WeKnowIt

Emerging, Collective Intelligence for Personal, Organisational and Social Use

FP7-215453

D3.1

Prototype of mass question answering

Dissemination level	Public
Contractual date of delivery	Month 12, March 31, 2009
Actual date of delivery	Month 12, March 31, 2009
Workpackage	WP3 Mass Intelligence
Task	T3.1 Mass question answering
Type	Prototype
Approval Status	Draft
Version	0.9
Number of pages	41
Filename	WeKnowIt-wp3-d3.1-20090402.doc

Abstract

This report describes services developed for the mass question answering prototype. Services where designed to address issues of information quality and its relevance in the Consumer Social Group scenario and improve resource annotation in the Emergency Response use case. They were developed and tested using the question answering system. This document includes description of five proposed services: local community detection based on tags, spam detection, latent topic detection and document categorization, topic-expert finder, and measuring answer quality. Each service is described in details in separate sections. Services described here were originally intended to be tested and included on LYCOS iQ system; currently they were only tested using historical data from this system.

co-funded by the European Union

History

Version	Date	Reason	Revised by
0.1	2009-03-02	Initial document	M. Janik
0.2	2009-03-03	Service: local community detection	S. Papadopoulos
0.3	2009-03-11	Service: Spam Detection in Question Answering Systems	G.Kalfas, S.Papodopoulos
0.4	2009-03-12	Service: Latent topic detection, expert finder	M. Janik
0.5	2009-03-16	Service: Answer quality measures	M. Janik
0.6	2009-03-19	Rewriting parts of different sections – language and style.	M. Janik
0.7	2009-03-25	Corrections according to internal review	M. Janik
0.8	2009-03-30	Final corrections.	M. Janik
0.9	2009-04-02	Updated after second review	M. Janik

Author list

Organization	Name	Contact Information
UoKob	Janik, Maciej	janik@uni-koblenz.de
CERTH-ITI	Symeon Papadopoulos	papadop@iti.gr
CERTH-ITI	George Kalfas	gkalfas@iti.gr

Executive Summary

Prototype of mass question answering consists of a collection of services designed for utilizing massive user input to improve different areas of the Collective Intelligence. They were developed and tested with the question-answering system. In the center of interest were the services responsible for ensuring the content quality, classifying the content, and providing related information to the given input. We support the following functionality with the five defined services in this package:

- local community detection based on tag network,
- spam detection,
- latent topic detection in a document,
- topic expert finder based on the document content,
- measuring quality of an answer with regard to the question.

Services described in this document originate from the analysis of the user requirements of the Consumer Social Group (CSG) scenario and Emergency Response (ER) use case. Quality assurance of the information provided to the system is one key issues both for the consumer use case and for the emergency use case. Services helping with providing appropriate annotations, like tags or categories, are directly applicable in the WeKnowIt scenarios. Implementation of the services is based on the information available in the LYCOS iQ Question Answering system, which was used as a testing ground. Despite utilizing specific data from the LYCOS iQ for testing and development of the services, their functionality is primarily designed to provide specific solutions for both use cases in WeKnowIt.

This document includes descriptions of the five proposed services, used methodologies, their implementation and usage.

Abbreviations and Acronyms

CSG	Consumer Social Group (scenario)
ER	Emergency Response (use case)
LDA	Latent Dirichlet Allocation
LYCOS iQ	Question and answering system from Lycos
NMI	Normalized Mutual Information
OWL	Ontology Web Language
OWL-QL	Ontology Web Language Query Language
OWL-DL	Ontology Web Language Description Language
OSGi	Open Services Gateway initiative
QA	Questions and Answers (Lycos iQ system)
RDF	Resource Definition Framework
XML	eXtensible Markup Language
SPARQL	SPARQL Protocol and RDF Query Language
WP2	Work Package 2 of WeKnowIt (media intelligence)
WP4	Work Package 4 of WeKnowIt (social intelligence)

Table of Contents

List of Figures

List of Tables

1. Introduction

The Web 2.0 provides an interactive platform that enables people to share knowledge, discuss, exchange ideas or create content that other users may find useful. In the CSG participants rely on the information provided to the system by other users to shape their actions and decisions. The users have to be able to verify whether the provided information or recommendation is valid for them. In ER scenario, trust and verification of information is even more crucial. Additionally, during the emergency citizens need all the support they can get from the system to automate at least parts of the tasks of providing and retrieving most relevant information. The existing mass of information in the system can be successfully utilized to achieve this goal.

The services described in this document where designed to address issues of information quality and its relevance in the Consumer Social Group scenario and improve resource annotation in the Emergency Response use case. For the development, testing and the source of the mass data for analysis we selected the LYCOS iQ system. These specific implementation of the services were partially tailored to fit into the LYCOS iQ platform, but the proposed functionality supports the requirements of the WeKnowIt use cases presented before.

Question and answer systems, like the LYCOS iQ, are the marketplaces of knowledge. They have their own specific characteristics of user interaction, information flow and design. On the other hand, user expect the same relevancy and trust in information, as in CSG or ER scenario. In the question and answering systems parties can ask questions and also provide answers to questions of other users. Users asking questions expect to receive relevant answers that also show high level of expertise. On the other hand, people that provide answers should answer questions mostly from their areas of expertise. To facilitate finding proper experts and targeting questions to the most appropriate users, topic hierarchies and groups of interest are introduced. Additionally, users posting questions can associate tags with the question to indicate which topics or areas of interest are relevant. When the question receives an answer, user would like to know how relevant it is to the posted question and how much the provided answer can be trusted.

