Chapter 9
A Content Management System for Chatbots

Abstract In this chapter we describe the industrial applications of our linguistic-based relevance technology for processing, classification and delivery of a stream of texts as data sources for chatbots. We present the content pipeline for eBay entertainment domain that employs this technology, and show that text processing relevance is the main bottleneck for its performance. A number of components of the chatbot content pipeline such as content mining, thesaurus formation, aggregation from multiple sources, validation, de-duplication, opinion mining and integrity enforcement 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, Chap. 5) 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 that takes phrase-level and sentence-level information into account. In the partial case of SG, 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.

9.1 Introduction

Building a relevant and efficient content management system (CMS) is a key to successful consumer application such as chatbot. 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

B. Galitsky, *Developing Enterprise Chatbots*,
https://doi.org/10.1007/978-3-030-04299-8_9

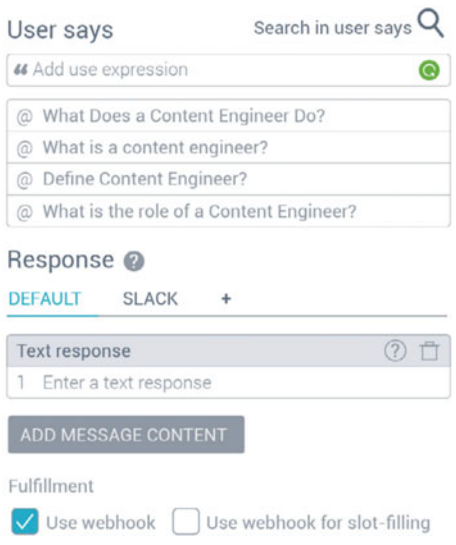
various data components, and generating search, recommendation and personalization results. Moreover, these platforms support search engine optimization, mobile application, share content with partner network, and other functionality. In this chapter that is written as a report from the field, we share the experience building a CMS for eBay.com entertainment, with the focus on text relevance. This CMS has also been providing entertainment content for indexing by Bing.com.

The main limitation of currently available CMSs such as WordPress, Atlassian, Kentico, Drupal, Hubspot and others is a quality, authenticity and integrity of content obtained from various sources including the web, text relevance analysis, relevance assessment to potential user intent and other sematic features. The other shortcoming is that these CMS are not well suited to handle conversational data for chatbots.

Content is a key to a chatbot acceptance by a user audience. Content quality is essential at many levels—ranging from FAQ pages to hard-coded chatbot dialogues to an extended, comprehensive content delivered to a user. Also, there is a unique set of restrictions and user needs associated with chatbot content. Nowadays, developers are using legacy tools and repurposing or retrofitting legacy content into chatbots, so in a lot of cases users are unimpressed. The challenge here is that in many cases the authors of chatbot content rely on yesterday's tools for building content for the modern, state-of-the-art interfaces. Properly designed chatbot content is not a plain structure but instead a well designed conversations (Tneogi 2018). Content that is designed for bots must have short and precise pieces of text that are quick to consume, emojis for meaningful, visual attention, and a high ratio of images and gifs in chatbot utterances.

For manual content creation for chatbots, one would need a wsyiwig editor. It is a system in which content, text and graphics, can be edited in a form closely resembling its appearance when printed or displayed as a finished product, such as a printed document, web page, or slide presentation. Such editor is designed for writing short pieces of content, internal emoji and image content-driven suggestions. The editor would need to be enabled with visualizing the content as it is created in various bot platform interfaces and a smart way to adapt it to different needs. Also, this CMS would need a speech-to-text engine built for the ability to support voice bot. Finally, such CMS would need a *text analysis and relevance tool at runtime* that helps authors ensure their content style is conversational. This is the focus of this chapter. Figure 9.1 shows a front end of a simple chatbot CMS that manages intent-response pairs of dialogflow.com. For a robust chatbot, a much more complex form of association between requests and responses is needed, combining search engineering (Chaps. 5, 6, 7, and 8) and dialogue management (Chaps. 10 and 11).

In a domains such as entertainment or finance, 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 do not get search, recommendation and personalization results they are expecting. 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 (Galitsky et al. 2011). In this chapter we demonstrate how a use of linguistic technology such as machine learning



@ What Does a Content Engineer Do?
@ What is a content engineer?
@ Define Content Engineer?
@ What is the role of a Content Engineer?

☑ Use webhook ☐ Use webhook for slot-filling

Fig. 9.1 A trivial CMS for a chatbot. Associations between intents and responses are maintained

syntactic parse trees for relevance assessment can benefit a CMS. To systematically learn linguistic structures, we rely the *syntactic generalization* (SG, Chap. 5) 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 (Galitsky 2014).

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. It should 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 chapter. 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 (Galitsky 2003). Solving classification problems is an important task of the content pipeline, and these problems need to be solved in the dynamically changing environments (Brzezinski and Stefanowski 2011; Kuncheva 2004), especially adapting to particular user interests. Required text mining technology needs to be expressive, sensitive and domain-independent at the same time (Sidorov 2014). Using keyword-level classifiers for our domain would lead to a huge classification dimension and possibly over-fitting, so we need a more accurate representation of linguistic data such as parse trees and machine learning means for them (Galitsky et al. 2012).

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 (2013) and Mahout (2013) are capable of parsing and learning a large amount of textual data, but the relevance and semantic features are left behind. We will evaluate how an implementation of machine learning of parse trees can improve a number of automated CMS 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 (Galitsky and de la Rosa 2011) 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 thesaurus for all kinds of entertainments, maintain and update it to achieve relevance (Galitsky 2013). 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 relevance-related 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.

9.1.1 From Search to Personalized Recommendations to Chatbots

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 the thesaurus 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 the 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, inter-personal trust and attempt to recognize user intentions (Varshavsky et al. 2010), a context-aware methodology of adjustment to prior knowledge about users is still to be developed.

While 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 nowadays. In this chapter we address the root problem of personalization quality, propose a solution which worked for Zvents.com of eBay, and evaluate it for a vertical recommendation domain.

9.2 Relevance-Related Problems in a Content-Management System

The content pipeline to be introduced was designed and implemented in eBay Entertainment and 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 to serve as an entertainment content provider. Although we evaluated relevance processing in the entertainment domain only, the expectations are that relevance can be supported in any chatbot domain by similar means.

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, as its accuracy goes up, the performance of the overall content portal improves.

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 and 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.

9.2.1 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. 9.2).

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.

The chatbot content pipeline includes the four following component (Fig. 9.2):

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 and Symeonidis 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 combines 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. Thesaurus of entities is built here to support search (Chap. 8 and Galitsky and Kovalerchuk 2014).

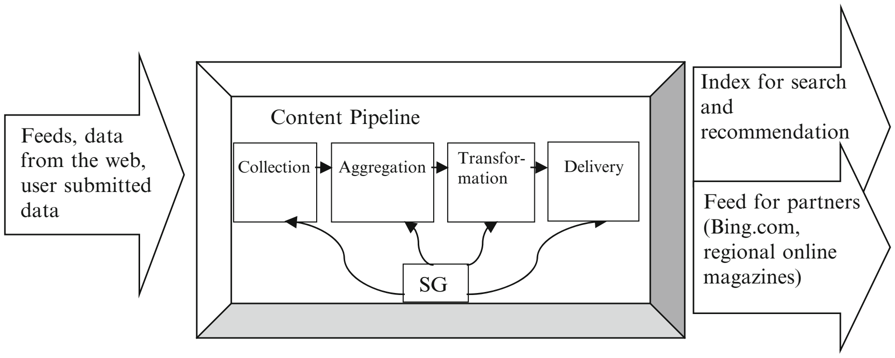

Feeds, data from the web, user submitted data
Content Pipeline
Collection
Aggregation
Transfor-mation
Delivery
SG
Index for search and recommendation
Feed for partners (Bing.com, regional online magazines)

Fig. 9.2 Input, output and the components of content pipeline and its relevance support by SG

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 content coming from the authors with low reputations and from the sources which are not reputable. The opinionated content should be separated from the 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 about this performer.

The third component, *Content Transformation* concerns building the index for search and recommendation. To do that, we need to take into account the strength of association of the information with the entities, the factual vs the opinionated content for an entity such as a performer (event data vs user feedback), entities position it time and their 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 users 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.

This chatbot CMS can be viewed as the one converting arbitrary unstructured content on the web to a well-structured, tagged, rated content (Galitsky and Levene 2007) 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, organized 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. 9.3). When 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 being identical. 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 thesaurus, inversion of content, building search index and other components are supported by other engines.

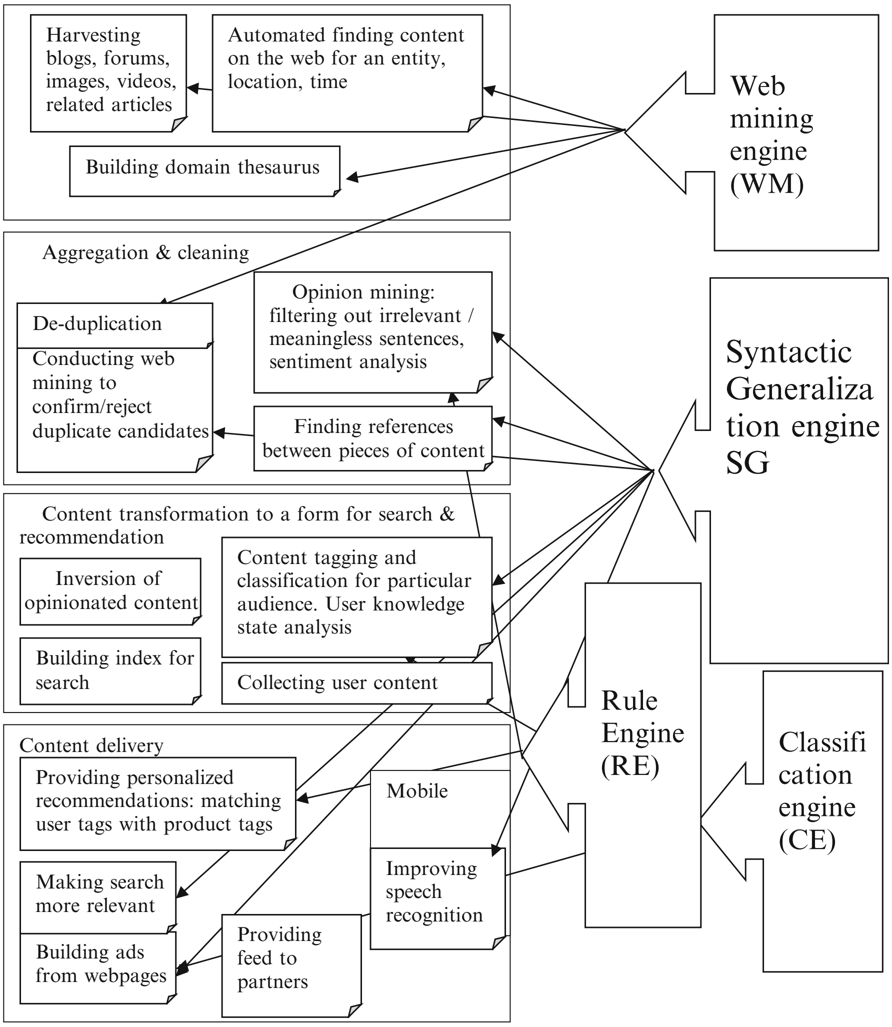

Harvesting blogs, forums, images, videos, related articles
Automated finding content on the web for an entity, location, time
Building domain thesaurus
Web mining engine (WM)
Aggregation & cleaning
De-duplication
Conducting web mining to confirm/reject duplicate candidates
Opinion mining: filtering out 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
Content tagging and classification for particular audience. User knowledge state analysis
Building index for search
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. 9.3 Detailed architecture of the content pipeline. The content processing components are on left and the engines are on the right

9.2.2 The Engines Assuring CMS Relevance

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.

9.2.3 Content Processing Units

9.2.3.1 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 do not 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 relying on 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 vs. 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.

9.2.3.2 Content Mining Unit

Content mining units rely 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 vs 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.

9.2.3.3 Thesaurus Unit

Thesauri are used to filter out irrelevant search results by verifying that important keywords from a query are also in a search result (Chap. 8). The thesaurus 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 thesaurus to filter out irrelevant answers, we search for a thesaurus 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. Example thesaurus related to CMS entities is shown in Fig. 9.4. More details are available in Chap. 8.

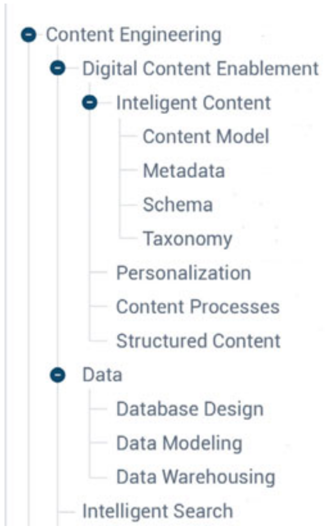


Fig. 9.4 CMS thesaurus

9.2.3.4 Opinion Mining Unit

One of 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, the opinion mining problem 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 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 happen to be 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, identifying irrelevant opinions, is rather hard one and requires a sensitive treatment of sentence structure, because a difference between a meaningful and a 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 in square brackets):

‘*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.

9.2.3.5 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 a 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 and compare how similar search results are. 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.

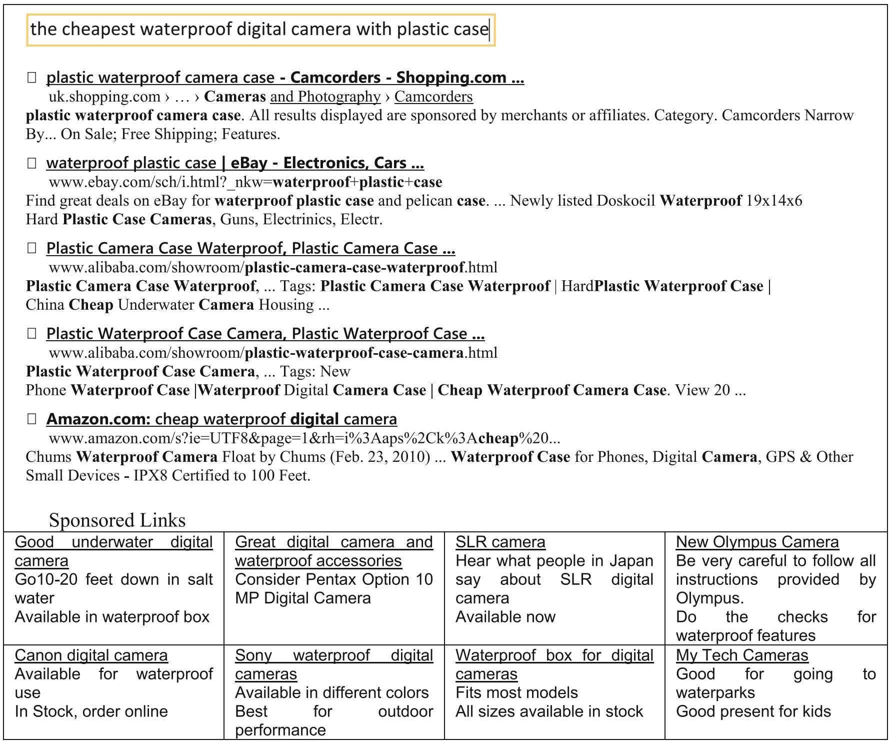
We also look up the entities being de-duped at various entertainment domain-specific sites and social network sites to make sure most possible phrasings to express each entity is covered. We compute what we call "similarity in web search space" among the search results for a pair of entities to reject/confirm 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 thesaurus 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 (Galitsky et al. 2011).

