sentiment analysis: polarity assessment

| Domain | Data set size (# positive /#negative examples) | Baseline F-measure, obtained by Stanford NLP Sentiment analyzer, **positive** | Baseline F-measure, obtained by Stanford NLP Sentiment analyzer, **negative** | Stanford NLP Sentiment + SG-based rules, F-measure, **positive** | Stanford NLP Sentiment + SG-based rules, F-measure, **negative** |
|---|---|---|---|---|---|
| digital camera reviews | 220/60 | 56.4% | 51.8% | 61.3% | 53.9% |
| wireless services reviews | 120/80 | 61.5% | 59.8% | 66.4% | 62.7% |
| laptop reviews | 300/100 | 63.8% | 65.2% | 71.2% | 70.8% |
| auto reviews | 250/100 | 59.6% | 68.1% | 67.6% | 68.3% |
| kitchen appliances reviews | 200/100 | 60.9% | 64.8% | 68.4% | 70.3% |

SG-based templates improves the sentiment recognition F-measure by 8.0%. The baseline approach handled negative cases better than the positive ones, and the SG template – enabled approach experienced more difficulties with negative sentiments than with positive.

### 6.1.3 Evaluation of search engine marketing unit

We selected the evaluation dataset of 1000 webpages related to the same products we used for the above evaluations. We then applied the information extraction algorithm Section 5.5 to form candidate sentences for inclusion in an ad line. The training dataset of 10000 sentences was formed from Google sponsored links scraped from search results for product-related searches. Each candidate sentence is classified as appropriate or inappropriate to form an ad line. WEKA C4.5 is used for the baseline, and nearest neighbor-based SG is used to estimate an improvement of classification into these two classes (Table 6).

Table 6: Evaluation of SEM tool: candidate expression classification F-measure to form an ad line

| Domains | Classification data set size (# positive /#negative examples) | Baseline F-measure, obtained by WEKA C4.5 | SG precision relating to a class, % | SG recall relating to a class, % | SG F-measure |
|---|---|---|---|---|---|
| digital camera webpages | 200/800 | 77.1% | 88.4% | 65.6% | 75.3% |
| wireless services webpages | 200/800 | 70.7% | 82.6% | 63.1% | 71.6% |
| laptop | 200/800 | 63.0% | 69.2% | 64.7% | 66.9% |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| webpages |  |  |  |  |  |
| auto sales webpages | 200/800 | 67.5% | 78.5% | 63.3% | 70.1% |
| kitchen appliances webpages | 200/800 | 70.2% | 78.0% | 68.7% | 73.1% |

For the SG-supported classification, there is a modest 2.4% improvement is classification, once candidate expression is extracted. It turns out that a bag-of-words approach has a satisfactory performance, measuring the similarity of candidate sentences with the elements of the training set. We believe that a better extraction technique might have a higher impact on the overall quality of built ad lines, extracted from webpages.

Precision for extraction of ad lines for the same five categories of products is higher than the one for the above tasks of sensible/meaningless classes in Section 6.1.2. A the same time recall of the former is lower than that of the latter, and resultant F-measure is slightly higher for ad lines information extraction, although the complexity of problem is significantly lower. In can be explained by a rather high variability of acceptable ad lines ('sales pitches') which have not been captured by the training set.

## 6.2 Performance analysis of personalized recommendations

In this section we perform evaluation of personalized recommendations. Since this component is the consumer of the content pipeline, this evaluation can be viewed as the one for the overall system. User interface of apps.facebook.com/discover_zvents is shown in Fig. 7.

For evaluation of personalization we split the set of personalization users into the following five groups with respect to how complete their Facebook profile, how many *likes* they have and how representative they are of user interests:

1) Novice or inactive user.
2) Intermediate user with some relevant categories (music, outdoor).
3) Intermediate users with a number of categories and likes.
4) Advanced user accumulating many likes and systematically managing them.
5) Advanced user accumulating many likes and not managing them.

For each above group, we conduct evaluation of the portion of relevant events suggested by the recommendation system. We use two recommendation scenarios:

1) A user does not specify any query and all available events are personalized. Rather larger set of events is subject to reduction (Table 7a).
2) A user specifies search query for a certain type of Events (Table 7b). Then we personalize events which satisfy the user condition; a rather small set of events is subject to reduction for personalization.

In our evaluation, each user produced a set of twenty requests and received ten events on average for each recommendation. The left column of Tables 7a and 7b indicates the percentage of events found satisfactory when most popular (for everyone) events were recommended, and the right two columns for personalization results show the satisfaction percentages for the events personalized for given user. The first personalization column shows a naïve personalization which does not use generalization and relies solely on matching *likes* as keywords. This is our personalization baseline. The second, rightmost column shows the satisfaction percentage for the personalization algorithm presented in the previous section.

Table 7a: evaluation of increase of the % of relevant recommendation without initial search query

| Satisfaction | Satisfaction without personalization,% | Satisfaction with personalization,% | |
|---|---|---|---|
| | | Without generalization of likes | With generalization of likes |
| Group 1 | 67 | 61 | 58 |
| Group 2 | 64 | 69 | 76 |
| Group 3 | 63 | 76 | 82 |
| Group 4 | 71 | 86 | 89 |
| Group 5 | 69 | 68 | 73 |

Table 7b: evaluation of increase of the % of relevant recommendation with a search query for specific kind of events

| Satisfaction | Satisfaction without personalization,% | Satisfaction with personalization,% | |
|---|---|---|---|
| | | Without generalization of likes | With generalization of likes |
| Group 1 | 64 | 56 | 59 |
| Group 2 | 68 | 68 | 74 |
| Group 3 | 66 | 73 | 81 |
| Group 4 | 69 | 79 | 87 |
| Group 5 | 68 | 72 | 80 |

What we can see in general is that the more detailed Facebook profile is, the higher the improvement in the percentage of relevant events. This is true for both sessions with search and without search. However, when there is a high number of *likes* in diverse categories which are not managed, it is hard to assess the validity of each likes and one can observe a decrease of relevance improvement by means of personalization (last rows of both columns).

Overall, one can conclude that personalization delivers recommendations for products which would not be discovered by users otherwise: they would have to manually run multiple searches per each of their current likes/interests to discover if a relevant event is happening at their location.