Looking from the perspective of posting a question, users are interested in flexible solutions to provide such descriptions of the relevant topics, so people with appropriate expertise are able to easily find it. When reading a posted answer, users would like to know if it contains relevant information with regard to the posted question, and how trustworthy the provided content is. We propose the following services to address the presented issues and provide their short description below:

- local community detection,

- spam detection,
- latent topic detection,
- topic expert finder,
- answer quality measures.

Local community detection is a service that suggests related tags to the one entered by the user. By analysis of tag co-occurrences in the existing document corpora, we can create a network, or community, of related tags. When a user enters a tag into the system, the service can provide a whole community of relevant and related tags. In case of the resource upload in the ER use case, either users themselves or services from the WP2 responsible for content analysis can provide initial tags. The local community detection service can extend the initial tags and recommend user the most relevant annotations of the new content. Such support from the WeKnowIt system can significantly ease the uploading and annotating process, especially during the emergency when citizen are under strong pressure. In the context of the LYCOS iQ system, the service helps user assigning tags to the question that are most informative or most recognizable in certain topic.

Detection of latent topics in document helps users to go beyond the boundaries of pre-defined topic structures. Imagine a traveler searching for interesting things, which are related to the specific news or article she or he found. Officially assigned categories, like sports, travel or politics, may not be sufficient. An analyzed document can contain multiple topics, even some of which are not pre-defined in the system. Latent topics, which can be discovered in the existing document corpora by analysis of documents' content, can serve as an alternative to an existing topic hierarchy. The collective knowledge emerge from the content provided by users, and may be used either to locate the potential experts in given topic, or to further explore topics together with their underlying document structures, that are interested for the users.

Latent topic analysis in the document-author network of document corpora additionally reveals hidden structure of social groups that form around certain topics. It is especially valid for travelers' community, where multiple users contribute to a wide spectrum of subjects, but rarely form more formal interest groups. Such result from the mass analysis can be fed into the social services developed within scope of WP4 to analyze social interactions between users of the CSG.

Information quality is one of the important aspects for services providing any kind of information. In the CSG, the travelers expect to get relevant answers for their queries, answers that can be trusted. They also want to omit any kind of spam, which may be put by others to the system. A spam detection service is a natural realization of a first barrier for receiving only relevant information. In this implementation, the focus is on detecting low-level spam. After spam is filtered out, it is feasible to ask

about the relevance of the provided content with respect to the original query. In the context of the LYCOS iQ test case, user wants to verify how relevant is the provided answer to the given query. The presented service for answer quality measures tries to relate topics imposed by the question, with topics found in the answer. The answer relevancy is measured by checking which topics from the question and to which extent are covered by the provided answer. For this particular solution we use a general knowledge ontology to help capturing and defining topics.

The service for finding topic experts for a given document (or topic) can be applied both for queries and results. When a traveler from the CSG posts a query to the system, the service can be used to suggest a group of users which have expertise in the required areas and should be able to provide most adequate information. While verifying the quality of received results, user can check whether authors of the returned documents have the expertise in the areas covered by the content. Such information helps to verify the credibility of received information.

All five services introduced here are described in details in the subsequent sections of this document. Each of them addresses a specific issue raised during the analysis of use cases in WeKnowIt, and contributes into creating the overall Collective Intelligence of the system.

2. Service: Local Community Detection

Relational data are frequently represented in the form of networks. In that form, the entities of interest (e.g. users, images, questions, tags) are abstracted as the nodes (or vertices) of the network and the relations between them are represented by the edges of the network. Representing relations among entities of real-world systems frequently results in networks with complex structure, which can frequently be decomposed into meaningful modules (groups of network nodes) or **communities**, as they are frequently termed in the related literature (see Section 2.1). Communities can be valuable for understanding the inherent organization and processes of such networks.

To this end, we make available a Local Community Detection service within the D3.1 prototype. **The service can detect groups of closely related entities that are represented in the form of graphs**. The intended use for this service is the detection of topically related tags in tag co-occurrence networks (see Section 2.2). However, there are no particular assumptions made that would limit the service application to tag networks. Therefore, a series of other applications could be served, e.g. social networks, user generated content (e.g. flickr images), etc.

The service implementation builds upon the JUNG framework [1], a popular Java framework for representing and managing graph structures. The service performs community detection at a local level, i.e. it progressively scans the input graph starting from a seed node until the community around the given seed node has been formed. For that reason, the service is applicable to massive graphs since it scales with the size of the output community and the average degree of the contained nodes. Implementation details are provided in Section 2.3. The service has been evaluated by means of both artificial networks with synthetic community structure and the tag networks (English and German) formed by the tagging activities of users of the LYCOS iQ Question & Answers (QA) system. Section 2.4 provides an overview of the attained results.

2.1. Background

Community detection has lately received significant interest in the context of graph theory and complex network research. There is still no standard definition of what a community is; a popular one is the following:

A community is a subset of nodes in a network for which the number of edges connecting the community nodes with each other is larger than the number of edges connecting community nodes to the rest of the network.