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.

9.2.3.6 Chatbot 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. 9.5).



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

Fig. 9.5 SEM unit for automated generation of ads from a webpage

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. 9.5). 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. 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 techniques implemented as parallel sub-units:

1. Syntactic information extraction problem is formulated as extracting and modifying a triple of expressions from a web pages such that these expressions:
   - serve as a page abstract, express its main topic, 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 product parameters.

2. Template-based approach; it finds a series of existing ad mined on the web, which are semantically close to the given webpage, and modifies them to form original ad 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 h

9.2.3.7 Speech Recognition Semantics Unit

This unit assures we have 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 chapter).

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.

9.2.3.8 Search Unit

SG engine improves search relevance by measuring similarity between query and sentences in search results (or their 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 in its phrase thesaurus because they are not necessarily sufficiently frequent. SG is essential when these phrases are interpretable only when they occur within a certain natural language expression. Further details are provided in Chap. 5 and Galitsky 2012.

9.2.3.9 Personalization Unit

Personalization of search results to take into account user interest and user profile is becoming a must feature in todays content delivery systems. In particular, a number of recommendation systems use social profile to tailor search results to the 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 at the category level (categories of user likes) for the wide range of interests. Since our recommendation engine is focused on the ‘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 section, 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.

9.3 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.

9.3.1 *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.

9.3.2 *Mapping Categories of Interest/Thesauri*

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} ∪ {subcategories 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 (Bordini and Braubach 2006; Rahwan and Amgoud 2006).

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).

9.3.3 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, García and Simari 2004) 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 $P = (\Pi, \Delta)$ 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 $P = (\Pi, \Delta)$ 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 $= (\Pi \cup A)$
2. the set $(\Pi \cup A)$ 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. $\Pi \cup \{h, h_1\}$ 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 (García and Simari 2004).

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, 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 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

Λ = [<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 [Λ, <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 (Table 9.1).

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 =*

Table 9.1 An example of a Defeasible Logic Program for modeling category mapping

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.

{(┐ *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. 9.6.

Having shown how to build dialectic tree, we are 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.

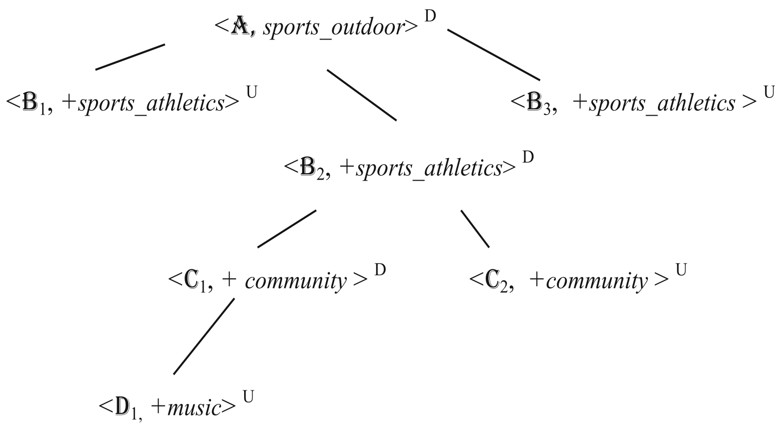


Fig. 9.6 Dialectical tree for category *sports_outdoor* using the DeLP Category Mapping (Table 9.1)

9.4 The Algorithms for High-Relevance CMS

In this section we provide the algorithm details in a more formal way. Thesaurus building, de-duplication, sentiment analysis, SEM information extraction and SG engine are the components where the algorithm implementation details are fairly important.

9.4.1 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 observe 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 and 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.

9.4.2 Analyzing Sentiments by Parse Tree Navigation

One of the most important and most difficult tasks in marketing is to estimate how consumers view various products. A simple example illustrates the problem to be solved. Let us consider an example of a supplier of LCD screens for personal digital assistants (PDAs), and we need to figure out what positive and negative impressions the public holds about your product. The available dataset includes 300,000 customer support reviews about an entire product line. The objective is to determine what aspects of this product line are viewed favorably or unfavorably, without trying to manually read all these reviews to understand the public sentiment.

When purchasing online, consumers are interested in researching the product or service they are looking to purchase. Currently, this means reading through reviews written on websites of different vendors that happen to offer the product or service. For example, if the consumer is interested in purchasing a digital camera, several on-line vendors allow consumers to post reviews of cameras on the website. Gathering information from such reviews is still a very time-consuming process as there is little way to sort the reviews for the features that are of interest to any one potential buyer so the potential buyer must read through them manually. Sometimes reviewers rate a product with a given number of stars in addition to making comments. An average high or low number of stars is not necessarily very informative to a potential buyer, especially if she is concerned about certain features on the camera. For example, a potential buyer may want a camera from which the photographs come out with very true colors as opposed to oversaturated colors. Other features, such as the weight of the camera or the complexity of the controls are of lesser concern to this potential buyer.

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

(continued)

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 (Galitsky et al. 2012) for more details). For a horizontal (unlimited) domain (like electronics, which is rather wide), all terms from need expressions cannot be defined via a thesaurus. 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.

The examples of extraction of two need expressions are shown at Fig. 9.7. 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 node is '*be*', and the need expression is branching from the vertex 'g*et*'.

Once the need expressions are extracted, they need to be normalized and grouped. Normalization transforms need expressions into sequences of words in the normal form, without prepositions and articles. After that, the 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. 9.8), 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 *viewfinder* from very different aspects. Notice that both syntactic relations between *viewfinde*r and second word vary, 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.

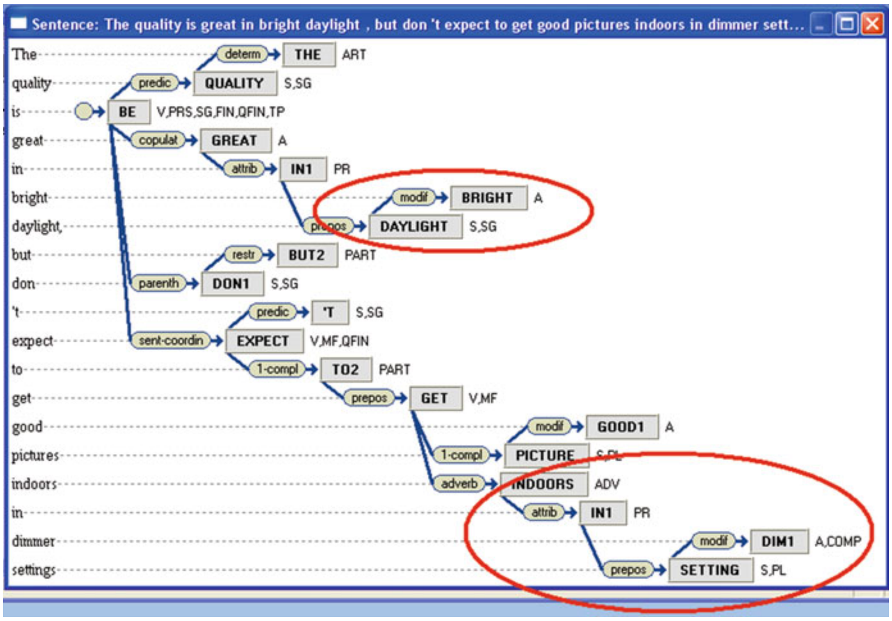


Fig. 9.7 Syntactic parse tree for sentences about digital camera with two pairs of sentiment-need expressions (circumscribed)

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 Viewfinders
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 resolution of 1280x1024 or 1.3 million pixels

Fig. 9.8 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 (Makhalova et al. 2015); 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 percentage of word pairs where the 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.

9.4.3 Agglomerative Clustering of Search Results

Search queries that express broad user intent frequently return fairly large result sets so that a user needs to navigate them. The idea of clustering search results into semantically similar groups in real time and presenting descriptive summaries of these groups to the user is almost three decades old. The clustering allows search user to identify useful subset of the results, when can in turn be clustered to identify narrower subsets (Tunkelang 2018). Clustering helps the user to quickly navigate to the relevant subset of the search results; this subset can in turn be clustered to navigate to even narrower subsets.

Clustering search results need to meet the following criteria to be usable. Firstly, each cluster should be associated with a meaning communicated with the user (by labels, snippets or individual search results indicative of this cluster). Secondly, search results of the same cluster should have a similarity with each other. Each cluster needs to be a coherent subset of possible search intents. Thirdly, search results assigned to different clusters should be substantially different from one another. Each cluster needs to contain a distinct subset of search intents.

The main difficulties in clustering search results are defining the similarity function, adjusting the clustering algorithm, and producing informative snippets for the obtained clusters. It is straight-forward to determine whether two search results are near-duplicates. However, determining that two results are semantically similar is a much harder problem, dependent on an appropriate similarity threshold that consistently derives systematic, distinct clusters.

A search result clustering algorithm needs to address the following issues (Leouski and Croft 1996):

1. Implement clustering as a classification of a document into a cluster. Documents can be treated as vectors of weight-term pairs. The system designer needs to decide on which terms to chose and whether to use the whole document or only a part of it as the source of terms;
2. Select the classification algorithm. The existing clustering techniques vary in accuracy, robustness, speed and storage requirements;
3. Engineer the output of the classifier, or cluster representations. The classification process results in a set of clusters, where every cluster contains documents about a

unique topic. Clusters can be represented using a selected document or term list, and more creativity with cluster representation is needed;

4. Determine the evaluation settings. After the classification tool is created, the results need to be analyzed and performance evaluated its from the effectiveness and efficiency viewpoint. The evaluation of how the effective are clustering results in comparison with the flat list is fairly difficult.

Typical clustering approaches involve embedding documents into a vectors and then computing a geometric function on them, such as cosine, to measuring their similarity. While such approaches have a solid theoretical foundation, the results are frequently random and illogical, highly subject to the peculiarities of the documents being clustered.

Hierarchical clustering algorithms are either top-down or bottom-up (Manning et al. 2008). The former class of algorithms tackles each document as a singleton cluster at the outset and then successively merge (or agglomerate) pairs of clusters until all clusters have been merged into a single cluster that contains all documents. Bottom-up hierarchical clustering is therefore called hierarchical agglomerative clustering. Top-down clustering requires a method for splitting a cluster, doing it recursively until individual documents are reached.

9.4.3.1 Description of GreedySearch Algorithm

The input of the algorithm is a user query q in NL and a subset of snippets A^*_{last} ranked by their relevance for the last successful refined query, each snippet $a \in A^*_{last}$ has a particular real-valued weight $w \in$ R. These weights are assigned to snippets by a search engine and reflect not only relevance to the query, but also might take into account the user's profile, item popularity, geo-location, his search history, etc. The input at the initial call is a user query q and the empty set of snippets A^*_{last}. We first present the search function *GreedySearch* followed by the clustering algorithm *AgglomerativeClustering* towards the end of this subsection.

At the first step (line 1) the request is sent to a search engine. Then, a function δ is applied to the set of returned snippets A and the request q in order to obtain their unique formal representations $\delta(q)$ and $A_\delta = \{\delta(a) \mid a \in A\}$, respectively. This representation makes texts comparable to each other.

GreedySearch algorithm, a strategy for specification of queries looks as follows:

```
Input: query $q$ in NL, snippet set for the last relevant refinement $A^*_{last}$
Output: ordered set of answers $A^*$ in natural language $A^* = GreedySearch(q, A^*_{last})$
```

```
1  $\mathcal{A} \leftarrow QuerySearchEngine(q)$
2  $(\delta(q), \mathcal{A}_\delta) \leftarrow ComputeFormalRepresentation(q, \mathcal{A})$
3  $S \leftarrow SimilarityMatrix(\mathcal{A}_\delta)$
4  $\mathcal{C} \leftarrow AgglomerativeClustering(\delta(q), \mathcal{A}_\delta)$
5  $found \leftarrow False$
6  $q_{aug} = \emptyset$
7  while not $found$ and $(\mathcal{C} \neq \{\emptyset\})$ do
8      $C \leftarrow TakeTheLargestCluster(\mathcal{C})$
9      $\mathcal{T} \leftarrow \delta^{-1}(ComputeDifference(C, \delta(q)))$
10     $r \leftarrow GetUserFeedback(\mathcal{T})$
11     if $r = showDetails$ then
12         $q_{aug} \leftarrow \emptyset$
13         $found \leftarrow True$
14     end
15     else if $r = relevant$ then
16         $q_{aug} \leftarrow q \cup \mathcal{T}$
17         $found \leftarrow True$
18     end
19     else
20         $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C\}$
21     end
22 end
23 if not $found$ then
24     $ranking \leftarrow sort(\{w_a \cdot \delta(a) \mid \delta(a) \in C\})$
25     $\mathcal{A}^* \leftarrow selectInGivenOrder(\mathcal{A}, ranking)$
26     if $q_{aug} \neq \emptyset$ then
27         $\mathcal{A}^* \leftarrow GreedySearch(q_{aug}, \mathcal{A}^*)$
28     end
29 end
30 else
31     $\mathcal{A}^* \leftarrow \mathcal{A}^*_{last}$
32 end
33 return $\mathcal{A}^*$
```

To compute clusters (line 4) of similar snippets we use two matrices: the matrix of syntactic similarity S and search relevance similarity matrix W with the entries

$s_{ij} = sim(\delta(a_i),\delta(a_j))$, $i,j = 1,\ldots,|A|$ and
$w_{ij} = rel\_sim(w_i,w_j)$, $i,j = 1,\ldots,|A|$, respectively.