One of the advantages of social network-based personalization is that a user becomes aware of much more events she would discover otherwise. We evaluate the proportion of events which would be exposed to a user, and call it *event accessibility* measure:

1) Using email notification (passive approach, users get email notifications with events they would potentially attend);
2) Using search (active approach, users try to find events they might want to attend);
3) Using personalization (passive, but expected to be a high-relevance approach);

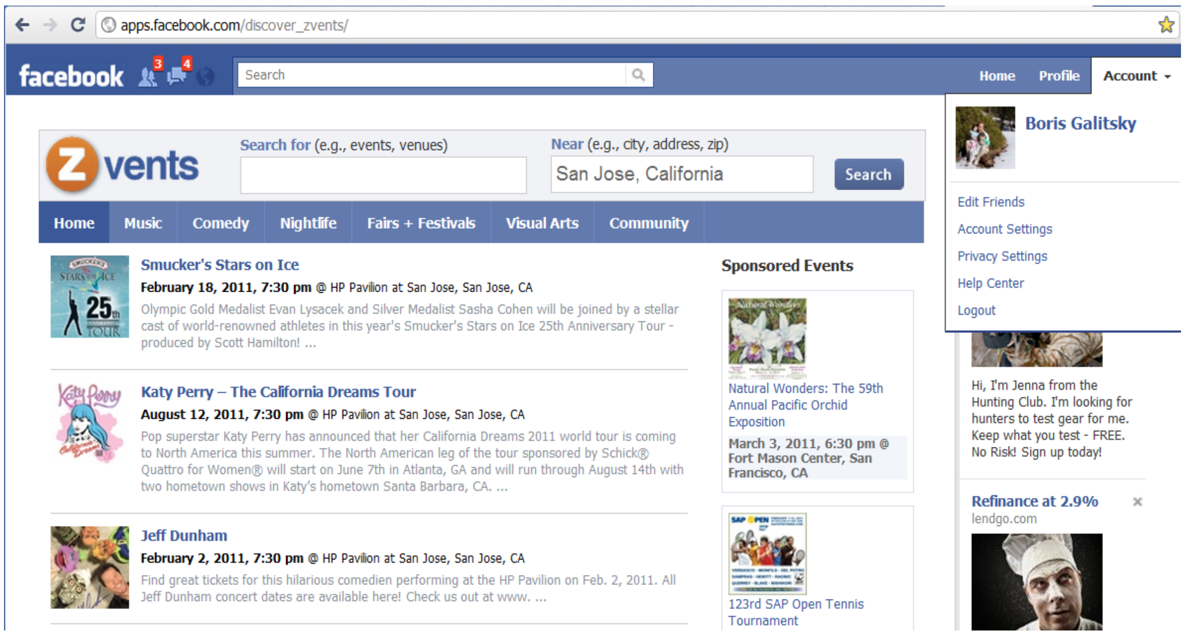
For each user we build a total set T of events we believe are of interest to a person, using means other than personalization-related. We selected a ticket purchase data and user click data as most relevant and averaged through users with similar interest to derive T for the total set of potentially interesting events for a given class of users. Then we evaluate the size of $E_1$, $E_2$ and $E_1$ as subsets of T according to our definition above.

We selected 15 major metropolitan areas and 5 averaged users with their favorite categories of events. For each of these users, we calculated $E_1$ value based on search result by location and then filtering out events with foreign categories for a given user. $E_2$ is calculated assuming average category-based search session of 5 queries, and $E_3$ is obtained as a result of personalization to the selected averaged customer profiles.

One can see that personalization gives increase of 37% over the set of events that is being sent to an average user by email. A search session gives less than a quarter of events of potential interest offered by personalization (Table 8).

| Location | T | E1 | E2 | E3 |
|---|---|---|---|---|
| New York | 13092 | 120.5008 | 41.80151 | 245.0025 |
| San Francisco | 5522 | 57.52668 | 18.76983 | 105.9543 |
| Las Vegas | 4282 | 47.99049 | 15.02309 | 40.57705 |
| Los Angeles | 4102 | 43.14766 | 14.01716 | 51.11965 |
| Boston | 3798 | 41.84619 | 12.52423 | 59.65823 |
| Chicago | 3515 | 41.03406 | 11.60954 | 40.69687 |
| Houston | 3075 | 38.4163 | 10.84504 | 32.02998 |
| Atlanta | 2757 | 27.67643 | 9.053296 | 36.87898 |
| Nashville | 2693 | 27.96139 | 9.374129 | 30.10135 |
| Austin | 2574 | 24.21604 | 9.135278 | 66.29651 |
| Denver | 2518 | 26.31187 | 8.022806 | 32.02995 |
| Lexington | 2140 | 23.17031 | 6.879982 | 18.03972 |
| Charleston | 2131 | 23.32734 | 7.239115 | 18.01083 |
| Philadelphia | 2062 | 17.99938 | 6.714275 | 27.12133 |
| San Diego | 1930 | 23.17136 | 6.062084 | 21.06925 |
| St Louis | 1910 | 17.58314 | 6.349103 | 17.23174 |
| Washington | 1875 | 17.30389 | 6.53234 | 9.099314 |
| Fresno | 1867 | 20.01329 | 6.526486 | 14.70334 |
| Seattle | 1861 | 15.6457 | 5.52391 | 30.42235 |
| Average | 3352.8 | 34.4 | 11.1 | 47.1 |
| Percentage personalization improves the number of discovered events |  | 137% | 423% | 100% |

Table 8: Categories of most popular events in cities with the highest numbers of events



**Smucker's Stars on Ice**
**February 18, 2011, 7:30 pm** @ HP Pavilion at San Jose, San Jose, CA
Olympic Gold Medalist Evan Lysacek and Silver Medalist Sasha Cohen will be joined by a stellar cast of world-renowned athletes in this year's Smucker's Stars on Ice 25th Anniversary Tour - produced by Scott Hamilton! ...

**Katy Perry – The California Dreams Tour**
**August 12, 2011, 7:30 pm** @ HP Pavilion at San Jose, San Jose, CA
Pop superstar Katy Perry has announced that her California Dreams 2011 world tour is coming to North America this summer. The North American leg of the tour sponsored by Schick® Quattro for Women® will start on June 7th in Atlanta, GA and will run through August 14th with two hometown shows in Katy's hometown Santa Barbara, CA. ...

**Jeff Dunham**
**February 2, 2011, 7:30 pm** @ HP Pavilion at San Jose, San Jose, CA
Find great tickets for this hilarious comedien performing at the HP Pavilion on Feb. 2, 2011. All Jeff Dunham concert dates are available here! Check us out at www. ...

Natural Wonders: The 59th Annual Pacific Orchid Exposition
**March 3, 2011, 6:30 pm @ Fort Mason Center, San Francisco, CA**

123rd SAP Open Tennis Tournament

Hi, I'm Jenna from the Hunting Club. I'm looking for hunters to test gear for me. Keep what you test - FREE. No Risk! Sign up today!