In addition to this definition, a popular measure of the extent to which a set of nodes form a community was proposed in [2]. This measure is termed modularity and quantifies how much a given partition of a graph

into communities is different from a random partition of the same graph to the same number of communities. A whole class of methods has been presented thereafter which cast the problem of community detection as a modularity maximization problem. Alternatively, there have been numerous efforts to quantify network community-ness and to solve community detection by means of other methodologies. The survey papers in [3] and [4] provide a well-explained introduction and thorough treatment of the problem.

A class of methods has been recently proposed which rely only on the local graph structure around an input node in order to identify the community around it. These methods are particularly suited to solve this problem when the underlying network is very large. Examples of such methods are described in [5], [6], [7], [8] and [9]. The implementation of this service is also based on a variant of a local method, which was proposed in [10].

2.2. Usage

For the moment, the service is applicable to tag co-occurrence networks. For instance, within the LYCOS iQ QA system users tag their questions with one or more tags. In case two tags are used to tag the same question (i.e. they co-occur in the context of this question) a link is created between these two tags in the network. Tag and co-occurrence frequencies can be stored along with the nodes and edges of the network.

The service usage is straightforward. The service exposes a single method named "*findCommunity*" which accepts a tag string as an input and outputs the community, i.e. a set of tags, containing this tag. The resulting community is expected to contain topically related tags; thus, it can be used for (a) tag recommendation (given a single tag recommend a set of related tags), (b) tag similarity (compute the overlap, e.g. Jaccard coefficient, between the communities around two given tags), (c) topic profile construction (starting from one or more topic keywords create vectors of related tags-keywords).

In terms of responsiveness, one should expect that the service will quickly identify the community around an input tag. The response time depends on the size of the output community and the average degree of the contained tags, but is typically expected to be in the order of one second.

In the future, we plan to extend the service to also accept a list of tags as input. Currently, it is possible to emulate this by calling the service multiple times (once for each tag of the input list) and consider only the intersection of the resulting communities as the output community. Such usage, however, is discouraged, since it is computationally inefficient.

2.3. Implementation

The service builds upon the JUNG framework [1], version 2.0-beta. The basic principles underpinning the implementation are described in [10]. In summary, the community detection algorithm starts a local graph exploration process starting from the input seed tag. The exploration process is carried out by first visiting the tag's direct neighbors, then the neighbors of the neighbors and so on. Once a tag is visited, it is considered as a candidate for attachment to the community-under-construction; then, a **local bridging function** (see below) is calculated for the edge that led to visiting the candidate tag. This function quantifies the extent that this edge acts as a bridge between communities (i.e. inter-community edge). If the edge is found to be a bridge, then the candidate tag is considered as community boundary, i.e. it is not part of the community-under-construction and it further blocks the graph exploration process to continue through it. Further, nodes that act as **hubs**, i.e. they have very large degree, are considered as blocking elements to the exploration process (but they are included in the community if the edge that leads to them is not a bridge).

The local bridging function of an edge e_{st} between two nodes s and t is defined by the following formula:

$$b_L(e_{st}) = 1 - \frac{|N(s) \cap N(t)|}{\min[(d(s)-1),(d(t)-1)]},$$

where $N(s)$, $N(t)$ are the sets of neighbor vertices for nodes s and t respectively, and $d(s)$, $d(t)$ are their degrees. In practice, it was found that the second-order local bridging is more effective in detecting bridging edges. This is given by the following formula:

$$b_L^{'}(e_{st}) = a \cdot b_L(e_{st}) + (1-a)\frac{1}{|N(e_{st})|}\sum_{e \in N(e_{st})} b_L(e),$$

where a is a mixing parameter (with values between 0.0 and 1.0) and $N(e_{st})$ is the set of edges that are adjacent to e_{st}.

In the delivered implementation, two additional heuristics were incorporated in order to restrict the number of community members returned by the service. First, the graph exploration process stops at a 2-hop distance from the seed node, since it was found that tag networks are small-world (i.e. one can reach almost any node in the network in 2-hops). Second, when a node is considered for attachment to the community-under-construction, its co-occurrence frequency is checked against some threshold in order to ensure relevance to the existing community tags.

2.3.1. Deployment

Currently, the service has been packaged and delivered in the form of a Java library (jar). The library is bundled together with its dependencies (e.g. JUNG jars) and a short readme file describing how it can be used from command-line (for demonstration purposes). In addition, a tag network is provided (encoded in a custom text format) for testing purposes. The tag network was created from the SQL dump of the English LYCOS iQ database.

Soon (by M14 of the project), the service will be packaged as an OSGi module, suitable for deployment within the WKI runtime platform.

2.4. Evaluation

The evaluation of the implemented service has been carried out by means of both artificial networks with synthetic community structure and two real tag networks (English and German) created from the LYCOS iQ QA system. The synthetic community network generation process was introduced in [2] and is also documented in [10]. According to this a network of 128 nodes with an average degree of 16 is created. The network comprises four communities of equal size (32 nodes). Then, a percentage p_{out} of each node's edges are connected to nodes outside its community, while the rest of the nodes link to nodes of the same community. As a measure for community detection accuracy, we use the number of correctly classified nodes (F_C) used by [2] and the Normalized Mutual Information (NMI) that was introduced in [4] as a measure of community detection success. Table 1 summarizes the obtained results. The results indicate that the community detection accuracy achieved by the implemented service is at least as accurate as the one attained by a global (and thus much slower) state-of-the-art method.