We assume that the values of both similarity matrices are scaled to [0,1]. Centroids of the computed clusters $\mathcal{C}$ are the candidates for a new refined request. Specific information about the clusters is being presented to the user until a cluster with relevant specification is found (lines 7–22). The interaction with the user is carried out in 4 steps:

1. The biggest clusters C is chosen, i.e., $C = argmax_{C \in \mathcal{C}} |\{\delta(a) \mid \delta(a) \in C\}$ (line 8);
2. The added information in C w.r.t. q is computed. In can be done formally by computing the difference between a centroid of cluster C and $\delta(q)$ (see *ComputeDifference* function, line 9);

3. The computed difference is translated into a set of phrases $\mathcal{T}$;
4. $\mathcal{T}$ is shown to the user and feedback $r \in \{ShowDetails, Relevant, Irrelevant\}$ is received. The feedback defines the further strategy of the chatbot.

ShowDetails means that the user has found the information she searched for and all the snippets/documents corresponding to the cluster will be returned to the user ranked by their relevance weights (line 25) assigned by the search engine. *Relevant* answer is the case where the user has found a proposed query specification quite useful, but not enough (i.e., the further query specification is required), in this case a new augmented query q_{aug} is sent to the search engine (line 27) via the recursive call of *GreedySearch*(q_{aug}, A^*), *Irrelevant* answer describes the case where specifications do not contain relevant information. When all proposed specifications in $\mathcal{C}$ are irrelevant, the algorithm returns a subset of snippets from a cluster with the last relevant specification (line 31).

9.4.3.2 Agglomerative Clustering Algorithm

Agglomerative clustering is applied to the snippets to get short specifications of a request. The termination criteria ensures that each centroid of clusters (i.e., the shared information of snippets in a cluster) will be the shortest specification of the request. We denote a cluster by capital letter C and the corresponding centroid by lower case letter c. For the sake of convenience we define some functions that will be used in listing of the *AgglomerativeClustering* algorithm.

As mentioned above, requests and snippets are given in NL. We define a mapping δ: $L \rightarrow V$ that maps a text in natural language to a unique formal representation, L is a space of all possible texts in natural language, V is a space of their formal representations. Further we consider the examples of spaces V and discuss how the functions defined in this section can be rewritten for the considered spaces.

sim: $V \times V \rightarrow [0,1] \subset$ R is a function that evaluates similarity between two objects, the similarity between an object and its copy is equal to 1.

merge: $V \times V \rightarrow V$ is a function that returns a shared description of its two arguments, the shared description is in the same space as the merged arguments. *is_included*: $V \times V \rightarrow \{True, False\}$ is a function that returns *True* if the description of the first argument is included in the description of the second one, *False* otherwise.

rel_sim: $R \times R \rightarrow [0,1] \subset$ R is a function that evaluates relevance similarity between two objects by their relevance weights, the similarity between an object and its copy is equal to 1.

Input: query $\delta(q)$, snippet set A_δ

Output: set of subsets of snippets $\{A^* | A^* \subseteq A\}$ = *AgglomerativeClustering*($\delta(q)$, A_δ)

```
1  $\mathcal{C} \leftarrow \mathcal{A}_\delta$
2  $S \leftarrow SyntacticSimilarityMatrix(\mathcal{C})$
3  $W \leftarrow RelevanceSimilarityMatrix(\mathcal{C})$
4  $s_{ij} = min_{i,j=1,\dots,|S|}(k_1 S + k_2 W)$
5  $c = merge(c_i, c_j)$
6  $w_C = merge(c_i, c_j)$
7  while $is\_included(\delta(q), c)$ do
8      $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{c_j, c_j\}) \cup \{c\}$
9      $S \leftarrow SyntacticSimilarityMatrix(\mathcal{C})$
10     $W \leftarrow RelevanceSimilarityMatrix(\mathcal{C})$
11     $s_{ij} = min_{i,j=1,\dots,|S|}(k_1 S + k_2 W)$
12     $c = merge(c_i, c_j)$
13 end
14 return $\{\mathcal{A}^* \mid \mathcal{A}^* \subseteq \mathcal{A}, \delta(a) \in C, C \in \mathcal{C}\}$
```

Agglomerative clustering receives a query $\delta(q)$ and a snippet set A_δ as input, represented in the space where *sim*, *merge* and *is_included* functions are defined. Initially, each snippet $a \in A_\delta$ is an individual cluster centroid in $\mathcal{C}$. Pairwise syntactic similarity between cluster centroids is stored in a matrix S of the size $| \mathcal{C} | \times | \mathcal{C} |$, the relevance similarity is stored in matrix W of the same size $| \mathcal{C} | \times | \mathcal{C} |$. On each iteration, the most similar cluster centroids are chosen (line 11) to compute a new centroid c, which is their shared description (line 12). The weight of a new cluster C is the maximal relevance weight of its members, i.e., $w_C = \max\{w_a \mid \delta(a) \in C\}$. Here we use capital letters for clusters and lowercase letters for their centroids, i.e. $C \subseteq A_\delta$ for a cluster and c for its centroid.

To compute similarity between centroids, both syntactic and relevant similarities are taken into account. We use a weighted average of the similarities, i.e., similarity between centroids c_i and c_j is defined as $k_1 s_{ij} + k_2 w_{ij}$, where $k_1, k_2 \in$ R are coefficients of importance of syntactic and relevance similarities, respectively. If a newly created centroid contains the description of the original query (i.e., it retains complete information about the query) the two merged centroids are replaced by their shared description, the weight of the cluster is the maximal weight of the members of the merged clusters, i.e., $w_C = \max\{w_a \mid \delta(a) \in C_i \cup C_j\}$. When all the centroids that do not lose the information from the original query are computed (the centroids that includes as much snippets as possible and retain information from the query), the subsets of snippets corresponding to the computed centroids are returned.

9.4.3.3 Similarity Used by Clustering

Vector Space Model Let us consider the simplest model of text representation. Once the snippets are received, a new set of terms from $\mathcal{A} \cup \{q\}$ is computed. The N found terms correspond to the vector entries. Each text is represented by a vector of size N and filled with 0 s and 1 s. The "1" at i means that the ith term is contained in the text.

1. $merge(d_1,d_2) = d_1 \cdot d_2$
2. $sim(d_1,d_2)$:
   (a) $sim(d_1,d_2) = JaccardSimilarity(d_1,d_2)$
   (b) $sim(d_1,d_2) = CosineSimilarity(d_1,d_2)$
   (c) $sim(d_1,d_2) = SimpleMatchingCoefficent(d_1,d_2)$
3. $is\_included(d_1,d_2) = d_1 \subseteq d_2 \equiv merge(d_1,d_2) = d_1$

The following similarity measure is based on Parse Thickets (Chap. 7)

1. $merge(d_1,d_2) = d_1 \sqcap d_2$
2. $sim(d_1,d_2)$:
   (a) $sim^{max}(d_1, d_2) := max_{chunk \in (d_1 \sqcap d_2)} Score(chunk)$
   (b) $sim^{avg}(d_1, d_2) := \frac{1}{|(d_1 \sqcap d_2)|} \sum_{chunk \in (d_1 \sqcap d_2)} Score(chunk)$
3. $is\_included(d1,d2) = d1 \sqsubseteq d2$
   (a) Relevance Similarity

$$rel\_sim(w_i, w_j) = 1 - \frac{|w_i - w_j|}{\max_{i,j \in 1,\ldots,|A|} w_{ij}}$$

We have observed that search results clustering is a well-established approach for presenting and organizing search result sets, and the search engineering community have continued to work on improving it. Clustering sounds great in theory but in practice turns out to be not always effective, logical and crisp. Overall, clustering is a valuable tool to respond to the queries with broad intent scope.

9.5 Building Conclusive Answers

9.5.1 Concluding a Question Answering Session

In this section we focus on the issue of how to conclude a chatbot session in a comprehensive manner, to satisfy a user with detailed extended answer. For a question-answering session, the goal is to enable a user with thorough knowledge related to her initial question, from a simple fact to a comprehensive explanation. Sometimes, a short and concise answer such as account balance or person name suffices. However, frequently, a longer answer including multimedia content compiled from multiple sources is most appropriate. This answer is expected to be a comprehensive source of information on a topic, including definitions, recipes and explanations. In this section we focus on the algorithm of forming such answers.

After an initial question of a user, a number of clarification steps usually follow. Then, once the chatbot collected all necessary information, it can decide on what kind of answer is most suitable for a given session. For a factoid question, a brief specification of the value of the parameters or attributes in question is delivered.

Otherwise, a final answer about an entity mentioned in question, such as introduction of a bank account or a rule for how to close it, is issued.

Traditional chatbots do not possess this feature. Deterministic chatbots instead provide short replies by texts indexed in a typical search index. Statistical learning and especially deep learning – based chatbots attempt to use learning to tailor its answers to user session, but only brief texts can be obtained as a result. Even if they are meaningful, texts obtained as a result of such learning are short and not comprehensive. In a machine learning environment, typically each reply is obtained as a result of learning, and no special attention is given to a concluding answer.

These are the requirements for the complete, comprehensive answer that gives a user a chance to get a deep understanding of an entity/topic and a good way to conclude a dialogue:

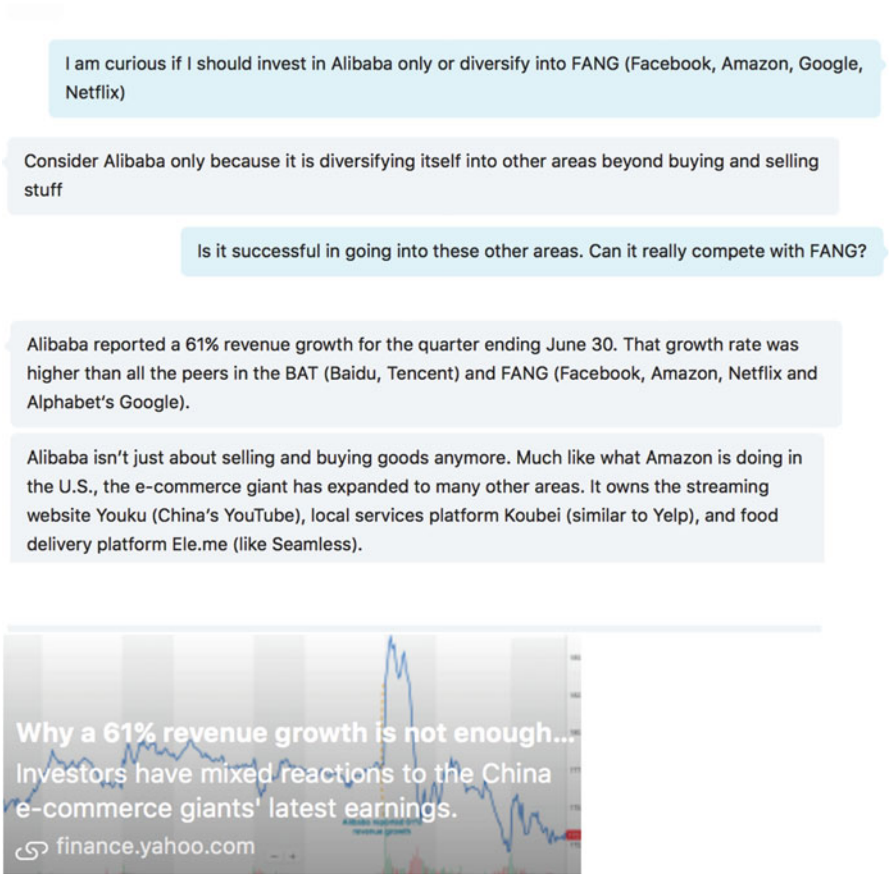
1. An answer has to be compiled from multiple sources to assure an unbiased, objective description. If it is opinionated, multiple opinions from a broad spectrum of perspectives must be compiled in a coherent manner. This dialogue conclusive answer has to be comprehensive to provide sufficient information for a user to be satisfied with a chatbot session. If the further questions based on this answer arise, the user can start a new chatbot session keeping in mind a specific focus;
2. An answer should be as easy to perceive and as intuitive as possible. Therefore combination of images, videos and audio files is beneficial. The answer compilation method should be domain – independent and adhere to certain presentation standards;
3. An answer should have a table of content and references, if it spans over multiple pages.

An example of a conclusive answer for a brief dialogue is shown in Fig. 9.9. The user requested a recommendation about investment, received it and expressed her doubts. The chatbot provides the comprehensive conclusive answer entitled '*Why a 61% revenue growth is not enough for Alibaba'* with detailed information on competitiveness including the charts. In this section we explore technologies for automated building of such answers.

One of the essential problem to be solved building a conclusive answer is to form its logical structure from the logic of a dialogue (Galitsky and Ilvovsky 2017) and user interests, intentions and possible misunderstanding expressed in it.

9.5.2 Building a Structure of Conclusive Answer

An answer flow should reflect the structure of preceding dialogue, if it is available and non-trivial. Also, if some controversial intermediate questions and/or answers lead to it, they should be reflected in the structure of this conclusive answer. This answer should constitute a document with section structure reflecting either the generally accepted sequence of topics for this type of entity (such as a biography for a person or account usage rules) or the logical flow of a dialogue occurred so far



I am curious if I should invest in Alibaba only or diversify into FANG (Facebook, Amazon, Google, Netflix)

Consider Alibaba only because it is diversifying itself into other areas beyond buying and selling stuff

Is it successful in going into these other areas. Can it really compete with FANG?

Alibaba reported a 61% revenue growth for the quarter ending June 30. That growth rate was higher than all the peers in the BAT (Baidu, Tencent) and FANG (Facebook, Amazon, Netflix and Alphabet's Google).

Alibaba isn't just about selling and buying goods anymore. Much like what Amazon is doing in the U.S., the e-commerce giant has expanded to many other areas. It owns the streaming website Youku (China's YouTube), local services platform Koubei (similar to Yelp), and food delivery platform Ele.me (like Seamless).

Why a 61% revenue growth is not enough...
Investors have mixed reactions to the China e-commerce giants' latest earnings.
finance.yahoo.com

Fig. 9.9 A conclusive answer to a client having a doubt concerning investment recommendation by a chatbot

(such as why first attribute, the value of the second attribute, and why the value is such and such for the third attribute). An example for the latter case would be a certain credit card, its overdraft fee amounts, reasons the card can be cancelled and possibilities for interest rate increases.

For most basic questions like product features such documents are available and do not need to be constructed. However, for a broad number of topics and issues, when a user interest is beyond the definition and rules, selecting an existing pre-written document is insufficient and a specific one tailored to demands of a given user needs to be constructed.