**Refinance at 2.9%**
lendgo.com

Fig. 7: User interface of personalization system. It gets user geo-location from her IP and its preferences from her Facebook profile. List of recommended events for a given user changes as the location changes.

## 6.3 Performance analysis of SG-supported search relevance

We conducted evaluation of relevance of SG – enabled search engine, based on Yahoo and Bing search engine APIs, used as a baseline. We base our evaluation on external APIs to avoid dependence on previous components of the content pipeline and focus on how SG and taxonomy improves search without taking into account content quality.

For our evaluation, we use customers' queries to eBay entertainment and product-related domains. We started from simple questions referring to a particular product, a particular user sentiment / need. We then proceeded to multi-sentence forum-style request to share a recommendation. To perform a comparison of SG-based search with the baseline, we used web search engines instead of eBay's on search, but ran the product-oriented queries. In our evaluation we split the totality of queries into noun-phrase class, verb-phrase class, how-to class, and also independently split in accordance to query length (from 3 keywords to multiple sentences). The evaluation was conducted by the quality assurance personnel.

To compare the relevance values between search settings, we used first 100 search results obtained for a query by Yahoo and Bing APIs, and then re-sorted them according to the score of the given search setting (syntactic generalization score and taxonomy-based score). To evaluate the performance of a hybrid system, we used the weighted sum of these two scores (the weights were optimized in an earlier search sessions). Accuracy of a particular search setting (query type and search engine type) is calculated, averaging through 20 search sessions. This measure is more suitable for product-related searches delivering multiple products, than Mean Reciprocal Rank (MRR), calculated as $1/n \sum_{i=1\ldots n} 1/rk_i$

where $n$ is the number of questions, and $rk_i$ is the rank of the first correct answer to question $i$. MRR is used for evaluation of a search for information, which can be contained in a single (best) answer, whereas a product search might include multiple valid answers.

For each type of phrase for queries, we formed a positive set of 2000 correct answers and 10000 incorrect answers (snippets) for training; evaluation is based on 20 searches. These answers were formed from the quality assurance dataset used to improve existing production search engine before the current project started. To compare the relevance values between search settings, we used first 100 search results obtained for a query by Yahoo and Bing APIs, and then re-sorted them according to the score of the given search setting (SG score). The results are shown in Table …

To further improve the product search relevance in eBay setting, we added manually formed templates that are formed to enforce proper matching with popular questions which are relatively complex, such as

*see-VB *-JJ -*{movie-NN ∪ picture-NN∪ film-NN } of-PRP best-JJ {director-NN ∪ producer-NN ∪ artist-NN ∪ academy-NN} award-NN [for-PRP]*, to match questions with phrases

*Recommend me a movie which got academy award for best director*
*Cannes Film Festival Best director award movie*
*Give me a movie with National Film Award for Best Producer*
*Academy award for best picture*
*Movies of greatest film directors of all time*

Totally 235 templates were added, 10-20 per each entertainment category or genre.

Table 9: Evaluation of search

| Query | phrase sub-type | Relevancy of baseline Yahoo search, %, averaging over 20 searches | Relevancy of baseline Bing search, %, averaging over 20 searches | Relevancy of re-sorting by SG, %, averaging over 20 searches | Relevancy of re-sorting by using taxonomy, %, averaging over 20 searches | Relevancy of re-sorting by TK, %, averaging over 20 searches | Relevancy of re-sorting by using taxonomy and SG, %, averaging over 20 searches | Relevancy improvement for hybrid approach, comp. to baseline, % |
|---|---|---|---|---|---|---|---|---|
| 3-4 word phrases | noun phrase | 86.7 | 85.4 | 87.1 | 93.5 | 88.0 | 93.6 | 8.8 |
| | verb phrase | 83.4 | 82.9 | 79.9 | 92.1 | 79.3 | 92.8 | 11.6 |
| | how-to expression | 76.7 | 78.2 | 79.5 | 93.4 | 78.8 | 93.3 | 12.0 |
| | average | 82.3 | 82.2 | 82.2 | 93.0 | 82.0 | 93.2 | 11.3 |
| 5-10 word phrases | noun phrase | 84.1 | 84.9 | 87.3 | 91.7 | 88.3 | 92.1 | 9.0 |
| | verb phrase | 83.5 | 82.7 | 86.1 | 92.4 | 86.9 | 93.4 | 11.2 |
| | how-to expression | 82.0 | 82.9 | 82.1 | 88.9 | 81.6 | 91.6 | 11.1 |
| | average | 83.2 | 83.5 | 85.2 | 91.0 | 85.6 | 92.4 | 10.8 |
| 2-3 sentences | one verb one noun phrases | 68.8 | 67.6 | 69.1 | 81.2 | 80.8 | 83.1 | 12.2 |
| | both verb phrases | 66.3 | 67.1 | 71.2 | 77.4 | 78.4 | 78.3 | 11.7 |
| | one sent of how-to type | 66.1 | 68.3 | 73.2 | 79.2 | 79.9 | 80.9 | 12.0 |
| | average | 67.1 | 67.7 | 71.2 | 79.3 | 79.7 | 80.8 | 11.9 |

We observe that using SG only improves the relevance of search in cases where query is relatively complex. For shorter sentences there is just a slight improvement in accuracy, for medium-length queries of 5-10 keywords we get <2% improvement, and <5% improvement for multi-sentence query. As the absolute performance of search naturally drops when queries become more complex, relative contribution of syntactic generalization increases. TK outperformed the stand-alone SG by 4% but is well below the hybrid SG+taxonomy search accuracy.

Notice that in a vertical domain of eBay entertainment where the taxonomy coverage is good (most questions are mapped well into taxonomy), SG usually improves the relevance on its own, and as a part of hybrid system. However there are cases with no improvement. Taxonomy-based method is always helpful in a vertical domain, especially for a short queries (where most keywords are represented in the taxonomy) and multi-sentence queries (where the taxonomy helps to find the important keywords for matching with a question). We can conclude for a vertical domain that a taxonomy should be definitely applied, and SG possibly applied, for improvement of relevance for all kinds of questions. Relevance of the hybrid system is improved by about 15%.