	F_C		NMI	
p_{out}	WKI	GN	WKI	GN
0.01	100	100	1.0	1.0
0.05	100	100	1.0	1.0
0.10	100	50	1.0	0.86
0.15	99	50	0.98	0.86
0.20	74	50	0.84	0.86
0.25	24	0	0.56	0.02

Table 1. Community detection accuracy (WKI stands for WeKnowIt, i.e. our implementation and GN stands for Girvan-Newman which is described in [2])

Further insights into the performance of the implemented service were gained by using it on the tag network created from the tagging activities of the LYCOS iQ QA system. Here, only two examples from the English tag network are provided. Figure 1 depicts the community that was detected around the tag "computers" (the original community was larger and tags with low frequency were removed to render the image clearer). Figure 2 depicts the community that was created by providing the tag "history" as input to the service. By inspecting these communities, one can note that the tags belonging to them are topically related and therefore the service is suitable for the usage scenarios described in Section 2.2.

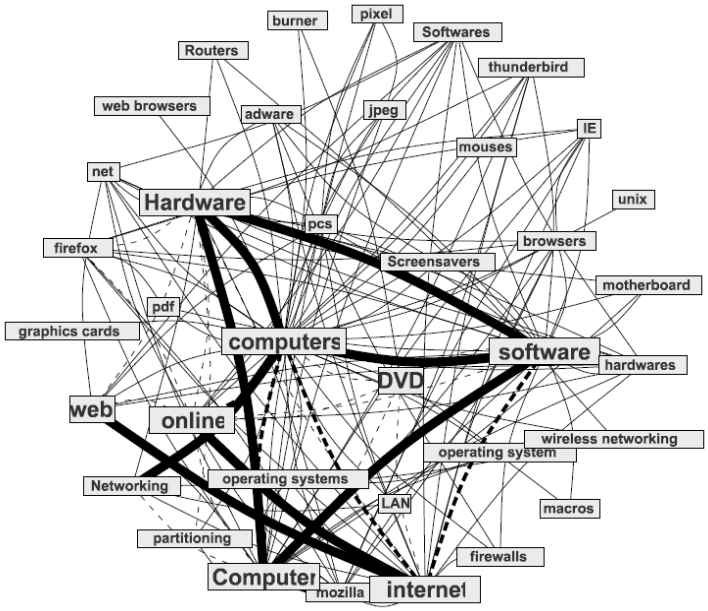

Figure 1: Tag community around tag "computers".

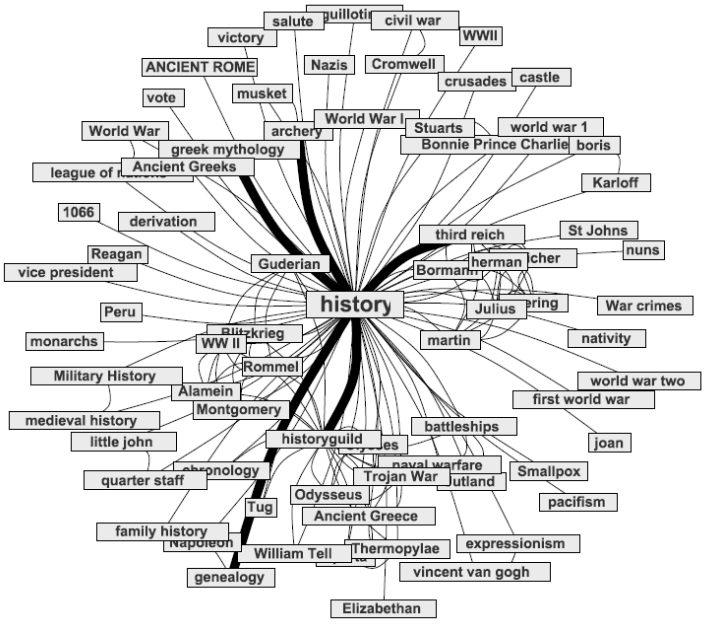

Figure 2: Tag community around tag "history".

3. Service: Spam Detection

Over the last years we have witnessed the emergence social web applications. One of the examples is the travel-related sites. Users, like in the CSG scenario, not only require to be able to book hotel or airplane ticket, but also expect from the system to provide recommendations or allow to post questions, discuss with other users. A recent development is that the publicly available services tend to evolve from topic-focused forums, which are usually visited by a limited number of people interested in a particular subject, to general purpose websites, where users can pose questions regarding any matter of interest to them. It is understood that the latter websites are visited and used frequently by what can be up to millions of users either asking or answering questions, making the moderation by the administrators a difficult task.

To this end, we make available an automatic Spam Answer Detection service tailored for the LYCOS iQ system and the available dataset within the D3.1 prototype. **The service can detect spam answers at the time that they are posted, thus relieving human experts to a great extent from the burden of moderation**. Mainly the nature of spam met in QA systems can be divided into two major concrete categories, namely the low-level (lexical) and the high-level spam (see Section 3.1.1). The spam detection process consists of several targeted filters arranged in a pipeline configuration (see Figure 3). Implementation and configuration details are provided in Section 3.2. The service has been evaluated with the use of a dataset that was formed by the users of the LYCOS iQ QA system. Section 3.3 provides an overview of the attained results.

3.1. Background

3.1.1. Spam categorization

The term "spam" is composite and versatile. In order to address the problem correctly and comprehensively, spam must be analyzed into categories [13][14]. Based on observations on the Lycos IQ dataset we found out that the spam could be effectively divided into two major spam classes, namely the low-level and the high-level spam.