Hence there are two kinds of sources/options for building a document structure, or its table of content (TOC):

1. If a user does not indicate in a dialogue a preference for a specific issues associated with entity, a default structure is provided. It can be mined from the

general web sources such as Wikipedia and domain-specific sources such as Investopedia.com. For example, the TOC for the topic *Adjusted Gross Margin* would use the section structure from the respective Investopedia page https://www.investopedia.com/terms/a/adjusted-gross-margin.asp such as the main definitions, treatment in depth, associated topics and others. In this case it is possible to build TOC in a hierarchical manner.

2. If a user has a specific concern about an entity, such as '*Why banks can increase APR without advance notice',* then the TOC is built from multiple documents' section titles. These documents are identified on the web or intranet to be relevant not just to the main entity but also to the *Why* part. A document can start with a section on APR but then proceed to various cases on how banks increased the APRs and associated issues.

We use a high-level discourse structure of human-authored text to automatically build a domain-dependent template for given topic, such as event description, biography, political news, chat and blog. In case of a dialogue or a text containing some kind of argumentative structure, this template is based on a sequence of communicative actions. In a general case we follow a certain epistemic structure extracted from multiple texts in a particular domain (for example, for a music event we present a performer biography, previous concerts, previous partnerships, and future plans).

Let us consider the following dialogue and its conclusive answer (Table 9.2).

The reader can see that this dialogue leads to Option 2 rather than to Option 1, since the user is frustrated about the NSF and is trying to understand why it happened and how to avoid it. A generic answer about an entity would probably upset this chatbot user further since he believes he knows general stuff about NSF. Therefore the conclusive answer should focus on a specific user issue/misunderstanding exposed in the previous utterances of a dialogue.

Table 9.2 Two options for dialogue flow

C(customer): Why was I charged a Non-sufficient fund fee (NSF)?	
Bank: Paying out of your account, you made your balance negative at some point	
C: But I first made a deposit and then made a payment for a lower amount	
Bank: Your deposit might not has been processed by the time you made your payment	
C: How can I stay positive on my account balance?	
Bank (with conclusive answer):	
Option 1: Generic Answer about an entity	Option 2: Answer specifically addressing customer concern
Non-sufficient Fund Fee (NSF)	***Non-sufficient Fund Fee (NSF): making sure your balance is positive***
Definition: Non-sufficient Fund Fee is a fee charged by the bank…	*Check deposits…*
Amount…	*Processing time…*
Ways to avoid…	*Inter-bank transactions…*
Why banks charge NSF…	*Link accounts…*

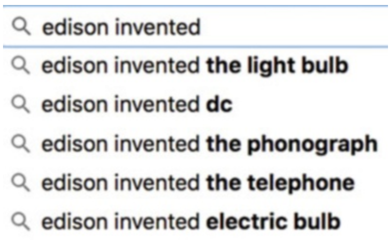

edison invented
edison invented **the light bulb**
edison invented **dc**
edison invented **the phonograph**
edison invented **the telephone**
edison invented **electric bulb**

Fig. 9.10 Auto-complete feature to discover the attributes of entities to build section structure

To form a TOC from the above dialogue, the following phrases from user utterances need to be used as queries to establish the section structure of the conclusive answer:

1. *Non-sufficient fund fee (NSF)*
2. Why was I charged
3. *Make a deposit*
4. *Make a payment for a lower amount*

These phrases (extended with synonyms) should match some section structures of certain documents about NSF and banking customer support logs: they will form a skeleton of the resultant answer.

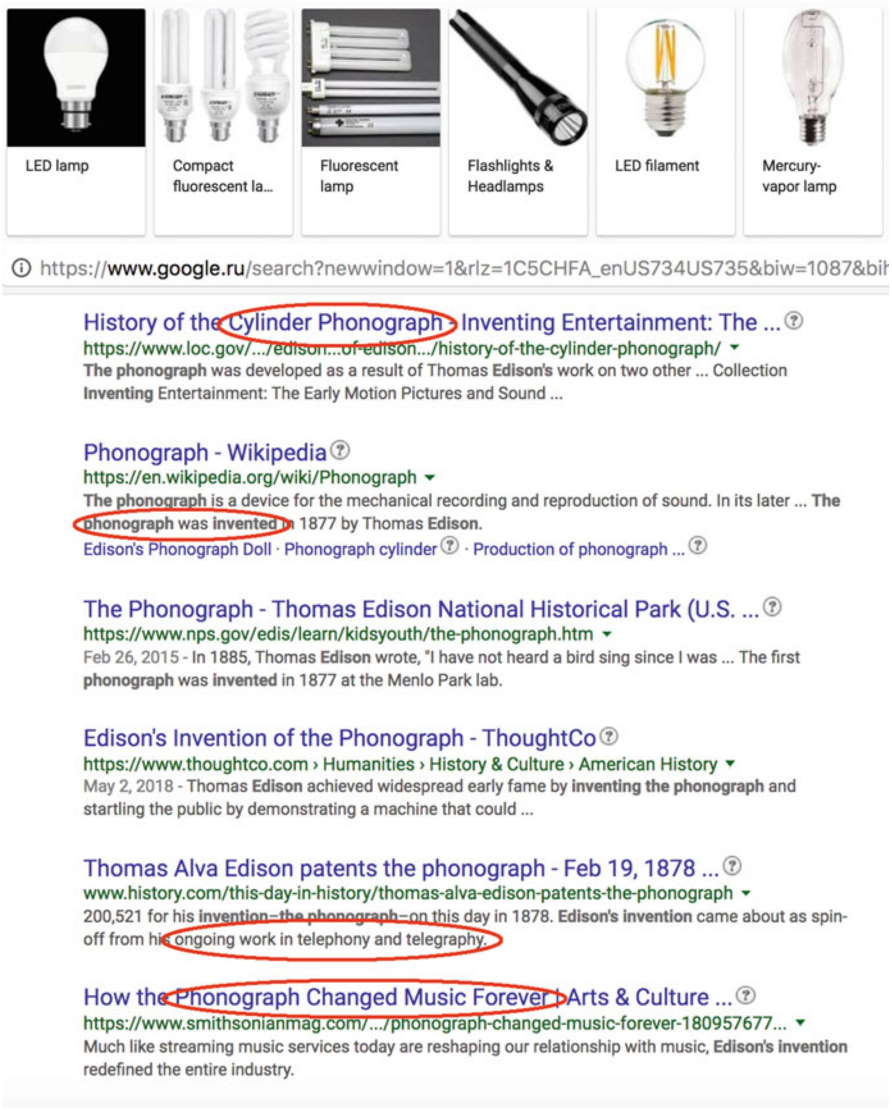
A good way to discover attributes for entities to form a structure of a document is an auto-complete feature for web search. If an entity in the preceding dialogue is '*Edison invented'* then the final concluding document can have the following TOC (Fig. 9.10). These auto-complete results (Google 2018) are the queries to the document index on the one hand and the section titles on the other hand.

To build a hierarchical TOC, we form search queries as entity (document title) plus the discovered section title: {'*Edison invented the light bulb', 'Edison invented the phonograph'*, …}.

For the first query, we visualize the types of light bulbs (following Google search) which can form subsections of the section 'Light bulbs' (Fig. 9.11 on the top). For the second query, we obtain the search results and attempt to extract noun phrases sound as section titles (on the bottom). Such noun phrases should include two-three modifiers (three-four words total) and do not include very specific infrequent words, non-English words and non-alphanumeric tokens.

The infrastructure for preparing content for building answers is shown in Fig. 9.12. Various available sources are used, including the written documents and web pages explaining entities, their attributes and specifying business rules. Case-based information can be available in the form of customer support logs, various forms of corresponding with customers or internal issue logs. All these sources with diverse structures need to be converted into a unified form which adheres to the following:

- A chunk of text needs to contain a paragraph-size text: two to six sentences, 60–150 words;
- This chunk of text should be self-contained; it should neither start with a reference to a previous paragraph nor end with a reference to a following one.



https://www.google.ru/search?newwindow=1&rlz=1C5CHFA_enUS734US735&biw=1087&bih

History of the Cylinder Phonograph > Inventing Entertainment: The ...
https://www.loc.gov/.../edison...of-edison.../history-of-the-cylinder-phonograph/
The phonograph was developed as a result of Thomas **Edison's** work on two other ... Collection **Inventing** Entertainment: The Early Motion Pictures and Sound ...

Phonograph - Wikipedia
https://en.wikipedia.org/wiki/Phonograph
The phonograph is a device for the mechanical recording and reproduction of sound. In its later ... **The phonograph** was **invented** in 1877 by Thomas **Edison**.
Edison's Phonograph Doll · Phonograph cylinder · Production of phonograph ...

The Phonograph - Thomas Edison National Historical Park (U.S. ...
https://www.nps.gov/edis/learn/kidsyouth/the-phonograph.htm
Feb 26, 2015 - In 1885, Thomas **Edison** wrote, "I have not heard a bird sing since I was ... The first **phonograph** was **invented** in 1877 at the Menlo Park lab.

Edison's Invention of the Phonograph - ThoughtCo
https://www.thoughtco.com › Humanities › History & Culture › American History
May 2, 2018 - Thomas **Edison** achieved widespread early fame by **inventing the phonograph** and startling the public by demonstrating a machine that could ...

Thomas Alva Edison patents the phonograph - Feb 19, 1878 ...
www.history.com/this-day-in-history/thomas-alva-edison-patents-the-phonograph
200,521 for his **invention—the phonograph**—on this day in 1878. **Edison's invention** came about as spin-off from his ongoing work in telephony and telegraphy.

How the Phonograph Changed Music Forever | Arts & Culture ...
https://www.smithsonianmag.com/.../phonograph-changed-music-forever-180957677...
Much like streaming music services today are reshaping our relationship with music, **Edison's invention** redefined the entire industry.

Fig. 9.11 A visualization of attributes for an entity (on the top). Extracting phrases for topics from search results (on the web, intranet or an arbitrary document index, on the bottom)

This assessment can be made by means of discourse-level analysis (Chaps. 7 and 11) or in a simpler, string – based manner. Chunk-of-text extractor performs the task according to the above requirements. Once chunks of text are extracted from various sources, they are put into the index so they can be combined in a chatbot answer document.

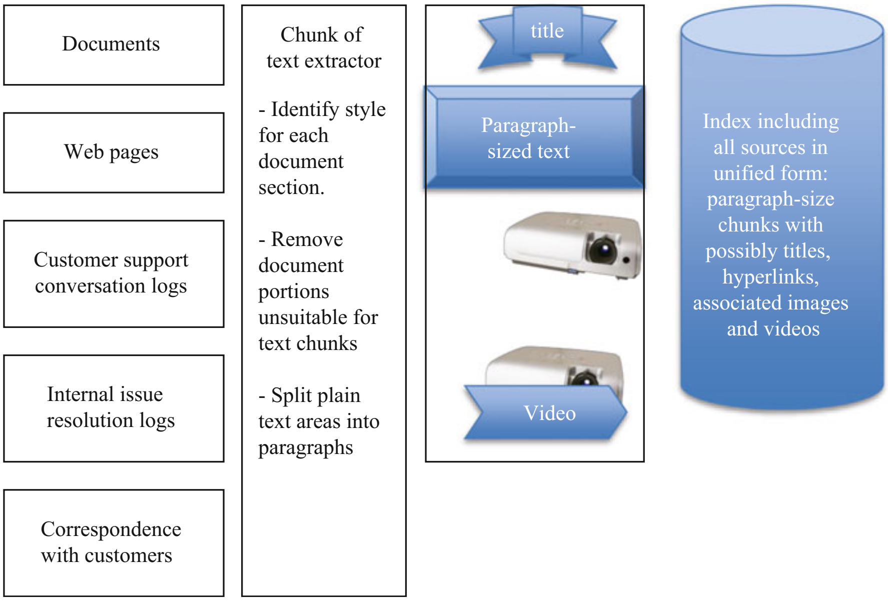

Documents
Web pages
Customer support conversation logs
Internal issue resolution logs
Correspondence with customers
Chunk of text extractor
- Identify style for each document section.
- Remove document portions unsuitable for text chunks
- Split plain text areas into paragraphs
title
Paragraph-sized text
Video
Index including all sources in unified form: paragraph-size chunks with possibly titles, hyperlinks, associated images and videos

Fig. 9.12 Preparing available content for building answers as documents

Chunks of text to be inserted into an answer document need to be extracted from a proper area at a webpage or a proper section of a document, and cleaned. We follow (Baroni et al. 2008, Cai et al. 2003 and Pasternack and Roth 2009) for the algorithm of text extraction from a webpage. Given a page, we need to decide if the page contains an article with desired content, and if it does, find a contiguous block of HTML in the webpage starting with the first word in the article and ending with the last. Finally, we need to remove everything other than the article text (and its included markup tags) itself, such as ads, from the extracted block and output the result. When the first word or last word is nested within one or more pairs of tags, the relevant opening and ending tags are appended to the beginning and ending of the extracted block, respectively. Otherwise, when this nesting is not as above, this one or more pairs of tags can be left open, disrupting the article text's formatting, so we ignore this case.

A chart for the algorithm for building the structure of a conclusive answer is shown in Fig. 9.13. Firstly, a right step in the dialogue to conclude it needs to be determined (a component on the top). Also, a conclusive comprehensive answer is not always a good end for a dialogue. If a dialogue leads to a transaction or a user seems to be knowledgeable enough then no comprehensive answer would be required: the dialogue will be concluded with a transaction confirmation and user knowledge confirmation respectively.