## 7 Related Work and Discussions

From the semantic standpoint, SG can be viewed as semantic inference for classification. Most work in automated semantic inference from syntax deals with much lower semantic level then semantic classes we manage in this work. [17] presents a principled, integrated approach to *semantic entailment*. The authors developed an expressive knowledge representation that provides a hierarchical encoding of structural, relational and semantic properties of the text and populated it using a variety of machine learning based tools. An inferential mechanism over a knowledge representation that supports both abstractions and several levels of representations allowed them to begin to address important issues in abstracting over the variability in natural language. Certain reasoning patterns from this work are implicitly implemented by parsing tree matching approach proposed in the current study.

Usually, classical approaches to semantic inference rely on complex logical representations. However, practical applications usually adopt shallower lexical or lexical-syntactic representations, but lack a principled inference framework. [8] proposed a generic semantic inference framework that operates directly on syntactic trees. New trees are inferred by applying entailment rules, which provide a unified representation for varying types of inferences. Rules are generated by manual and automatic methods, covering generic linguistic structures as well as specific lexical-based inferences. The current work deals with syntactic tree transformation in the graph learning framework (compare with [14, 36]), treating various phrasings for the same meaning in a more unified and automated manner.

The special issue of Information Sciences Journal on Intelligent knowledge-based models and methodologies for complex information systems [16] provides a good comparison framework for this study. Generalized association rule extraction is a powerful tool to discover a high level view of the interesting patterns hidden in the analyzed data [7]. However, since the patterns are extracted at any level of abstraction, the mined rule set may be too large to be effectively exploited in the decision making process. Thus, to discover valuable and interesting knowledge a post-processing step such as additional generalization control of rules is usually required [25]. In case of associated rules for text in the form of trees with nodes for words and/or POS, SG is capable of maintaining the proper generality of the extracted association rules, and can be viewed as an alternative approach.

SemEval Conference is an adequate forum to compare text similarity approaches. (Gomez et al 2015) describes the approach for the Community Question

Answering Task, which was presented at the SemEval 2015. The transforms the answers of the training set into a graph based representation for each answer class, which contains lexical, morphological, and syntactic features. The answers in the test set are also transformed into the graph based representation individually. After this, different paths are traversed in the training and test sets in order to find relevant features of the graphs. As a result of this procedure, the system constructs several vectors of features: one for each traversed graph. (Zarrella et al 2015) explored mixtures of string matching metrics for similarity measures, alignments using tweet-specific distributed word representations, recurrent neural networks for modeling similarity with those alignments, and distance measurements on pooled latent semantic features. Logistic regression was applied to integrate these component into the hybrid architecture. For twitter data, which is much noisier and higher variability than the text for the pipeline in this paper, the authors achieved F-measure of 71.6 which is significantly lower of F-measure of above 90% achieved in the current study for search. (Phuoc An Vo et al 2015) submitted three runs for SemEval 2015, Task #2 "Semantic Textual Similarity", English subtask, combining typical features (lexical similarity, string similarity, word n-grams, etc) with syntactic structure features, outperforming the best system of the 2014 dataset.

Table 10 helps to compare our results of the subtasks of the sentiment analysis tasks with those of SemEval. Whereas sentiments are extracted from all texts (including tweets) in the major evaluation tasks, for the industrial applications it is essential to only show *meaningful* sentences for the purpose of recommendation, and tailor them to the *knowledge state* of a user receiving recommendation.

| Experiment | Twitter | | | SMS | LiveJournal |
|---|---|---|---|---|---|
| | Dev | Test | Sarcasm | | |
| Majority | 29.19 | 34.64 | 27.73 | 19.03 | 27.21 |
| Phrase-Based System | 62.09 | 64.74 | 40.75 | 56.86 | 62.22 |
| Tweet-Level System | 62.4 | 63.73 | 42.41 | 60.54 | 69.44 |
| Combined System | 64.6 | 65.42 | 40.02 | 59.84 | 68.79 |

Table 10: A comparison between the different systems using the Twitter training corpus provided by the SemEval Sentiment Analysis task (Rosenthal et al 2014)

Traditionally, semantic parsers are constructed manually, or are based on manually constructed semantic ontologies, but these are is too delicate and costly. A number of supervised learning approaches to building formal semantic representation have been proposed (Mooney, 2007). Unsupervised approaches have been proposed as well, however they applied to shallow semantic tasks (e.g., paraphrasing [39], information extraction [6], and semantic parsing [19]. The problem domain in the current study required much deeper handling syntactic peculiarities to perform classification into semantic classes. In terms of learning, our approach is closer in merits to unsupervised learning of complete formal semantic representation. Compared to semantic role labeling [13] and other forms of shallow semantic processing, our approach maps text to formal meaning representations, obtained via generalization.

Statistical learning has been applied to syntactic parse trees as well. Statistical approaches are generally based on stochastic models [55]. Given a model and an observed word sequence, semantic parsing can be viewed as a pattern recognition problem and statistical decoding can be used to find the most likely semantic representation.

For a few decades, most approaches to NL semantics relied on mapping to First Order Logic representations with a general prover and without using acquired rich knowledge sources. A concise Frege-type graph language a tool for logic programming is presented in (Redey 1993), implementing the principle that the logical structure of natural language statements can be constructed in a language equivalent of first-order logic established solely via the basic natural grammatical relations.

Significant development in NLP, specifically the ability to acquire knowledge and induce some level of abstract representation is expected to support more sophisticated and robust approaches. A number of recent approaches are based on shallow representations of the text that capture lexico-syntactic relations based on dependency structures and are mostly built from grammatical functions extending keyword matching [20]. Such semantic information as WordNet's lexical chains [40] can slightly enrich the representation. Learning various logic representations [52] is reported to improve accuracy as well. (de Salvo Braz et al 2003) makes global use of a large number of resources and attempts to develop a flexible, hierarchical representation and an inference algorithm for it. (Yang 2012) combines techniques such as finite state machines, Part-Of-Speech tags, conceptual graphs, domain ontology and dependency tree to convert a patent claim into a formally defined conceptual graph. However, we believe neither of these approaches reaches the high semantic level required for practical application.

Web-based metrics that compute the semantic similarity between words or terms [34] are complementary to our measure of similarity. The fundamental assumption is used that similarity of context implies similarity of meaning, relevant web documents are downloaded via a web search engine and the contextual information of words of interest is compared (context-based similarity metrics). It is shown that context-based similarity metrics significantly outperform co-occurrence based metrics, in terms of correlation with human judgment.