As low-level or lexical spam we consider the following cases:

- Profane and vulgar terms or phrases in general.
- Illegal terms (i.e. "3fasdfasdd").
- Internet lingua (i.e. "LOL" or "u r" instead of "you are").
- Low quality grammar and syntax.
- Different language (i.e. Question in English, answer in Spanish).

High-level spam comprises the following cases:

- Plagiarism (copied & pasted from other source).
- Off-topic or totally irrelevant answers.
- Answers that contain commercial information or are of advertising nature.
- Ironic or mocking answers.
- Answers that suggest illegal actions (i.e. how to overcome copyright restrictions or how to smuggle merchandize through customs).

Obviously, a text extract could be characterized as both low- and high-level spam, i.e. these two broad spam categories are not mutually exclusive. According to the spam type different techniques are employed to detect it. In general, low-level spam is considered as easier to detect than high-level.

3.1.2. Low-level spam detection

In order to detect lexical spam a series of methods are used, ranging from simple list-based dictionary matching to *n-grams* language models. The most frequently applied approach regarding dictionary matching is the use of a profane terms list. Although this method is particularly easy, fast to implement and requires light processing, it comes with a series of disadvantages:

- Very easy to overcome. A malicious user can substitute a letter with a symbol, thus making the new word untraceable.
- It is language dependent, meaning the list must be populated from the start by a human moderator, for every language that the system supports.
- Usually the list takes a significant amount of time to reach a mature state in terms of completeness.

Due to the above the use of dictionary matching techniques is considered to be incomplete and can only be used as a first step to detect the obvious spam attempts.

A more advanced approach is the use of the ***n-grams* language models** [11][12]. N-gram models are a type of probabilistic model for predicting the next item in a sequence. N-grams are used in various areas of statistical natural language processing and genetic sequence analysis. An n-gram is a sub-sequence of n items from a given sequence. The items in question can be phonemes, syllables, letters, words or base pairs according to the application. An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram"; size 3 is a "trigram"; and size 4 or more is simply called an "n-gram". Some language models built from n-grams are "(n−1)-order Markov models". More concisely, an n-gram model

predicts x_i based on $x_{i-1}, x_{i-2}, \cdots, x_{i-n}$. In probability terms, this is equal to $P(x_i \mid x_{i-1}, x_{i-2}, \cdots, x_{i-n})$. By using a very large dataset of well-formatted text, we can calculate the probability of the transition between any two characters (bigrams) of the dataset's alphabet, thus calculating the plausibility of any given sequence of characters. An ideal dataset for this task is Wikipedia for a number of reasons:

- Vast amount of text containing diverse vocabulary and topics.
- Good quality of text due to its encyclopedic nature and constant moderation by the community.

Although the bigrams methodology as described above yields good results, we employ *Term Frequency (TF)* as an additional measure. For each term that is encountered in Wikipedia, a TF value is calculated, which stands for the times that this particular term is contained throughout the whole Wikipedia dataset. The TF measure prohibits the spam detection service to mark a low plausibility word as spam if it has a high TF value, thus acting as a safety net.

3.1.3. High-level spam detection

Detecting high-level spam can be a challenging task, especially in QA systems that deal with a large variety of topics. High-level spam detection focuses mainly around two problems: a) Is the answer correct regarding the specific question and b) Is the answer posed in a correct and formal way? In order to answer the first question the assumption is made that the correct answer already exists in text form somewhere on the web. Moreover, QA systems present a specific trait that distinguishes them from plain text. The text is given in pairs of questions and answers. The rule that these texts must abide is that they should semantically refer to the same topic. In theory if someone supplied the result of the semantic extraction of both the text contained in a question and its answer to a search engine, queries made from correct answers must have greater hit count than queries made from spam answers [15].

By taking advantage of the fact that questions are grammatically constructed in a finite number of ways the following solution for extracting the semantics (i.e. the most important nouns and accompanying adjectives or verbs) from the question text is proposed. With the use of a large dataset of questions we create a Most Frequent Terms list, regarding the specific language of the dataset. Since questions are mostly short texts by removing all the words that are contained in the Most Frequent Terms list and applying standard NLP analysis to the remaining terms we extract the nouns and adjectives that contain the main meaning of the question. Table 2 displays the top 50 words encountered in the English version of the Lycos iQ dataset.

On the other hand semantic extraction from the answer is more challenging and very difficult to achieve with high accuracy. Applying standard NLP analysis may not always lead to the desired results, and since a text contained in an answer can be indefinitely big, too large queries could occur, making them unsuitable for submission to a search engine. For that reason different combination algorithms are considered and evaluated at this moment, and will be presented in a future version of this document.

1	the	11	you	21	be	31	from	41	so
2	a	12	for	22	this	32	not	42	me
3	to	13	do	23	my	33	why	43	at
4	is	14	have	24	if	34	does	44	your
5	of	15	on	25	with	35	they	45	which
6	and	16	that	26	but	36	when	46	know
7	in	17	how	27	was	37	who	47	get
8	what	18	are	28	there	38	any	48	one
9	I	19	can	29	as	39	has	49	just
10	it	20	or	30	would	40	an	50	all

Table 2. Example of English Frequent Terms extracted from real QA dataset in descending order of frequency

The second question regarding the formality of a given answer is being evaluated in a totally different manner. Correct answers and facts should be presented with neutrality and objectivity [3]. The orientation detection mechanism builds upon the SentiWordNet framework [17]. SentiWordNet is a lexical resource for opinion mining. SentiWordNet assigns to each synset of WordNet [18] three sentiment scores: positivity, negativity, objectivity. By using SentiWordNet we can determine the ratio of the words that express negative or positive emotion and therefore are not objective. If this ratio exceeds a certain threshold the answer is going to be considered as low quality, and thus will be marked as possible spam.