Depending on dialogue type, we build the structure of a conclusive answer (Option 1 and Option 2 from Table 9.2). On the left, we build sections of conclusive

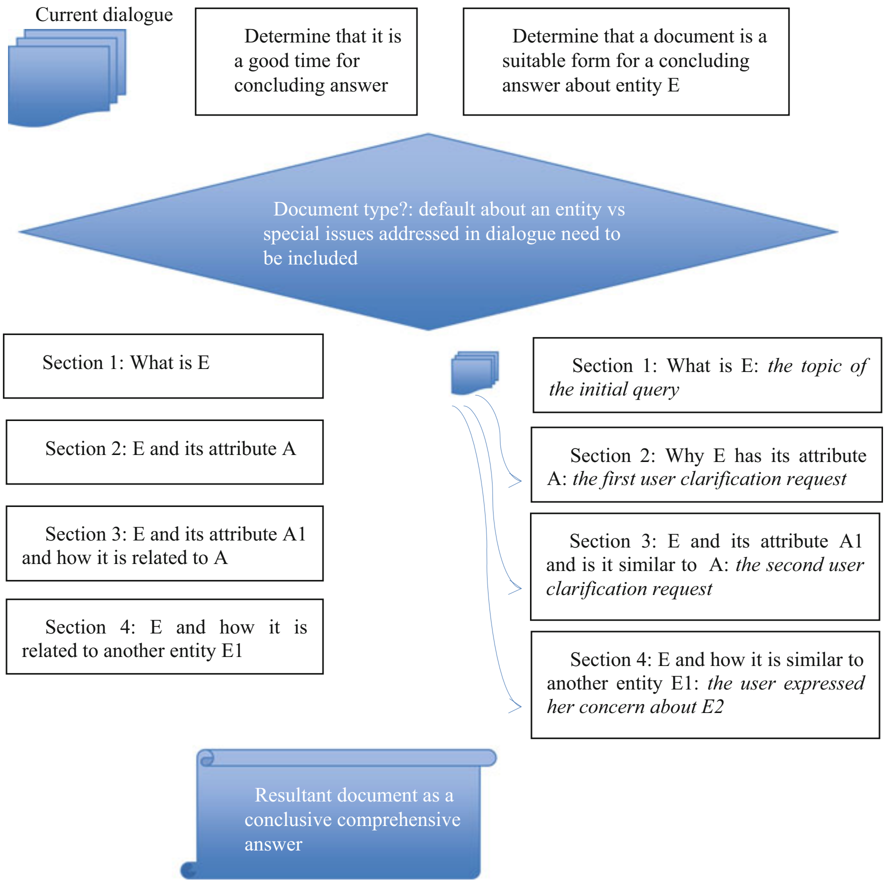

Current dialogue
Determine that it is a good time for concluding answer
Determine that a document is a suitable form for a concluding answer about entity E
Document type?: default about an entity vs special issues addressed in dialogue need to be included
Section 1: What is E
Section 2: E and its attribute A
Section 3: E and its attribute A1 and how it is related to A
Section 4: E and how it is related to another entity E1
Section 1: What is E: *the topic of the initial query*
Section 2: Why E has its attribute A: *the first user clarification request*
Section 3: E and its attribute A1 and is it similar to A: *the second user clarification request*
Section 4: E and how it is similar to another entity E1: *the user expressed her concern about E2*
Resultant document as a conclusive comprehensive answer

Fig. 9.13 An algorithm for relying on the current dialogue to form a conclusive answer

answer from the structure of how entity and its attributes are introduced. On the right, we follow the questions, disagreements and misunderstanding of user utterances about an entity.

9.5.3 Content Compilation Algorithm

The chart for text fragment mining algorithm is shown in Fig. 9.14. We start with the seed, one or multiple sentences each of which will form one or more paragraphs about the respective topics of the TOC. These seed sentences can be viewed as either headers or informational centroids of content to be compiled. We now iterate through each original sentence, build block of content for each and then merge all blocks, preceded by their seed sentences together, similar to (Sauper and Barzilay 2000).

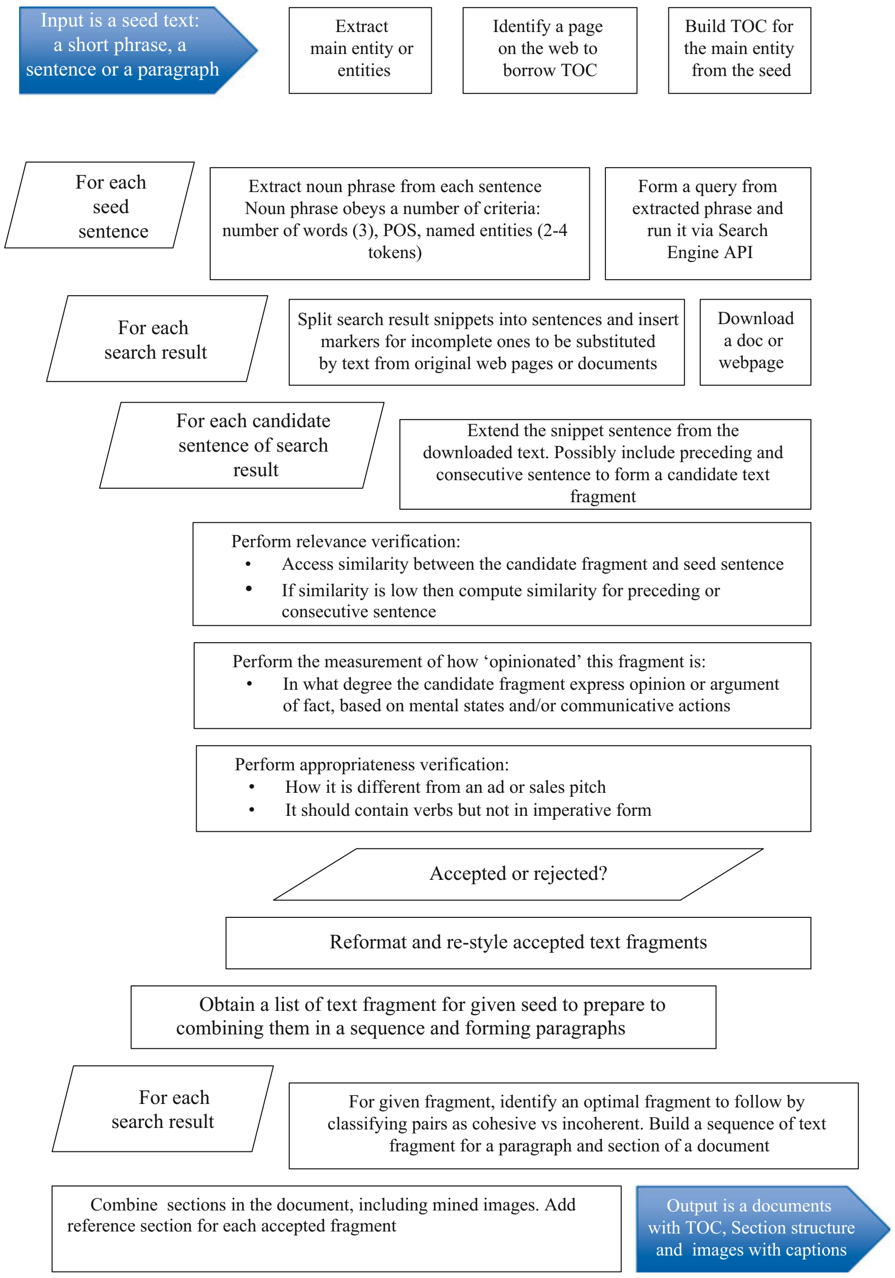

Input is a seed text: a short phrase, a sentence or a paragraph
Extract main entity or entities
Identify a page on the web to borrow TOC
Build TOC for the main entity from the seed
For each seed sentence
Extract noun phrase from each sentence Noun phrase obeys a number of criteria: number of words (3), POS, named entities (2-4 tokens)
Form a query from extracted phrase and run it via Search Engine API
For each search result
Split search result snippets into sentences and insert markers for incomplete ones to be substituted by text from original web pages or documents
Download a doc or webpage
For each candidate sentence of search result
Extend the snippet sentence from the downloaded text. Possibly include preceding and consecutive sentence to form a candidate text fragment
Perform relevance verification:
• Access similarity between the candidate fragment and seed sentence
• If similarity is low then compute similarity for preceding or consecutive sentence
Perform the measurement of how 'opinionated' this fragment is:
• In what degree the candidate fragment express opinion or argument of fact, based on mental states and/or communicative actions
Perform appropriateness verification:
• How it is different from an ad or sales pitch
• It should contain verbs but not in imperative form
Accepted or rejected?
Reformat and re-style accepted text fragments
Obtain a list of text fragment for given seed to prepare to combining them in a sequence and forming paragraphs
For each search result
For given fragment, identify an optimal fragment to follow by classifying pairs as cohesive vs incoherent. Build a sequence of text fragment for a paragraph and section of a document
Combine sections in the document, including mined images. Add reference section for each accepted fragment
Output is a documents with TOC, Section structure and images with captions

Fig. 9.14 A chart of the content compilation algorithm

To find relevant sentences on the web for a seed sentence, we form query as extracted significant noun phrases from this seed sentence: either longer one (three or more keywords, which means two or more modifiers for a noun, or an entity, such as a proper noun). If such queries do not deliver significant number of relevant sentences formed from search results, we use the whole sentence as a search engine query, filtering our content that is a duplicate to the seed (Galitsky and Kuznetsov 2013).

The formed queries are run via search engine API or scraped, using Bing; search results are collected. We then loop through the parts of the snippets to see which sentences are relevant to the seed one and which are not. For all sentences obtained from snippets, we verify appropriateness to form content on one hand, and relevance to the seed sentence on the other hand. Appropriateness is determined based on grammar rules: to enter a paragraph cohesively, a sentence needs to include a verb phrase and be opinionated (Galitsky et al. 2009). We filter out sentences that look like one or another form of advertisement, a call to buy a product, or encourages other user activity by means of an imperative verb.

Relevance is determined based on the operation of syntactic generalization (Galitsky et al. 2012), where the bag-of-words approach is extended towards extracting commonalities between the syntactic parse trees of seed sentence and the text mined on the web. Syntactic generalization score is computed as a cardinality of maximal common sub-graph between the parse trees of the seed and candidate sentences or text fragments. Syntactic generalization allows a domain-independent semantic measure of topical similarity, delivering stronger relevance than the search engine itself or the keyword statistics.

In addition to syntactic generalization, the tool verifies the common entities between seed and mined sentence, and applies general appropriateness metric. The overall score includes syntactic generalization score (the cardinality of maximal set of common syntactic sub-trees, Chap. 5) and appropriateness score to filter out less suitable sentences. Finally, mined sentences are re-styled and re-formatted to better fit together. The following section explains how paragraphs are formed from text fragments.

To find relevant sentences on the web for a seed sentence, we form a query as extracted significant noun phrases from this seed sentence: either longer one (three or more keywords, which means two or more modifiers for a noun, or an entity, such as a proper noun). If such queries do not deliver significant number of relevant sentences formed from the search results, we use the whole sentence as a search engine query, filtering our the content that is duplicate to the seed.

The formed queries are run via search engine API or scraped, using Bing, Yahoo API or Google, as well as their '/news' or '/blogs' subdomains depending on the topic of generated content; the search results are collected. We then loop through the parts of the snippets to see which sentences are relevant to the seed one and which are not. If only a fragment of sentence occurs in the snippet, we need to go to the original page, download it, find this sentence and extract it.

For all sentences obtained from snippets, we verify appropriateness to form a conclusive answer text on one hand, and relevance to the seed sentence on the other hand. Appropriateness is determined based on grammar rules: to enter a paragraph cohesively, a sentence needs to include a verb phrase and/or be opinionated; mental space of cohesive information flow has been explored, for example, in (Galitsky et al. 2008). Relevance is determined based on the operation of syntactic generalization (Galitsky et al. 2010), where the bag-of-words approach is extended towards extracting commonalities between the syntactic parse trees of a seed sentence and the one mined on the web. Syntactic generalization allows a domain-independent semantic measure of topical similarity between a pair of sentences. Without syntactic generalization, a combination of sentences mined on the web would not necessarily form a meaningful text.

In addition to syntactic generalization, the tool verifies common entities between the seed and the mined sentence, and applies a general appropriateness metric. The overall score includes the syntactic generalization score (the cardinality of the maximal common system of the syntactic sub-trees) and the appropriateness score to filter out less suitable sentences. Finally, the mined sentences are modified and re-formatted to better fit together, and are joined to form paragraphs.

9.5.4 A Brief Example of the Content Generation Flow

Imagine we have a user utterance (seed):

(S) '*Give me a break, there is no reason why you can't retire in ten years if you had been a rational investor and not a crazy trader*'.

We start with building TOC for the main entity here, *rational investor*. The other candidates for the main entity are rejected since they are too broad (such as *retire*, a single-word concept), or occur with a negation *not a crazy trader*.

Searching Wikipedia, we find a page for *rational investor* with redirect to *Homo economicus* https://en.wikipedia.org/wiki/Homo_economicus, where the following TOC is scraped:

1. History of the term
2. Model
3. Criticisms
4. Responses
5. Perspectives
6. Homo sociologicus

. . .

The items which can appear on the bottom such as *References* are common for all entities.

For each TOC item, we add a section title keyword to the seed expression. For the default section (here, *Model*), we just use the seed. We need to form queries which contain the main entities from the utterance, retain the meaning but are not too restrictive at the same time.

The main entity here is *retirement* in the form of the verb *retire* and it needs to be constrained by the noun phrase that follows *rational investor*. To form the second query, we combine *rational investor* and the next noun phrase, *not a crazy trader*. Notice that just a single noun phrase with two words is not restrictive enough, and a part of sentence, such as elementary discourse unit, like *there is no reason why you can't retire in ten years* would be too restrictive. Four-five keywords in a query are optimal. Hence two following queries are formed for search engine API:

(Q1) + retire + rational + investor
(Q2) + rational + investor not + crazy + trader

This is not a frequent user query, so web search results need to be further processed: https://www.google.com/search?q=%2Bretire+%2Brational+%2Binvestor.

The following snippet is selected as a candidate to be included in a conclusive answer, since it contains all keywords from Q1.

How to Make Rational Investing Decisions | Sound Mind Investing
https://soundmindinvesting.com/articles/.../how-to-make-rational-investing-decisions
Nov 1, 2014 – How to Make **Rational Investing** Decisions ... pleasant and you'll probably have more money to spend in **retirement** and leave to your heirs.

We download this webpage, extract text from it and find a paragraph that corresponds to the above snippet. We do that for all search results which contains all keywords from the query.

We consider two text fragments from the search results:

(A1a) *If you take the time to understand the psychology of **rational investing**, you'll make your life more pleasant and you'll probably have more money to spend in **retirement** and leave to your heirs.*

(A1b) *One needs many years of relevant data before deciding if a fund manager is truly skilled in **rational investing** or just lucky. Hence, by the time you have enough statistically relevant data to rely on, the manager is likely nearing **retirement.***

We now show the sentence similarity assessment via generalization operator (Chap. 5):

A ^ A_{1a} = RST-Condition (VP (..., NP *rational investing), *- retire)*
A ^ A_{1b} = NP *rational investing), *- retire.*

One can see that in the first search result A_{1a} *retire* and *rational investing* are connected in the similar way to the seed S: *relational investing* is connected by the rhetorical relation *Condition* to the phrase including *retire.* In A_{1b} the syntactic matching part is the same but these phrases occur in two different sentences and are related in a much more complex indirect way than in the seed. Hence A_{1a} is a good fragment to include in the conclusive answer and A_{1b} is not so good.