A number of methods measure text similarity numerically, similarly to SG score. [54] proposed a method to collect short text snippets to measure the similarity between pairs of snippets. The method takes account of both the semantic and statistical information within the short text snippets, and consists of three steps. Given a set of raw short text snippets, it first establishes the initial similarity between words by using a lexical database. The method then iteratively calculates both word similarity and short text similarity. Finally, a proximity matrix is constructed based on word similarity and used to convert the raw text snippets into vectors. Word similarity and text clustering experiments show that the proposed short text modeling method improves the performance of IR systems.

Proposed approach is tolerant to errors in parsing. For more complex sentences where parsing errors are likely, using OpenNLP, we select multiple versions of parsings and their estimated confidence levels (probabilities). Then we cross-match these versions and if parsings with lower confidence levels provide a higher match score, we select them.

It must be remarked that integrating argumentation and recommender systems is a recent research topic. DeLP has proven to be an efficient tool for achieving this integration, as exemplified in [15]. Other successful applications of DeLP involve news analysis [49], intelligent processing of web-based forms and intelligent robotic soccer [21], among many others.

DeLP has also been used for modeling ontology reasoning. Standard approaches to reasoning with description logics ontologies require them to be consistent. However, as ontologies are complex entities and sometimes built upon other imported ontologies, inconsistencies can arise. [32] presents δ-ontologies, a framework for reasoning with inconsistent ontologies, expressing them as defeasible logic programs. Given a query posed w.r.t. an inconsistent ontology, a dialectical analysis is performed on a DeLP program obtained from such ontology, where all arguments in favor and against the final answer of the query will be taken into account. The current paper presents an industrial system for handling a special case of ontology inconsistencies by using a different mechanism of mapping, but similar underlying logic of argumentation.

Defeasible reasoning is a rule-based approach for efficient reasoning with incomplete and inconsistent information. Such reasoning is, among others, useful for ontology integration, where conflicting information arises naturally; and for the modeling of business rules and policies, where rules with exceptions are often used. Category mapping is an example of such domain, where we need to systematically treat exceptions. [5] describes these scenarios with rules with exceptions in more detail, and reports on the implementation of a system for defeasible reasoning on the Web. The system (a) is syntactically compatible with RuleML; (b) features strict and defeasible rules and priorities; (c) is based on a translation to logic programming with declarative semantics; and (d) is flexible and adaptable to different intuitions within defeasible reasoning.

[42] proposed several kernel functions to model parse tree properties in kernel-based machines such as perceptrons or support vector machines. In this study, instead of tackling a high dimensional space of features formed from syntactic parse trees, we apply a more structural machine learning approach to learn syntactic parse trees themselves, measuring similarities via sub-parse trees and not distances in this space. The authors define different kinds of tree kernels as general approaches to feature engineering for semantic role labeling (SLR), and experiments with such kernels to investigate their contribution to individual stages of an SRL architecture both in isolation and in combination with other traditional manually coded features. The results for boundary recognition, classification, and re-ranking stages provide systematic evidence about the significant impact of tree kernels on the overall accuracy, especially when the amount of training data is small. Structure-based methods of this study can leverage limited amount of training cases too.

The accuracy of our structural machine learning approach is worth comparing with the other parse tree learning approach based on statistical learning of SVM. (Moschitti 2008) compares performances of bag-of-words kernel, syntactic parse trees and predicate argument structures kernel, as well as semantic role kernel and confirms the accuracy improves in this order and reaches F-measure of 68% on TREC dataset. Achieving comparable accuracies, kernel-based approach requires manual adjustment on one hand, however it does not provide similarity data in explicit form of common-sub-phrases. Structural machine learning methods are better suited for performance-critical production environments serving hundreds millions of users because it better fits modern software quality assurance methodologies on one hand and can be implemented based on standard search libraries like Lucene on the other hand. Logs with found commonality expressions are maintained and tracked which assures required performance as system evolves in time and text classification domains change.

## 8 Conclusions

In this report from industry we addressed an issue of relevance in a content pipeline for consumer Internet portal. Although a limitation of keyword-based approach for content collection, cleaning, aggregation and indexing is well understood, there is no plausible alternative with proved performance is currently available. A full-scale linguistic processing (which is thought to be non-scalable for search and recommendation on an industrial scale) turned out to be essential to provide relevance in a domain independent manner. We demonstrated that a wide range of content pipeline components can rely on matching of syntactic parse trees for sentences and phrases to maintain relevance. We formalized this matching according to the traditions of inductive learning to be a least general generalization of parse trees. We also conducted the comparative analysis of syntactic generalization and tree kernel approaches and selected the former as superior for industrial applications, based on our evaluation of their performances.

Unlike TK, the structural approach of SG shows all intermediate generalization results, which allows tracking of how class separation rules are built at each level (pair-wise generalization, generalization ^ sentence, generalization ^ generalization, (generalization ^ generalization) ^ generalization, etc.) Although SVM can handle a high number of features including noisy ones, it can hardly outperform the situations where selected features are meaningful. Among other disadvantages of SVM [51] are a lack of transparence of results: it is hard to represent the similarity as a simple parametric function, since the dimension of feature space is rather high. Also, SVM is subject to over-fitting when a number of training examples is low compared to the number of features; results are very sensitive to a choice of kernel. It is hard to adapt SVM to multi-class classification settings. Overall, a tree kernel approach can be thought as statistical AI, and proposed approach follows along the line of logical AI traditionally applied in linguistics two-three decades ago. The current study suggests that the latter one is more suitable for traditional software development methodology for industrial applications.

Parsing & chunking (conducted by OpenNLP) followed by SG are significantly slower than other operations in a content management system and comparable with operations like duplicate search. Verifying relevance, application of SG should be preceded by statistical keyword-based methods. In real time application components, such as search, we use conventional TF*IDF based approach (such as SOLR/Lucene, see also (Erenel et al 2012)) to find a set of candidate answers of up to 100 from millions of documents and then apply SG for each candidate. For off-line components, we use parallelized map/reduce jobs (Hadoop) to apply parsing and SG to large volumes of data. This approach allowed a successful combination of efficiency and relevance for serving more than 10 million unique site users monthly at datran.com/allvoices.com, zvents.com and eBay.com.

Our solution of the meaningful vs meaningless opinionated sentence problem demonstrates how a very weak semantic signal concealed in a syntactic structure of sentence can be leveraged. Obviously, using keyword-based rules for this problem does not look promising.

We observed that contribution of SG to classification tasks varies with the problem formulation, classification settings and domain. Whereas SM tool shows an insignificant contribution of SG, other classification tasks leverages SG noticeably due to the importance of weaker syntactic signals.