3.2. Usage

For the moment, the service is applicable to any QA-like application. The service usage is straightforward. The service exposes two public methods named "*isLexicalSpam*" and "*isSpam*". The first one is used for detection of the low-level spam and accepts any given text as an input and produces a unique result out of three possible answers: a) SPAM (the text provided is spam), b) NO_SPAM (the text provided is appropriate) and c) SPAM_CANDIDATE (the system could not determine with sufficiently high confidence the nature of the text, and human moderators should draw their attention on it). The second service implementation is used for detection of high-level spam and accepts a Question and an Answer object

pair as input. Both objects hold all the vital information needed to fully represent a question and a responding answer posted in a QA system respectively (i.e. time it was posted, contained text, user's id etc.). In terms of responsiveness, one should expect that the service will provide very quick response times making it possible to be used in online environments, given that the libraries and objects required by the service will be already loaded in the memory of the host computer.

As mentioned in section 3.1 the service offers two public methods. Currently only the "*isLexicalSpam*" service is implemented, whereas the "*isSpam*" service is still under development and will be presented in a later version of this document.

The low-level spam detecting service consists of a series of four filters arranged in a specific order:

- StopWords Filter,
- Profanity Filter,
- Plausibility Filter, and
- Term Frequency (TF) Filter.

Each sentence is analyzed into a series of tokens, each one representing a word. Afterwards each token gets passed into the filtering system until it is determined whether it constitutes lexical spam or not. Figure 3 presents schematically the current four-filter layout.

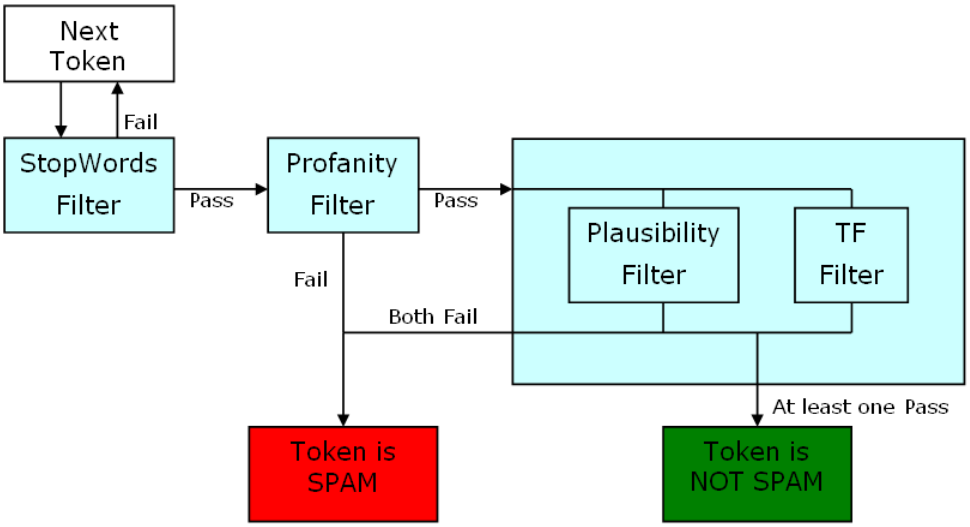

Figure 3. Low-Level Spam detection process

3.2.1. StopWords Filter

This filter implements a simple dictionary look-up based on a list that contains the most common words (stop words) of a specific language. If the word is matched against a stop word that is included on the list the

token is considered to be very common and is disregarded (illustrated as fail in Figure 1) due to the following:

- Reduction of processing time, since stop words are definitely not spam and so there is no need for further checks. This leads to a significant processing time gain due to the very frequent encounter of that type of words.
- Stop words are not considered to be useful regarding the evaluation of a sentence, since they are so frequent that they are equally contained in both spam and non-spam texts.

If the token is not contained in the stop words list, it is passed on to the profanity filter; else the system accepts the next token in line.

3.2.2. Profanity Filter

This filter is also based on a simple dictionary look-up. The tokens are matched against a list of profane words that is populated by human moderators. If the token is contained in the profane words list, it is automatically considered to be spam; else it is passed on to both the Plausibility and TF filters.

3.2.3. Plausibility Filter

The plausibility filter produces a number of type *double* in the interval [0,1]. This number represents the product of the probabilities of each consecutive bigram contained in the token. The probability of each bigram is calculated according to a bigrams graph. In the bigrams graph each node represents a character and each directed edge contains the frequency of the co-occurrence between the two nodes. If the plausibility of the token is greater than *TokenThreshold* the token is accepted. The *TokenThreshold* is calculated differently for every possible word length up to 50 characters, to ensure that larger words will not have a disadvantage against smaller ones. Internally the *TokenThreshold* value is calculated as a percentage of the median term in an ordered list containing the plausibilities of each term encountered in the Wikipedia dataset.