Once we obtain an unordered list of text fragments for a section, we need to find an optimal order to form the section text. For example, if both above text fragments are accepted (not just the first one), the second should follow the first since it contains the conclusion *…Hence….* And both these fragments are related to the same main entity. Still, the resultant text would not read well since there is a strong deviation of topics towards finding an account manager, which is not the main topic of this section. Given an unordered set of text fragments or paragraphs, we cannot assure cohesiveness of the resultant text but instead at least find an optimal order for these fragments, to minimize a disturbance of content flow and a coherence of the resultant text.

To solve the problem of an optimal sequence, we rely on discourse analysis. It turns out that certain features of logical organization of text encoded via discourse trees are much more stronger criteria of text cohesiveness in comparison with maintaining a topic, as most content generation algorithms do. We devote Chaps. 7, 10 and 11 to this topic, being the focus of this book.

9.5.5 Modeling the Content Structure of Texts

In this section, we consider the problem of modeling the content structure of texts within a specific domain, in terms of the attributes of an entity this texts expresses and the order in which these topics appear. Some research intended to characterize texts in terms of domain-independent rhetorical elements, such as schema items (McKeown 1985) or rhetorical relations (Mann and Thompson 1988; Marcu 1997). Conversely, (Barzilay and Lee 2004) focus on content, domain-dependent dimension of the structure of text. They present an effective knowledge-lean method for learning content models from un-annotated documents, utilizing a novel adaptation of algorithms for Hidden Markov Models. The authors apply their approach to two complementary tasks: information ordering and extractive summarization. The experiments showed that incorporating content models in these applications gives a substantial improvement.

In general, the flow of text is determined by the topic change: how attributes of an entity evolve. (Barzilay and Lee 2004) designed a model that can specify, for example, that articles about mountains typically contain information about height, climate, assents, and climbers. Instead of manually determining the evolution of attributes (the topics for a given domain), a distributional view can be taken. It is possible to machine learn these patterns of attribute evolution directly from un-annotated texts via analysis of word distribution patterns. (Harris 1982) wrote that a number of word recurrence patterns are correlated with various types of discourse structure type.

Advantages of a distributional perspective include both drastic reduction in human effort and recognition of "topics" that might not occur to a human expert and yet, when explicitly modeled, aid in applications. A success of the distributional approach depends on the existence of recurrent patterns. In arbitrary document collections, such recurrent patterns might be too variable to be easily detected by statistical means. However, research has shown that texts from the same domain tend to exhibit high similarity (Wray 2002). At the same time, from the cognitive science perspective, this similarity is not random and is instead systematic, since text structure facilitates a text comprehension by readers and their capability of recall (Bartlett 1932).

We assume that text chunks convey information about a single attribute of an entity (a single topic). Specifying the length of text chunks can defines the granularity of the induced attribute/topic: we select the average paragraph length. We build a content model as a Hidden-Markov Model in which each state s corresponds to a distinct topic and generates sentences relevant to that topic according to a state-specific language model p_s. Note that standard n-gram language models can therefore be considered to be degenerate (single-state) content models. State transition probabilities give the probability of changing from a given topic to another, thereby capturing constraints attribute evolution (topic shift).

We rely on the bigram language models, so that the probability of an n-word sentence $x = w_1\ w_2 \dots w_n$ being generated by a state s

$$p_s(x) = \prod_{i=1}^{n} p_s(w_i|w_{i-1})$$

We will now describe state bigram probabilities $p_s\ (w_i \mid w_{i-1})$

To initialize a set of attributes by partitioning all of the paragraphs (or text chunks) from the documents in a given domain-specific collection into clusters, we do the following. First, we create clusters via complete-link clustering, measuring sentence similarity by the cosine metric using word bigrams as features. Then, given our knowledge that documents may sometimes discuss new and/or irrelevant content as well, we create an AUX cluster by merging together all clusters containing # paragraphs < t (selected threshold). We rely on the assumption that such clusters consist of "outlier" sentences.

Given a set $= c_1, c_2, \dots, c_m$ of m clusters, where c_m is the AUX cluster, we construct a content model with corresponding states $s_1, s_2, \dots, s_m$. we refer to s_m as the insertion state.

For each state s_i $i < m$ bigram probabilities (which induce the state's sentence-emission probabilities) are estimated using smoothed counts from the corresponding cluster

$$p_{s_i}\left(w'|w\right) \overset{def}{=} \frac{f_{c_i}\left(ww'\right)+\delta_1}{f_{c_i}(w)+\delta_1|V|},$$

where *f c₁ (y)* is the frequency with which word sequence *y* occurs within the sentences in cluster c_i, and *V* is the vocabulary.

We want the insertion state s_m to simulate digressions or unseen attributes. We ignore the content of AUX cluster and force the language model to be complementary to those of the other states by setting

$$p_{s_m}\left(w'|w\right) \overset{def}{=} \frac{1-\max_{i:i<m} p_{s_i}\left(w'|w\right)}{\sum_{u\in V}\left(1-\max_{i:i<m} p_{s_i}(u|w)\right)}.$$

Our state-transition probability estimates arise from considering how the paragraphs from the same document are distributed across the clusters. For two clusters *c* and *c'* we define *D(c, c')* as the number of documents in which a paragraph from *c* immediately precedes one from *c'*. *D(c)* is the number of documents containing paragraphs from *c*. For any two states s_i and s_j, $i,j < m$, we rely on the following smooth estimate of the probability of transitioning from s_i to s_j:

$$p\left(s_j|s_i\right) = \frac{D\left(c_i, c_j\right)+\delta_2}{D(c_i)+\delta_2 m}.$$

Programming in NL is another area where the content structure of text is essential (Galitsky and Usikov 2008).

Building Answer Document Based on Similarity and Compositional Semantics The vector representations of the desired document can be obtained using a paragraph vector model (Le and Mikolov 2014) that computes continuous distributed vector representations of varying-length texts. The source documents' section that are semantically close (or similar) to the desired document is identified in this vector space using cosine similarity. The structure of similar articles can then be emulated, the important sections identified and assign relevant web-content or intranet content assigned to the sections.

We utilize the entire Wikipedia to obtain a D-dimensional representations of words/entities as well as documents using the paragraph vector distributed memory model (Le and Mikolov 2014). Similar articles are identified using cosine similarity between the vector representations of the missing entity and representations of the existing entities (entities that have corresponding articles). Content from the similar articles are used to train multi-class classifiers that can assign web-retrieved content on the red-linked entity to relevant sections of the article. The architecture of such system is shown in Fig. 9.15. The paragraph vector distributed memory model is used to identify similar documents to rely upon on one hand and also to make an inference of vector representations of new paragraphs retrieved from the web on the other hand.

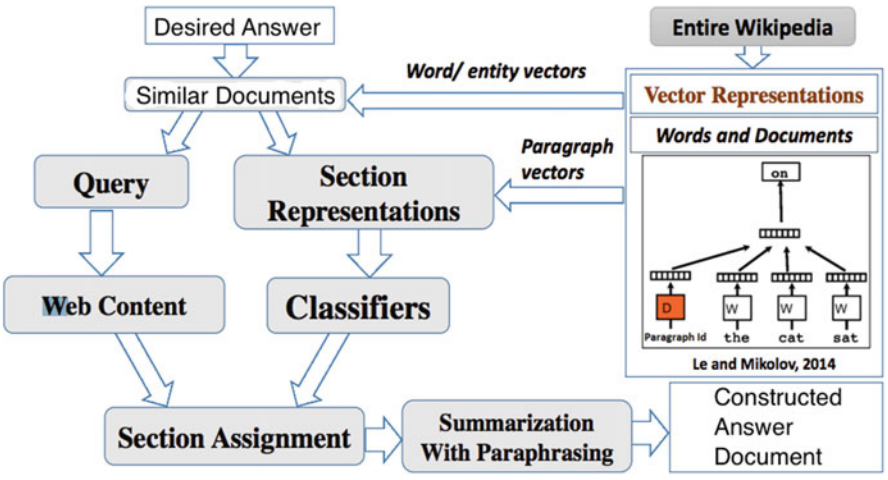


Fig. 9.15 Document generation approach based on similar document and wikipedia content

We take a sequence of words from a similar document and approach the last word that can be reused. Then we attempt to predict the next word using PV-DM. The PV-DM model is based on the principle that several contexts sampled from the paragraph can be used to predict the next word. Given a sequence of T words ($w_1, w_2, \ldots, w_T$), the task is to maximize the average log probability. In the top equation, c is the size of the context (number of words before and after the current word to be used for training). The conditional probability of w_{t+j} given w_t can be obtained by the softmax function (Bridle 1990) in the equation below, where v_{wt+j} and v_w refers to the output and the input vector representations of the word w, respectively. W refers to the total number of words in the vocabulary

$$F = \frac{1}{T}\sum_{t=1}^{t=T} \sum_{-c\le j\le c, j\ne 0} \log p\left(w_{t+j}|w_t\right)$$

$$p\left(w_{t+j}|w_t\right) = \frac{\exp\left(v'_{w_{t+j}} T_{v_{w_t}}\right)}{\sum_{w=1}^{W} \exp\left(v'_w T_{v_{w_t}}\right)}$$

9.5.6 Related Work on Conclusive Answers

Whereas chatbot algorithms in general belong to such computer science discipline as search engineering and general-purpose NLP, automated building of conclusive answers fall under the content generation area of AI. Automating answer creation, it is hard to compete with how human domains experts would do it; however, chatbots are expected to be capable of building tens of thousands of conclusive answer per vertical domain on the fly.

In the modern society, writing and creating content is one of the most frequent human activities. An army of content creators, from students to professional writers, produce various kinds of documents for various audiences. Not all of these documents are expected to be innovative, break-through or extremely important. The target of the tool being proposed is assistance with routine document creation process where most information is available on the web and needs to be collected, integrated and properly referenced (Galitsky and Kuznetsov 2013).

A number of content generation software systems are available in specific business domains (Johnson 2016). Most of content generation software are template-based which limits their efficiency and volume of produced content (Hendrikx et al. 2013). An interesting class of content generation system is based on verbalizing some numerical data. Also, content generation for computer game support turned out to be fruitful (Liapis et al. 2013). Deep-learning – based generation of a sequence of words has a limited applicability for large-scale content production industrial systems. In (Galitsky 2016) we built a content compilation assistance system that was suitable for producing a report that can be subject to and manual editing by students, researchers in various fields in science, engineering, business and law.

Previous work on content generation in general and specifically related to web content relied heavily on manually annotated information of Wikipedia categories (Sauper and Barzilay 2009; Banerjee and Mitra 2016). Articles in Wikipedia consist of sections. (Sauper and Barzilay 2009) retrieved content from the web on articles belonging to a certain category of diseases by using the most frequent section titles as keywords to retrieve relevant web search snippets, utilizing web mining, similar to what we do for chatbot answers. The most informative excerpts were selected using a perceptron-based framework and populated into the built web article. In a recent work, (Banerjee and Mitra 2016) proposed WikiKreator where contents in the Wikipedia sections were represented by topic-distribution features using Latent Dirichlet Allocation (LDA, Blei et al. 2003).

To build a document from multiple sources, sentences selected and paraphrased from multiple documents must be ordered such that the resulting article is coherent. Existing summarization systems did not tackle coherence, so discourse level consideration proposed in Chap. 10 needs to be utilized.

The discourse tree representation used in our content compilation system is a reduction of what is called parse thicket (Chap. 7), a combination of parse trees for sentences with discourse-level relationships between words and parts of the sentence in one graph. The straight edges of this graph are syntactic relations, and curvy arcs – discourse relations, such as anaphora, same entity, sub-entity, rhetoric relation and communicative actions. This graph includes much richer information than just a combination of parse trees for individual sentences would.

Galitsky (2016) introduced the tool has been advertised using Google AdWords and used by thousand of users searching for "free essay writing" to compile content for a variety of domains, including natural sciences and humanities. In this section the proposed and evaluated technique found a new application area in building answers for chatbots.

9.6 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 Sect. 9.2 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 https://github.com/bgalitsky/relevance-based-on-parse-trees/blob/master/src/main/java/opennlp/tools/parse_thicket/kernel_interface/MultiSentenceKernelBasedSearchResultsProcessor.java.

9.6.1 Performance Analysis of the Content Pipeline Components

We selected to evaluate the de-duplication unit because 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 σ_{within} is standard deviation of the differences within the annotations for the same text, σ_{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.

9.6.1.1 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 (Chap. 5). Analysis is performed varying the number of keywords from 1 to 5 (Table 9.3).

The baseline computation relied on (Levenshtein 1966) and (Jaccard 1912) string distance computations.

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,

Table 9.3 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

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.

9.6.1.2 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 9.4).

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.

The Rows of Table 9.5 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 about kitchen appliances. Therefore,

Table 9.4 Evaluation of sentiment analysis: meaningful vs meaningless sentences

Domain	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 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%

Table 9.5 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%

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.

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 (Socher et al. 2013) 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 (Jindal and Taneja 2017) to overwrite the decision of (Socher et al. 2013) when a given sentence gives a substantial overlap with one of the templates via SG (Table 9.6).

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

Table 9.6 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%

template – enabled approach experienced more difficulties with negative sentiments than with positive.

9.6.1.3 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 Sect. 9.5.5 to form candidate sentences for inclusion in an ad line. The training dataset of 10,000 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 9.7).

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. 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

Table 9.7 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 webpages	200/800	63.0%	69.2%	64.7%	66.9%
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%

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.

9.6.2 *Performance Analysis of Personalized Recommendations*

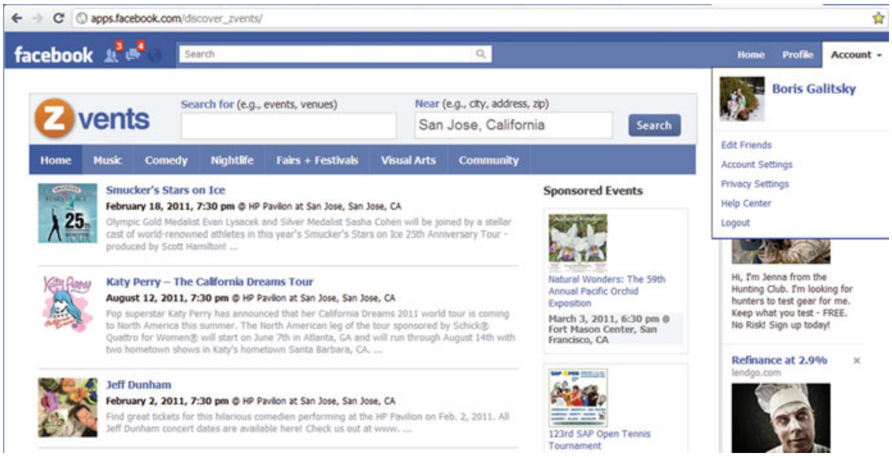
In this section we perform evaluation of personalized recommendations. Since this component is the consumer of the content pipeline, this evaluation can be viewed as the one for the overall system. User interface of apps.facebook.com/discover_zvents is shown in Fig. 9.16.