Proposed approach is tolerant to errors in parsing. For more complex sentences where parsing errors are likely, using OpenNLP [43], we select multiple versions of parsings and their estimated confidence levels (probabilities). Then we cross-match these versions and if parsings with lower confidence levels provide a higher match score, we select them.

Presented content pipeline can be applied to other languages besides English as long as search engine APIs and linguistic models are available for them. Web portals like http://www.become.com, http://www.become.co.jp/ used similar content pipeline to the one presented in this paper and supported German, French, Italian and other languages. Also, besides entertainment, similar content pipeline has been applied to a broad range of products and services at these web portals.

Using semantic information for query ranking has been proposed in [2,18]. However, we believe the current study is a pioneering one in deriving semantic information required for ranking from syntactic parse tree *directly*. In our further studies we plan to proceed from syntactic parse trees to higher semantic level and to explore other applications that would benefit from it.

The rough set theory can be used to measure similarity between texts as well. [35] presents a research on the construction of a new unsupervised model for learning a semantic similarity measure from text corpora. Two main components of the model are a semantic interpreter of texts and a similarity function whose properties are derived from data. The first one associates particular documents with concepts defined in a knowledge base corresponding to the topics covered by the corpus. It shifts the representation of a meaning of the texts from words that can be ambiguous to concepts with predefined semantics. With this new representation, the similarity function is derived from data using a modification of the dynamic rule-based similarity model, which is adjusted to the unsupervised case.

In evaluation of this study we demonstrated how Facebook users of various level of activity and sophistication benefits from personalization in a variety of degrees. Users with limited number of Facebook categories, or a high number of adjusted set of Facebook likes, leverages personalization the most.

In this paper we argue that the hardest personalization component is to map Facebook categories into ones of the system providing recommendation, given its content and set of products/services. We tackled this problem by introducing defeasibility relation between category mapping rules, which allowed for inconsistent rules to be defeated and retain only rules which deliver more relevant recommendations.

There are statistical approaches to category mapping for search such as [48] who presented an ANN-based ontology matching model for knowledge source discovery on the Semantic Web. With the emergence of the Semantic Web a high number of domain ontologies were developed, which varied not only in their structure. However a lack of an integrated view of all web nodes and the existence of heterogeneous domain ontologies drive new challenges in the discovery of knowledge resources which are relevant to a user's request. New approaches have recently appeared for developing web intelligence and helping users avoid irrelevant results on the web. In this study we used deterministic approach to category mapping which can be viewed in a broader framework of ontology mapping.

Aggregating likes from friends is another important area supporting personalization, where adequate treatment of product categories is the key. The value of enables retailers/service providers and buyers/users alike to utilize the influence of trusted friends and family within the shopping experience has been demonstrated. Similar to the presented system, retailers and manufacturers using (iGoDigital 2013) product recommendation platform, can leverage a consumers' social network to provide an added layer of personalization and relevancy within their shopping experience. Consumers benefit from immediate access to product recommendations and opinions from their social network as they research, browse, and complete purchases, adding relevance and authenticity.

Our evaluation demonstrates that using personalized instead of traditional recommendations, we significantly increase:

- Overall user satisfaction with recommendation system, because users have to deal much less with irrelevant recommendations
- The number of attended events, including ones requiring ticket purchase.

Hence personalized recommendation dramatically improves efficiency and effectiveness of the user decision process on which events to attend. The recommendation component presented in this paper is oriented to work with Facebook; however, other social network profiles can be handled in a similar manner, including international social profiles in various languages.

There are an open source SG and TK component available at /opennlp/sandbox/opennlp-similarity/ as a part of OpenNLP [43], a machine learning system for natural language processing. It can support content pipelines, providing such functionality as SG-supported web mining and speech recognition, assistance with creative writing, taxonomy building and search. It includes a request handler for SOLR which makes it linguistically enabled, so that a search engineer can apply it to her domain and easily observe if relevance is improved or not. The project has a detailed documentation including [25] and an extensive set of tests to quickly grasp its functionality and application areas.

In this study we described the content pipeline with a high rate and amount of incoming data on events, which cannot be handled by a conventional keyword-based computer system. To be able to combine efficient storage, processing and analysis requirement with desired relevance, a more efficient text processing technique is required, based on richer linguistic information. Furthermore, the data on events come in a wide spectrum of forms, including social networks and opinions, so that the text processing operation also needs to support not only linguistic generalization but also perform at the level of categories of entities.

We also demonstrated that once domain-independent efficient SG component is developed to tackle textual date, leveraging rich linguistic information available for learning of parse tree, *the same component* has been used for a wide number of distinct problems.

## Acknowledgements

I appreciate the help of our colleagues Carlos Chesñevar, Daniel Usikov, Josep Lluis de la Rosa, Alexey Neznanov, Boris Kovalerchuk, Dmitry Ilvovsky, Fedor Strok and Sergei O Kuznetsov with the discussion of various issues of this study.

## References

1. Abney, S. Parsing by Chunks, Principle-Based Parsing, Kluwer Academic Publishers, 1991, pp. 257-278.
2. Aleman-Meza, B., C. Halaschek, I. Arpinar, and A. Sheth, "A Context-Aware Semantic Association Ranking," Proc. First Int'l Workshop Semantic Web and Databases (SWDB '03), pp. 33-50, 2003.
3. Alsinet, T., Carlos Iván Chesñevar, Lluis Godo, Guillermo Ricardo Simari: A logic programming framework for possibilistic argumentation: Formalization and logical properties. Fuzzy Sets and Systems 159(10): 1208-1228 (2008)
4. Anderson, RM, Christian Herwarth Borgs, Jennifer Tour Chayes, Uriel Mordechai Feige, Abraham Flaxman, Adam Tauman Kalai, Seyed Vahab Mirrokni, Moshe Tennenholtz,
5. Antoniou, G., D. Billington, G. Governatori, and M. Maher (2001). Representation results for defeasible logic. ACM Transactions on Computational Logic, 2(2):255-287.
6. Banko, Michael, J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007 Open information extraction from the web. In Proceedingsof the Twentieth International Joint Conference on Artificial Intelligence, pages 2670–2676, Hyderabad, India. AAAI Press.
7. Baralis, E., Luca Cagliero, Tania Cerquitelli, Paolo Garza. Generalized association rule mining with constraints, Information Sciences, Volume 194, July.
8. Bar-Haim, R., Dagan, I., Greental, I. Shnarch, E. Semantic Inference at the Lexical-Syntactic Level AAAI-05.
9. Bordini, R. H.; Braubach, L. 2006. A survey of programming languages and platforms for multi-agent systems. In Informatica, 33–44.
10. Brzezinski, D. Jerzy Stefanowski: Accuracy Updated Ensemble for Data Streams with Concept Drift. In Proc. HAIS 2011, Springer Verlag Lecture Notes in Artificial Intelligence 6679, 2011,155-163.
11. Bunke, H. Graph-Based Tools for Data Mining and Machine Learning. Lecture Notes in Computer Science, 2003, Volume 2734/2003, 7-19.