Construction and population of the bigrams graph

For this task we used the latest Wikipedia database dump offered by the WikiMedia project. At the time of development the latest database dump was released in 08.10.2008. In order to extract the bigram frequencies the following procedure took place (see also Figure 2 for illustration):

1. The Wikipedia articles were extracted in HTML-compliant format.
2. The article text was cleared, by removing every special annotation format and HTML tags.

3. All the text is parsed character by character resulting in a table which contains the frequencies of the transitions between all the characters contained in the text.

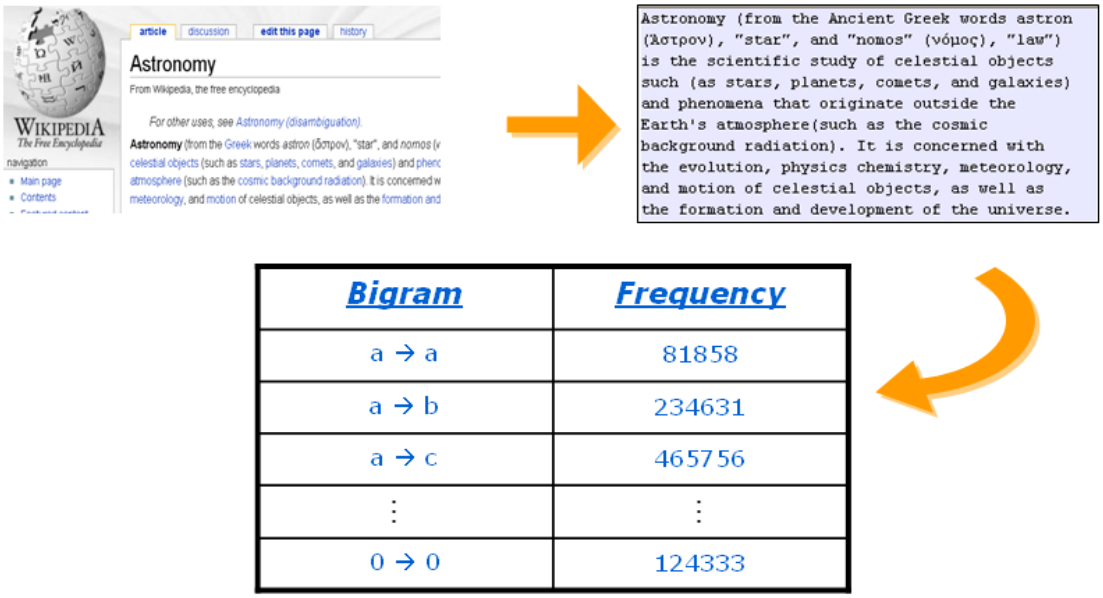

Figure 4. Low-Level Spam detection process

The graph is built using the JUNG framework [1], a popular Java framework for representing and managing graph structures.

3.2.4. Term Frequency (TF) Filter

As Figure 4 implies, the term frequency filter works in conjunction with the plausibility filter. It is being utilized as a safety net in cases where a legitimate but rare token appears to have a low plausibility score according to the plausibility filter. The latter could happen for example in the case of some large scientific words, whose bigrams sequence is not encountered often. In this case if this term appears numerous times in the Wikipedia dataset, it is considered to be legal and not spam.

3.2.5. Answer Evaluation

Each answer contains a number of tokens. After all tokens are passed from the filtering process as described above, each token is marked with a tag containing its spam verdict. If the ratio $\frac{SpamTerms}{TotalTerms}$ is greater than a predefined *SpamToleranceThreshold* value then the answer is considered to be spam. The correct value of *SpamToleranceThreshold* is further discussed in section 3.3.

3.2.6. Deployment

Currently, the service has been packaged and delivered in the form of a Java library (jar). The library is bundled together with its dependencies

(e.g. JUNG jars) and a short readme file describing how it can be used from command-line (for demonstration purposes).

Soon (by M14 of the project), the service will be packaged as an OSGi module, suitable for deployment within the WKI runtime platform.

3.3. Evaluation

The evaluation of the implemented service was based on the Lycos iQ dataset. The Lycos iQ dataset contains a large number of questions and answers posed by the public. Each answer is tagged with a flag denoting whether this answer is spam or not, and this decision was made by human moderators. By applying manual separation to the existing "Spam" answers set we created the "Lexical Spam" answers set, which is a subset of the original "Spam" set. During the tests we choose randomly 100 answers from the "Lexical Spam" and 100 answers from the "No Spam" category. Table 2 presents the results of the evaluation process.

Test Number	1	2	3	4	5	6	7	8	9	10
SpamThresholdTolerance	*0.12*	*0.11*	*0.10*	*0.09*	*0.10*	*0.10*	*0.10*	*0.10*	*0.10*	*0.10*
TokenThreshold	*0.35*	*0.35*	*0.35*	*0.35*	*0.40*	*0.45*	*0.50*	*0.55*	*0.60*	*0.65*
TP (True Detected Spam)	***61***	***64***	***64***	***66***	***69***	***73***	***74***	***76***	***78***	***78***
FP (Non-spam wrongly Detected as spam)	***0***	***2***	***3***	***11***	***4***	***5***	***6***	***7***	***9***	***11***

Table 3. Evaluation results

For definitions of *SpamToleranceThreshold* and *TokenThreshold* values please see Sections 3.2.5 and 3.2.3 respectively. Beginning at test #1 we can see that by setting *SpamToleranceThreshold* at 12% and *TokenThreshold* at 35% we gain good results with 61% of true spam being detected whereas no good answers where mistakenly caught as spam. By gradually reducing the *SpamToleranceThreshold* and increasing the *TokenThreshold* we gain better results regarding true spam detection, but on the other hand the system wrongly evaluates some legitimate answers as spam. At test #9 it can be observed that the true spam detection percentage has increased by 17%, but at the cost of a 9% increase in false spam detection.