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 9.8).



Smucker's Stars on Ice
February 18, 2011, 7:30 pm @ HP Pavilon 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 Pavilon 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 Pavilon 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. 9.16 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

Table 9.8 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 9.9 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

2. A user specifies search query for a certain type of Events (Table 9.9). 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 columns of Tables 9.8 and 9.9 indicate the percentages 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 a 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.

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 9.10).

9.6.3 *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 thesaurus improve 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.

Table 9.10 Categories of most popular events in cities with the highest numbers of events

Location	T	E1	E2	E3
New York	13,092	120.50	41.80	245.00
San Francisco	5522	57.53	18.77	105.95
Las Vegas	4282	47.99	15.02	40.58
Los Angeles	4102	43.15	14.02	51.12
Boston	3798	41.85	12.52	59.66
Chicago	3515	41.03	11.61	40.70
Houston	3075	38.42	10.85	32.03
Atlanta	2757	27.68	9.05	36.88
Nashville	2693	27.96	9.37	30.10
Austin	2574	24.22	9.14	66.30
Denver	2518	26.31	8.02	32.03
Lexington	2140	23.17	6.88	18.04
Charleston	2131	23.33	7.24	18.01
Philadelphia	2062	18.00	6.71	27.12
San Diego	1930	23.17	6.06	21.07
St Louis	1910	17.58	6.35	17.23
Washington	1875	17.30	6.53	9.10
Fresno	1867	20.01	6.53	14.70
Seattle	1861	15.65	5.52	30.42
Average	3352.8	34.40	11.10	47.10
Percentage personalization improves the number of discovered events		137%	423%	100%

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 thesaurus-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 $1/n\ \Sigma_{i\,=\,1\ldots n}\ 1/rk_i$ 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 10,000 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 9.11.

Table 9.11 Evaluation of chatbot search relevance

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 thesaurus, %, averaging over 20 searches	Relevancy of re-sorting by TK, %, averaging over 20 searches	Relevancy of re-sorting by using thesaurus 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

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.

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 &lt;2% improvement, and &lt; 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 standalone SG by 4% but is well below the hybrid SG + thesaurus search accuracy.

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

9.7 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. (de Salvo Braz et al. 2005) 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 Chap. 5.

The special issue of Information Sciences Journal on Intelligent knowledge-based models and methodologies for complex information systems (Cuzzocrea 2012) provides a good comparison framework for this chapter. Generalized association rule extraction is a powerful tool to discover a high level view of the interesting patterns hidden in the analyzed data (Baralis et al. 2012). 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 (Galitsky 2015). In case of the 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 has higher variability than the text for the pipeline in this chapter, the authors achieved F-measure of 71.6. It is significantly lower of F-measure of above 90% achieved in the current study for search. (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 9.12 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.

Until recently, most approaches to semantics involve one or another form of mapping into first order logic forms without using acquired rich knowledge sources.

Table 9.12 A comparison between the different systems using the Twitter training corpus provided by the SemEval Sentiment Analysis task (Rosenthal et al. 2014)

	Twitter				
Experiment	Dev	Test	Sarcasm	SMS	LiveJournal
Majority	29.19	34.64	27.73	19.03	27.21
Purchase-based system	62.09	64.74	40.75	56.86	62.22
Tweet-level system	62.4	63.73	42.41	69.54	69.44
Combined system	64.6	65.42	40.02	59.84	68.79

A concise Frege-type graph language programmed as a tool for logic programming is presented in (Redey 1993), implementing the principle that the logical structure of NL statements can be constructed in a language equivalent of the first-order logic established solely via the basic natural grammatical relations. (Nagarajan and Chandrasekar 2014) proposed a new sentiment analysis algorithm that achieves high accuracy results by taking into account the expectations of the customers along with the inclusion of neutral words for analysis.

A number of methods measure text similarity numerically, similarly to SG score. (Wenyin et al. 2010) 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 (Chesñevar et al. 2009). Sagui et al. (2009) implemented intelligent processing of web-based forms and intelligent robotic soccer (Ferretti et al. 2007), among many other applications.

DeLP has also been used for modeling thesaurus 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. Gomez et al. (2010) 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 thesaurus, a dialectical analysis is performed on a DeLP program obtained from such thesaurus, where all arguments in favor and against the final answer of the query will be taken into account. The current chapter presents an industrial system for handling a special case of thesaurus inconsistencies by using a different mechanism of mapping, but similar underlying logic of argumentation.

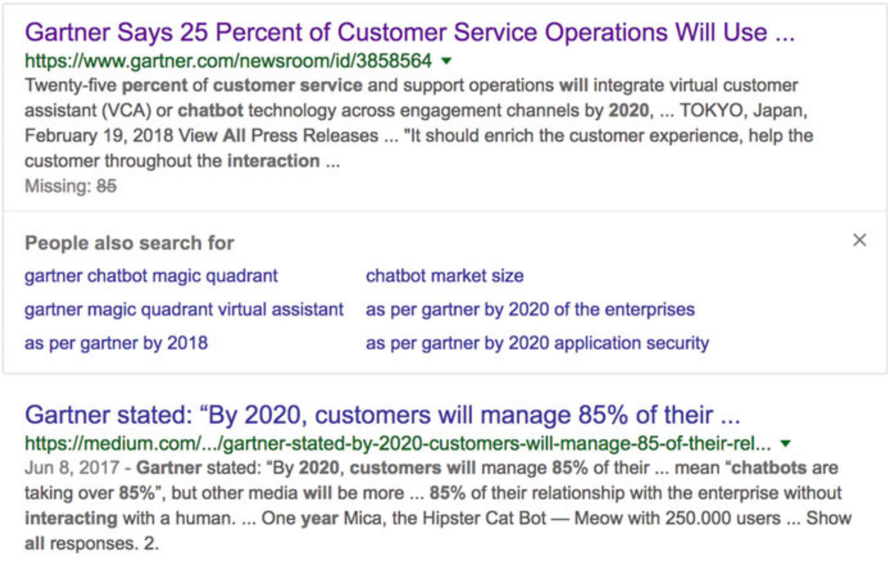
Defeasible reasoning is a rule-based approach for efficient reasoning with incomplete and inconsistent information. Such reasoning is, among others, useful for thesaurus integration, where conflicting information arises naturally; and for the modeling of business rules and policies, where rules with exceptions are often used. Category mapping is an (example of such domain, where we need to systematically treat exceptions. (Antoniou et al. 2001) 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 is:

- syntactically compatible with RuleML;
- based on strict and defeasible rules and priorities;
- based on a translation to logic programming with declarative semantics;
- flexible and adaptable to different intuitions within defeasible reasoning.

Tree Kernel methods (TK) for text learning 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 an unassigned instance by comparing it to the labeled training instances using this kernel function. 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 chapter, 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.

In our studies (Galitsky et al. 2012: Galitsky 2017) 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



Gartner Says 25 Percent of Customer Service Operations Will Use ...
https://www.gartner.com/newsroom/id/3858564
Twenty-five **percent** of **customer service** and support operations **will** integrate virtual customer assistant (VCA) or **chatbot** technology across engagement channels by **2020**, ... TOKYO, Japan, February 19, 2018 View **All** Press Releases ... "It should enrich the customer experience, help the customer throughout the **interaction** ...
Missing: ~~85~~

People also search for
gartner chatbot magic quadrant
chatbot market size
gartner magic quadrant virtual assistant
as per gartner by 2020 of the enterprises
as per gartner by 2018
as per gartner by 2020 application security

Gartner stated: "By 2020, customers will manage 85% of their ...
https://medium.com/.../gartner-stated-by-2020-customers-will-manage-85-of-their-rel...
Jun 8, 2017 - **Gartner** stated: "By **2020**, **customers will** manage **85%** of their ... mean "**chatbots** are taking over **85%**", but other media **will** be more ... **85%** of their relationship with the enterprise without **interacting** with a human. ... One **year** Mica, the Hipster Cat Bot — Meow with 250.000 users ... Show **all** responses. 2.

Fig. 9.17 Original source of information and its misrepresentation by a content aggregator

to other domains (Galitsky et al. 2005, 2009) 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.

In (Galitsky et al. 2013) and Chap. 7 we built a paragraph-level structure which is a sum of parse trees of sentences and called it *parse thicket*. In this chapter 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.

The issue of fake content has been rising dramatically in recent years. Motivated by a number of reasons, content aggregator distorts the content and make unsubstantiated claims with attribution for their advantage. In Fig. 9.17 we see an original Garner post mentioning 25% by 2020 (on the top), and the medium.com representation of Gartner claim boosted to 85%.

9.8 Conclusions

9.8.1 *From Search Engines to Chatbots*

The fields of relevance support for CMS have become critical to a modern workplace. Finding, documenting, and *knowing* things in an environment where data is dispersed, employees are always on the fly, and career paths change fast must be intuitive, simple, and seamless (Wade 2018). Finding content in a site structure requires a mental roadmap of where things live. Search may provide good results, but not direct answers; the answer is usually in the file it returns, meaning more time digesting to understand. Chatbots give users a chance to jump straight to the answer while pointing the to the source for reference, saving everyone time and bridging what is becoming a major gap in CMS (Fig. 9.18).

In a default site and library hierarchy, the files can be well organized using a strong folder or metadata structure. The effectiveness of the file hierarchy depends on the strategy used for initially organizing the content and how well the owner of the hierarchy has maintained the structure and the content over time. With search, the content is not organized before it gets to an inverse index. A search engine will provide multiple results in an organic fashion based on keyword matches, any metadata refiners, and, past popularity of the search results and other considerations. This can provide an efficient access when the users have no idea where to find the information of interest or intend to save time.

With a chatbot, the CMS manager is expected to predict what users want to see and provide direct responses and direct answers. From the users standpoint, the information is not organized either (even though on the back end it is) nor does it

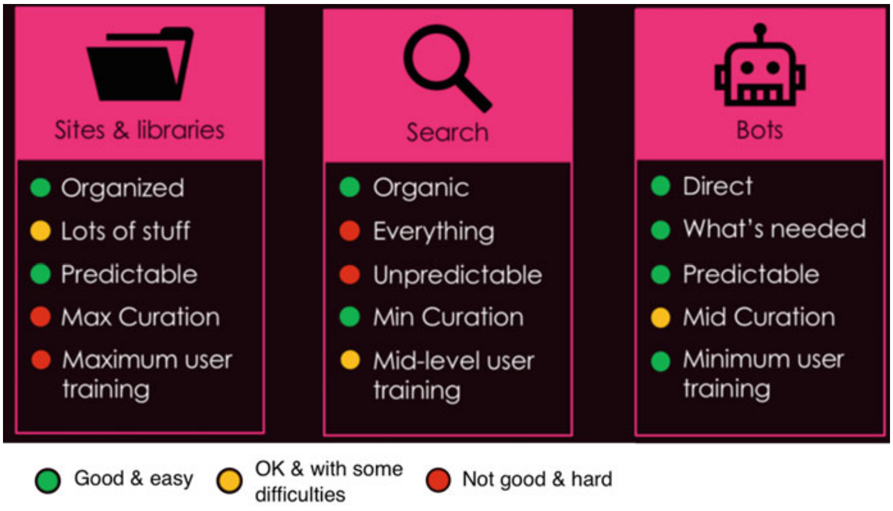


Fig. 9.18 From sites and libraries to search and to chatbots

provide organic options like search. Instead, the chatbot gives the best answer it has and also does it in a conversational way. This direct method of providing information means the user does less work for the same information gain and can perform the task over and over as necessary.

With search, the users are given the results that just recently combined everything that they have access to. Even a user who knows searching best practices on top of a system with a smart search setup including promoted ones and customer refined personalized ones, the user still has to deal with extraneous results that just are not always relevant. From keywords that overlap (e.g., "extension" for files info or telephone numbers) to outdated information, one must sift through plenty of hits in search due to the nature of its organic results. It can lead to an overall negative impact on the overall search experience.

With chatbots, the information available is fully specified by the developers who tune the information in the bot CMS. Chatbot developers direct users to the source information they seek. A good chatbot with relevant CMS has answers to most common questions for each group or department in an organization, actually answers the question being asked (rather than solely providing a source for the answer), and links back to the source as a reference for further information.

9.8.2 *Relevance in a CMS*

In this chapter, reporting from industry, we addressed an issue of relevance in a content pipeline for a chatbot. Although a limitation of keyword-based approach for content collection, cleaning, aggregation and indexing is well understood (Sidorov 2013), 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 (Chap. 5), phrases and paragraphs (Chap. 7) to maintain relevance. We conducted a comparative analysis of syntactic generalization, tree kernel and statistical approaches and selected the former as superior for industrial applications, based on our evaluation of their performances.

The explosion of chatbot applications and machine learning provides a new approach to real-time, personalized customer experiences. In comparison with search engines, chatbots are the ultimate culmination of the universal content management and personalization efforts. Organizations which do not acknowledge the transformative value of chatbots are expected to experience difficulties in reaching and retaining customers.

Vo and Popescu (2016) proposed a four-layer system that leverages word alignment, sentence structure and semantic word similarities for relevance assessment. Recently, Adidas' chatbot achieved a retention rate of 60%, far better than their app (Simplea 2018); Just Eat's chatbot saw a conversion rate of 266% higher than their average social ad; and CONVRG's chatbot received a response rate 3 times that of their email survey.

According to new research from Juniper, banking, healthcare, social, eCommerce and retail organizations saved $20 million this year, with a savings of $8 billion per year expected by 2022. Making content accessible and versatile is now more important than ever for content producers. According to (Gartner 2018), chatbots will power 25% of all customer service interactions by the year 2020.

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 (Suykens et al. 2003) 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 and 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 and Altınçay 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 (2018), 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 chapter 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 (Aleman-Meza et al. 2003; Ding et al. 2004). 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. (Janusz et al. 2012) present 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 chapter 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 (Rubiolo et al. 2012) who presented an ANN-based thesaurus 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 thesaurus 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 chapter 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 https://github.com/bgalitsky/relevance-based-on-parse-trees as a part of (OpenNLP 2018), 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, thesaurus 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 (Galitsky 2012) 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.