12. Cardie, C., Mooney R.J. Machine Learning and Natural Language. Machine Learning 1(5) 1999.
13. Carreras X. and Luis Marquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In Proceedings of the Eighth Conference on Computational Natural Language Learning, pp 89–97, Boston, MA. ACL.
14. Chakrabarti, D. and C. Faloutsos, "Graph Mining: Laws, Generators, and Algorithms," ACM Computing Surveys, vol. 38, no. 1, 2006.
15. Chesñevar, C. A. Maguitman, M. P. González. "Empowering Recommendation Technologies Through Argumentation" In "Argumentation in Artificial Intelligence", I.Rahwan, G.Simari (eds.). Springer Verlag, 2009 (505 p., in press). ISBN 978-0-387-98196-3
16. Cuzzocrea, Alfredo Editorial: Intelligent knowledge-based models and methodologies for complex information systems. Information Sciences, Volume 194, July (2012)
17. de Salvo Braz, R., Girju, R., Punyakanok, V., Roth, D. and Sammons, M. 2005 An Inference Model for Semantic Entailment in Natural Language, Proc AAAI-05.
18. Ding, L., T. Finin, A. Joshi, R. Pan, R.S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: A Search andMetadata Engine for the Semantic Web," Proc. 13th ACM Int'l Conf. Information and Knowledge Management (CIKM '04), pp.652-659, 2004.
19. Domingos P. and Poon, H. Unsupervised Semantic Parsing, with, Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, 2009. Singapore: ACL.
20. Durme, B. V.; Huang, Y.; Kupsc, A.; and Nyberg, E. 2003. Towards light semantic processing for question answering. HLT Workshop on Text Meaning.
21. Ferretti, E., Marcelo Errecalde, Alejandro Javier García, Guillermo Ricardo Simari: An Application of Defeasible Logic Programming to Decision Making in a Robotic Environment. LPNMR 2007: 297-302.
22. Folino, G., C. Mastroianni, S. Mostaghim and J. Suzuki (eds), BADS 2011: Proceedings of the 3rd IEEE Internatinoal Workshop on Bio-Inspired and Self-* Algorithms for Distributed Systems, ACM Press, June 2011.
23. Galitsky, B. Dobrocsi, G., de la Rosa, J.L. and Kuznetsov SO: Using Generalization of Syntactic Parse Trees for Taxonomy Capture on the Web. ICCS 2011: 104-117.
24. Galitsky, B. and Josep Lluis de la Rosa. Concept-based learning of human behavior for customer relationship management. Special Issue on Information Engineering Applications Based on Lattices. Information Sciences. Volume 181, Issue 10, 15 May 2011, pp 2016-2035 (2011).
25. Galitsky, B. Machine learning of syntactic parse trees for search and classification of text. Engineering Application of AI (2012).
26. Galitsky, B. Natural Language Question Answering System: Technique of Semantic Headers. Advanced Knowledge International, Australia (2003).
27. Galitsky, B., Daniel Usikov, Sergei O. Kuznetsov: Parse Thicket Representations for Answering Multi-sentence questions. 20th International Conference on Conceptual Structures, ICCS 2013 (2013).

28. Galitsky, B., Gabor Dobrocsi, Josep Lluis de la Rosa, Inferring the semantic properties of sentences by mining syntactic parse trees. Data & Knowledge Engineering (2012).
29. Galitsky, B., MP González, CI Chesñevar. A novel approach for classifying customer complaints through graphs similarities in argumentative dialogue. Decision Support Systems, 46-3, 717-729 (2009).
30. Garcia, A., and Simari, G. 2004. Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Programming 4:95–138.
31. Gildea, D. 2003. Loosely tree-based alignment for machine translation. In Proceedings of the 41th Annual Conference of the Association for Computational Linguistics (ACL-03), pp. 80–87, Sapporo, Japan.
32. Gomez, Sergio Alejandro, Chesñevar, Carlos Ivan, Simari, Guillermo Ricardo. Reasoning with Inconsistent Ontologies through Argumentation. Applied Artificial Intelligence. Volume 24, Issue 1 & 2, pp 102 – 148 (2010).
33. Hunch. hunch.com (2013).
34. Iosif, E. and Potamianos, A., "Unsupervised Semantic Similarity Computation Between Terms Using Web Documents," IEEE Transactions on Knowledge and Data Engineering, 13, Oct. 2009.
35. Janusz, A., Dominik Ślęzak, Hung Son Nguyen. Unsupervised Similarity Learning from Textual Data. Fundamenta Informaticae Volume 119 Issue - 3, 319-336.
36. Kapoor, S and H. Ramesh, "Algorithms for Enumerating All Spanning Trees of Undirected and Weighted Graphs," SIAM J. Computing, vol. 24, pp. 247-265, 1995.
37. Kuncheva, L.I.: Classier ensembles for changing environments. In: Roli, F., Kittler,J., Windeatt, T. (eds.) Multiple Classifier Systems. LNCS, vol. 3077, pp. 1, .Springer, Heidelberg (2004).
38. Last.fm www.last.fm (2013)
39. Lin, D., and Pantel, P. 2001. DIRT: discovery of inference rules from text. In Proc. of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2001, 323–328.
40. Moldovan, D.; Clark, C.; Harabagiu, S.; and Maiorano, S. 2003. Cogex: A logic prover for question answering. In Proc. of HLTNAACL 2003.
41. Moreda, P. Navarro, B., Palomar, M., Corpus-based semantic role approach in information retrieval. Data & Knowledge Engineering 61 (2007) 467–483
42. Moschitti, A., Kernel Methods, Syntax and Semantics for Relational Text Categorization. In proceeding of ACM 17th Conference on Information and Knowledge Management (CIKM). Napa Valley, California, 2008.
43. OpenNLP (2016) http://incubator.apache.org/opennlp/documentation/manual/opennlp.htm
44. Perrin, P and Fred Petry, An information-theoretic based model for large-scale contextual text processing. Information Sciences, Volume 116, Issues 2–4, 1999, Pages 229–252.
45. Plotkin, GD A note on inductive generalization. In B. Meltzer and D. Michie, editors, Machine Intelligence, volume 5, pages 153-163. Elsevier North-Holland, New York, 1970.