The evaluation version of both SG and TK are available at https://github.com/bgalitsky/relevance-based-on-parse-trees/tree/master/src/main/java/opennlp/tools/parse_thicket/kernel_interface and https://github.com/bgalitsky/relevance-based-on-parse-trees/tree/master/src/main/java/opennlp/tools/textsimilarity.

References

Aleman-Meza B, Halaschek C, Arpinar I, Sheth A (2003) A context-aware semantic association ranking. In: Proceedings of the first inernational workshop semantic web and databases (SWDB'03), pp 33–50

Antoniou G, Billington D, Governatori G, Maher M (2001) Representation results for defeasible logic. ACM Trans Comput Log 2(2):255–287

Banerjee S, Mitra P (2016) WikiWrite: generating wikipedia articles automatically. IJCAI, New York

Baralis E, Cagliero L, Cerquitelli T, Garza P (2012) Generalized association rule mining with constraints. Inf Sci 194:68–84

Baroni M, Chantree F, Kilgarriff A, Sharoff S (2008) Cleaneval: a competition for cleaning web pages. In: Calzolari N, Choukri K, Maegaard B, Mariani J, Odjik J, Piperidis S, Tapias D (eds) Proceedings of the sixth international language resources and evaluation (LREC'08)

Bartlett FC (1932) Remembering: a study in experimental and social psychology. Cambridge University Press

Barzilay R, Lee L (2004) Catching the drift: probabilistic content models, with applications to generation and summarization. HLT-NAACL

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1022

Bordini RH, Braubach L (2006) A survey of programming languages and platforms for multi-agent systems. Informatica 30:33–44

Bridle JS (1990) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: Neurocomputing. Springer, pp 227–236

Brzezinski D, Stefanowski J (2011) Accuracy updated ensemble for data streams with concept drift. In: Proceedings of HAIS 2011, Springer Verlag lecture notes in artificial intelligence 6679, pp 155–163

Cai D, Yu S, Wen J-R, Ma W-Y (2003) Extracting content structure for web pages based on visual representation. In: Zhou X, Zhang Y, Orlowska ME (eds) APWeb, volume 2642 of LNCS, Springer, pp 406–417

Cascading (2013) Welcome to the Cascading ecosystem. www.cascading.org

Chesñevar C, Maguitman A, González MP (2009. Empowering recommendation technologies through argumentation. In: Rahwan I, Simari G (eds) Argumentation in artificial intelligence, Springer Verlag, (505 p, in press). ISBN 978-0-387-98196-3

Cumby C, Roth D (2003) On kernel methods for relational learning. In: ICML, pp 107–14

Cuzzocrea A (Editorial) (2012) Intelligent knowledge-based models and methodologies for complex information systems. Inf Sci 194:1–282

de Salvo Braz R, Girju R, Punyakanok V, Roth D, Sammons M (2005) An inference model for semantic entailment in natural language. In: Proceedings of AAAI-05

Ding L, Finin T, Joshi A, Pan R, Cost RS, Peng Y, Reddivari P, Doshi V, Sachs J (2004) Swoogle: a search and metadata engine for the semantic web. In: Proceedings of 13th ACM international conference on information and knowledge management (CIKM'04), pp 652–659

Erenel Z, Altınçay H (2012) Nonlinear transformation of term frequencies for term weighting in text categorization. Eng Appl Artifi Intell 25(7):1505–1514

Ferretti E, Errecalde M, García AJ, Simari GR (2007) An application of defeasible logic programming to decision making in a robotic environment. In: LPNMR, pp 297–302

Galitsky B (2003) Natural language question answering system: technique of semantic headers. Advanced Knowledge International, Adelaide

Galitsky B (2012) Machine learning of syntactic parse trees for search and classification of text. Eng Appl AI 26(3):1072–1091

Galitsky B (2013) Transfer learning of syntactic structures for building taxonomies for search engines. Eng Appl Artif Intell 26(10):2504–2515

Galitsky B (2014) Learning parse structure of paragraphs and its applications in search. Eng Appl of AI 32:160–184

Galitsky B (2015). Finding a lattice of needles in a haystack: forming a query from a set of items of interest. In: FCA4AI@IJCAI

Galitsky B (2016) A tool for efficient content compilation. In: COLING Demo C16-2042 Osaka, Japan

Galitsky B (2017) Matching parse thickets for open domain question answering. Data Knowl Eng 107:24–50

Galitsky B, de la Rosa JL (2011) Concept-based learning of human behavior for customer relationship management. Spec Issue Inf Eng Appl Based on Lattices. Inf Sci 181 (10):2016–2035

Galitsky B, Ilvovsky D (2017) Chatbot with a discourse structure-driven dialogue management. In: EACL Demo E17-3022, Valencia, Spain

Galitsky B, Kovalerchuk B (2014) Improving web search relevance with learning structure of domain concepts. In: Clusters, orders, and trees: methods and applications, pp 341–376

Galitsky B, Kuznetsov SO (2013) A web mining tool for assistance with creative writing. In: ECIR 2013: advances in information retrieval, pp 828–831

Galitsky B, Levene M (2007) Providing rating services and subscriptions with web portal infrastructures. In: Encyclopedia of portal technologies and applications, pp 855–862

Galitsky B, Usikov D (2008) Programming spatial algorithms in natural language. In: AAAI workshop technical report WS-08-11, Palo Alto, pp 16–24

Galitsky B, Kuznetsov SO, Samokhin MV (2005) Analyzing conflicts with concept-based learning. In: International conference on conceptual structures, pp 307–322

Galitsky B, Kuznetsov SO, Kovalerchuk B (2008) Argumentation vs meta-argumentation for the assessment of multi-agent conflict. Proc. of the AAAI Workshop on Metareasoning

Galitsky B, Chen H, Du S (2009) Inversion of Forum Content Based on Authors' Sentiments on Product Usability. AAAI Spring Symposium: Social Semantic Web: Where Web 2.0 Meets Web 3.0, pp 33–38

Galitsky B, Dobrocsi G, de la Rosa JL (2010) Inverting semantic structure under open domain opinion mining twenty-third international FLAIRS conference

Galitsky B Dobrocsi G, de la Rosa JL, Kuznetsov SO (2011) Using generalization of syntactic parse trees for taxonomy capture on the web. In: ICCS, pp 104–117

Galitsky B, Dobrocsi G, de la Rosa JL (2012) Inferring the semantic properties of sentences by mining syntactic parse trees. Data Knowl Eng 81:21–45

Galitsky B, Usikov D, Kuznetsov SO (2013) Parse thicket representations for answering multi-sentence questions. In: 20th international conference on conceptual structures, ICCS

Galitsky B, Ilvovsky D, Kuznetsov SO (2015) Text classification into abstract classes based on discourse structure. In: Proceedings of recent advances in natural language processing, Hissar, Bulgaria, Sep 7–9 2015, pp 200–207

Garcia A, Simari G (2004) Defeasible logic programming: an argumentative approach. Theory Pract Logic Program 4:95–138

Gartner (2018) Gartner says 25 percent of customer service operations will use virtual customer assistants by 2020. https://www.gartner.com/newsroom/id/3858564

Go A, Bhayani R, Huang L (2009) Twitter sentiment classification using distant supervision. Technical Report. Stanford University

Gomez SA, Chesñevar CI, Simari GR (2010) Reasoning with inconsistent ontologies through argumentation. Appl Artif Intell 24(1 & 2):102–148

Gomez H, Vilariño D, Pinto D, Sidorov G (2015) CICBUAPnlp: graph-based approach for answer selection in community question answering task. In: Sem Eavl-2015, pp 18–22

Google (2018) Search using autocomplete. https://support.google.com/websearch/answer/106230

Harris Z (1982) Discourse and sublanguage. In: Kittredge R, Lehrberger J (eds) Sublanguage: studies of language in restricted semantic domains. Walter de Gruyter, Berlin, New York, pp 231–236

Hendrikx M, Meijer S, Van Der Velden J, Iosup A (2013) Procedural content generation for games: a survey. ACM Trans Multimed Comput Commun Appl 9(1), Article 1, 22 pages iGoDigital (2013) https://www.crunchbase.com/organization/igodigital

Jaccard P (1912) The distribution of the flora in the alpine zone. New Phytol 11:37–50

Janusz A, Ślęzak D, Nguyen HS (2012) Unsupervised similarity learning from textual data. Fundam Inform 119(3):319–336

Jindal R, Taneja S (2017) A novel weighted classification approach using linguistic text mining. Int J Comput Appl 180(2):9–15

Johnson MR (2016) Procedural generation of linguistics, dialects, naming conventions and spoken sentences. In: Proceedings of 1st international joint conference of DiGRA and FDG

Kong F, Zhou G (2011) Improving tree kernel-based event pronoun resolution with competitive information. In: Proceedings of the twenty-second international joint conference on artificial intelligence, vol 3, pp 1814–1819

Krippendorff K (2004) Reliability in content analysis: some common misconceptions and recommendations. Hum Commun Res 30(3):411–433

Kuncheva LI (2004) Classier ensembles for changing environments. In: Roli F, Kittler J, Windeatt T (eds) Multiple classifier systems, LNCS, vol 3077. Springer, Heidelberg, p 1

Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In Proceedings of the 31st International Conference on International Conference on Machine Learning – Volume 32 (ICML'14), Eric P. Xing and Tony Jebara (Eds.), Vol 32

Leouski AV, Croft WB (1996) An evaluation of techniques for clustering search results. UMass Tech Report #76. http://ciir.cs.umass.edu/pubfiles/ir-76.pdf

Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. Sov Phys Dokl 10(8):707–710

Liapis A, Yannakakis GN, Togelius J (2013) Sentient sketchbook: computer-aided game level authoring. In: InFDG, pp 213–220

Mahout (2013) https://mahout.apache.org

Makhalova T, Ilvovsky DA, Galitsky B (2015) News clustering approach based on discourse text structure. In: Proceedings of the First Workshop on Computing News Storylines @ACL

Mann WC, Thompson SA (1988) Rhetoric al structure theory: toward a functional theory of text organization. Text 8(3):243–281

Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval. Cambridge University Press, Cambridge UK

Marcu D (1997) The rhetorical parsing, summarization, and generation of natural language texts. Unpublished Ph.D. dissertation, University of Toronto, Toronto, Canada

Mavridis T, Symeonidis AL (2014) Semantic analysis of web documents for the generation of optimal content. Eng Appl Artif Intell 35:114–130

Moschitti A (2008) Kernel methods, syntax and semantics for relational text categorization. In: Proceeding of ACM 17th conference on information and knowledge management (CIKM). Napa Valley, California

McKeown KR (1985) Text generation: using discourse strategies and focus constraints to generate natural language text. Cambridge University Press, Cambridge, UK

Nagarajan V, Chandrasekar P (2014) Pivotal sentiment tree classifier. Int J Sci Technol Res 3 (11):190

OpenNLP (2018.) https://opennlp.apache.org/

Pak A, Paroubek P (2010) Twitter as a corpus for sentiment analysis and opinion mining. In: Nicoletta Calzolari N (ed) LREC'

Pasternack J, Roth D (2009) Extracting article text from the web with maximum subsequence segmentation. In: WWW '09: proceedings of the 18th international conference on world wide web, ACM, New York, pp 971–980

Rahwan I, Amgoud L (2006) An argumentation based approach for practical reasoning. In: International joint conference on autonomous agents and multi agent systems, pp 347–354

Rédey G (1993) Conformal text representation. Eng Appl Artif Intell 6(1):65–71

Rosenthal S, Ritter A, Nakov P, Stoyanov V (2014) SemEval-2014 task 9: sentiment analysis in Twitter. In: SemEval-2014

Rubiolo M, Caliusco ML, Stegmayer G, Coronel M, Gareli Fabrizi M (2012) Knowledge discovery through ontology matching: an approach based on an artificial neural network model. Inf Sci 194:107–119

Sagui F, Maguitman A, Chesñevar C, Simari G (2009) Modeling news trust: a defeasible logic programming approach. Iberoam J Artif Intell 12(40):63–72. Edited by AEPIA (Spanish Association of Artificial Intelligence), Madrid, Spain, ISSN 1137-3601

Sauper C, Barzilay R (2000) Automatically generating wikipedia articles: a structure-aware approach, Proceedings of ACL

Sauper C, Barzilay R (2009) Automatically generating wikipedia articles: a structure-aware approach. In: Proceedings of ACL. Suntec, Singapore, pp 2008–2016

Sidorov G (2013) Syntactic dependency based N-grams in rule based automatic English as second language grammar correction. Int J Comput Linguist Appl 4(2):169–188

Sidorov G (2014) Should syntactic N-grams contain names of syntactic relations? Int J Comput Linguist Appl 5(1):139–158

Simplea (2018) AI Marketing, Chatbots, and Your CMS. https://simplea.com/Articles/AI-Marketing-Chatbots-and-Your-CMS

Socher R, Perelygin A, Wu J, Chuang J, Manning C, Ng A, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: Conference on empirical methods in natural language processing (EMNLP 2013)

Suykens JAK, Horvath G, Basu S, Micchelli C, Vandewalle J (eds) (2003) Advances in learning theory: methods, models and applications, NATO-ASI series III: computer and systems sciences, vol 190. IOS Press, Amsterdam

Tneogi (2018) Conversational interfaces need a different content management system. Chatbot Magazine. https://chatbotsmagazine.com/conversational-interfaces-need-a-different-content-management-system-b105bb6f716

Tunkelang D (2018) Search results clustering. https://queryunderstanding.com/search-results-clustering-b2fa64c6c809

Varshavsky R, Moshe T, Yuval P, Wilson DB (2010) Group recommendations in social networks. US Patent App 20110270774, Microsoft

Vo NPA, Popescu O (2016) A multi-layer system for semantic textual similarity. In: 8th international conference on knowledge discovery and information Retrieval

Vo NPA, Magnolini S, Popescu O (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, USA

Wade M (2018) 5 ways chatbots are revolutionizing knowledge management. AtBot. https://blog.getbizzy.io/5-ways-chatbots-are-revolutionizing-knowledge-management-bdf925db66e9

Wenyin L, Quan X, Feng M, Qiu B (2010) A short text modeling method combining semantic and statistical information. Inf Sci 180(20):4031–4041

Wray A (2002) Formulaic language and the lexicon. Cambridge University Press, Cambridge

Zarrella G, Henderson J, Merkhofer EM, Strickhart L. (2015) MITRE: seven systems for semantic similarity in tweets. Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)