46. Rahwan, I., and Amgoud, L. 2006. An argumentation based approach for practical reasoning. In International Joint Conference on Autonomous Agents and Multi Agent Systems, 347–354.
47. Robinson JA. A machine-oriented logic based on the resolution principle. Journal of the Association for Computing Machinery, 12:23-41, 1965
48. Rubiolo, M., M. L. Caliusco, G. Stegmayer, M. Coronel, M. Gareli Fabrizi, Knowledge discovery through ontology matching: An approach based on an Artificial Neural Network model. pp 107-119. Inf. Sci. 194 (July 2012).
49. Sagui, F., A. Maguitman, C. Chesñevar, G. Simari. "Modeling News Trust: A Defeasible Logic Programming Approach" Iberoamerican Journal of Artificial Intelligence, Vol.12, No.40, pp.63-72. Edited by AEPIA (Spanish Association of Artificial Intelligence), Madrid, Spain, 2009. ISSN 1137-3601.
50. Stevenson, M. and Greenwood MA. 2005. A semantic approach to IE pattern induction. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005), Ann Arbor, Michigan.
51. Suykens J.A.K., Horvath G., Basu S., Micchelli C., Vandewalle J. (Eds.), Advances in Learning Theory: Methods, Models and Applications, vol. 190 NATO-ASI Series III: Computer and Systems Sciences,IOS Press, 2003.
52. Thompson, C., Mooney, R., and Tang, L. 1997. Learning to parse NL database queries into logical form. In Workshop on Automata Induction, Grammatical Inference and Language Acquisition.
53. Varshavsky, R. Tennenholtz; Moshe; Peres; Yuval; Wilson; David B. Group Recommendations in Social Networks. US Patent App 20110270774, Microsoft 2010.
54. Wenyin, L., Xiaojun Quan, Min Feng, Bite Qiu, A short text modeling method combining semantic and statistical information. Information Sciences, Volume 180, Issue 20, 15 October 2010, Pages 4031–4041.
55. Zhang, M., Zhou GD, Aw, A. Exploring syntactic structured features over parse trees for relation extraction using kernel methods, Information Processing and Management: an International Journal V 44 , Issue 2 (March 2008) 687-701.
56. Levenshtein, Vladimir I. (February 1966). "Binary codes capable of correcting deletions, insertions, and reversals". Soviet Physics Doklady 10 (8): 707–710.
57. Jaccard, Paul (1912), "The distribution of the flora in the alpine zone", New Phytologist 11: 37–50.
58. Galitsky, B. Transfer learning of syntactic structures for building taxonomies for search engines. Engineering Applications of Artificial Intelligence. Volume 26, Issue 10, November 2013, Pages 2504–2515.
59. Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. Conference on Empirical Methods in Natural Language Processing (EMNLP 2013).
60. Galitsky, B. Transfer learning of syntactic structures for building taxonomies for search engines. Engineering Applications of Artificial Intelligence. Volume 26 Issue 10, November, 2013 2504-2515.
61. Galitsky, B. Learning parse structure of paragraphs and its applications in search. Eng. Appl. of AI 32: 160-184 (2014).

62. Helena Gomez, Darnes Vilariño, David Pinto and Grigori Sidorov. CIC-BUAPnlp: Graph-Based Approach for Answer Selection in Community Question Answering Task. SemEavl-2015 18-22.
63. MITRE: Seven Systems for Semantic Similarity in Tweets Guido Zarrella, John Henderson, Elizabeth M. Merkhofer and Laura Strickhart. SemEavl-2015 12-17.
64. Duyu Tang, Furu Wei, Bing Qin, Ting Liu and Ming Zhou. Coooolll: A Deep Learning System for Twitter Sentiment Classification. SemEval-2014 12-17 208-212.
65. Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. Technical Report. Stanford University. 2009.
66. Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In Nicoletta Calzolari, N (ed), LREC' 2010.
67. Sara Rosenthal, Alan Ritter, Preslav Nakov and Veselin Stoyanov SemEval-2014 Task 9: Sentiment Analysis in Twitter. SemEval-2014
68. Bron, Coen; Kerbosch, Joep Algorithm 457: finding all cliques of an undirected graph", *Commun. ACM (ACM)* 16 (9): 575–577, 1973.
69. Krippendorff, K. (2004). Reliability in content analysis: Some common misconceptions and recommendations. Human Communication Research, 30(3).
70. Cumby, C. and Roth D. On Kernel Methods for Relational Learning. ICML, pp. 107-14. 2003.
71. Kong, F. and Zhou G. Improving Tree Kernel-Based Event Pronoun Resolution with Competitive Information. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, 3 1814-19. 2011.
72. Ngoc Phuoc An Vo, Simone Magnolini, Octavian Popescu. 2015. FBK-HLT: A New Framework for Semantic Textual Similarity. In Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval-2015), NAACL-HLT 2015, At Denver, US.
73. Berna Altınel, Murat Can Ganiz, Banu Diri, A corpus-based semantic kernel for text classification by using meaning values of terms, Engineering Applications of Artificial Intelligence, Volume 43, August 2015, Pages 54-66.
74. Zafer Erenel, Hakan Altınçay, Nonlinear transformation of term frequencies for term weighting in text categorization, Engineering Applications of Artificial Intelligence, Volume 25, Issue 7, October 2012, Pages 1505-1514, ISSN 0952-1976, http://dx.doi.org/10.1016/j.engappai.2012.06.013.
75. Boris Galitsky, Dmitry Ilvovsky and Sergey O. Kuznetsov. 2015. Rhetoric Map of an Answer to Compound Queries. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 681–686.
76. Shih-Yao Yang, Von-Wun Soo, Extract conceptual graphs from plain texts in patent claims, Engineering Applications of Artificial Intelligence, Vol-

ume 25, Issue 4, June 2012, Pages 874-887, ISSN 0952-1976, http://dx.doi.org/10.1016/j.engappai.2011.11.006.

77. Gábor Rédey, Conformal text representation, Engineering Applications of Artificial Intelligence, Volume 6, Issue 1, 1993, Pages 65-71, ISSN 0952-1976, http://dx.doi.org/10.1016/0952-1976(93)90042-V.

78. Themistoklis Mavridis, Andreas L. Symeonidis, Semantic analysis of web documents for the generation of optimal content, Engineering Applications of Artificial Intelligence, Volume 35, October 2014, Pages 114-130, ISSN 0952-1976, http://dx.doi.org/10.1016/j.engappai.2014.06.008.