4. Service: Latent Topic Detection

People perceive and use the web increasingly as a social medium that fosters interaction among people. Users on social sites communicate with each other, share experiences and knowledge, express opinions, form communities. In the question and answer system people share their knowledge by answering questions from a variety of topics.

The mass and variety of the incoming information can easily lead to situations, when users will not be able to find which questions are interesting for them, and which they can answer. To overcome this problem, apart from keyword search for questions and answers, systems offer free text tagging for identification of interesting topics together with a hierarchy of topics. However, a hierarchy of topics that is set by the system administrators is not always flexible enough to capture the variety and specificity of topics users would like to see.

Discussions, or questions and answers on specific subjects create informal groups of users that share similar interests; however no explicit relationships may exist between them. Social network analysis [19] has been used to analyze various online communities and studied the relationship and roles of network users. Nevertheless, social network analysis tools taken alone are not sufficient to identify online communities where no explicit links between users exist. To identify latent communities in social media where explicit links are not given, it is logical to use the concept of homophily theory. Homophily theory predicts that people are more likely to interact with individuals similar to themselves in respect to a variety of qualities and characteristics [20]. Actors contributing to common interest (e.g. the same topic) form an informal thematic group with no explicit ties between contributing members. Analysis of such homophily behavior leads to the discovery of relationships between actors - but only inferred relationships of closeness by interest which do not necessarily imply that the one actor is aware of the other. The discovered social characteristics and relationships between authors can be translated into a social network.

One of the approaches to discover informal groups of users that share similar interests is to detect latent topics in posted documents and to use this information for grouping users. It is specifically applicable in the areas where users exchange information about variety of topics. Travel-related sites, which are used in CSG can serve as very good example. People would like to know where to travel, what to do, what to see, whom to meet, and would like to address all these questions to the appropriate group of people. The spectrum of topics is so wide, that standard hierarchies and user groups may not be sufficient. Additionally, in this approach, topical groups of documents are discovered that can be successfully used to categorize new incoming documents. Such

categorization is complementary to the topical structure imposed by the system administrator. It can provide additional information about latent topics covered in a new document. It can suggest topically related documents that already exist in the system, or offer information about users that show expertise in the related topics.

Furthermore, the evolving interests of authors can also introduce changes in the underlying latent community. It has been observed that topics in social media exhibit spikes (sudden topics which are linked to current events or enjoying a limited-time interest) and chatters (more recurring, long term topics) indicating strong correlation [21][22]. Extension of the performed research in the discovery of latent topics includes the temporal analysis of topics and the underlying latent network evolution, changes in communities or identifying influential factors for such changes.

4.1. Method description

Proposed realization of latent topic detection is based on the Latent Dirichlet Allocation (LDA) [23].

Latent Dirichlet allocation (LDA) is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. LDA is a Bayesian multinomial mixture model and has become a popular method in text analysis due to its ability to produce interpretable and semantically coherent topics. In this model words are the basic analysis tokens. We associate documents with latent topics, using a learned multinomial distribution probability of words over topics.

LDA assumes the following generative process for each document d in a corpus D:

1. Choose $N \sim$ Poisson(ξ).
2. Choose $\Theta \sim$ Dir(a).
3. For each of the N words w_n:
   (a) Choose a topic $z_n \sim$ Multinomial(Θ).
   (b) Choose a word w_n from $p(w_n|z_n;\beta)$, a multinomial probability conditioned on the topic z_n,

where Θ is a multinomial distribution of topics, a is a parameter for Dirichlet distribution, and β is the parameterized matrix of word probabilities k x V, where $\beta_{ij}=p(w^j=1|z^i=1)$. Further details of the model and method are available in [23].

LYCOS iQ system consists of questions, associated answers and relevant tags assigned by the users. For each question posted in the system we construct a single document that is a combination of these three elements, which is later used to perform LDA analysis. Our problem space

can be factorized into users contributing to social media contents, homophily relations (closeness by interest) and social networks.

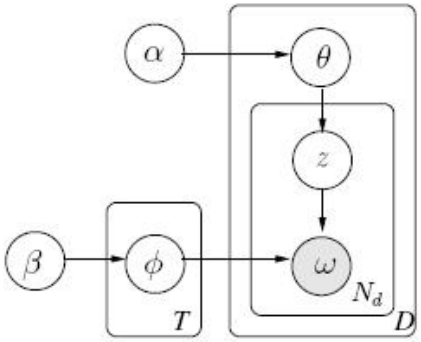

Figure 5 Topic-document-word LDA model.

In this model, each document can be described by a mixture of topics, and each word is characterized by probability distribution over multiple topics. LDA is a Bayesian network that generates a document using a mixture of topics. In its generative process, for each document *d*, a multinomial distribution *Θ* over topics is randomly sampled from a Dirichlet distribution with parameter *α*, and then used to generate each word, a topic *z* is chosen from this topic distribution, and a word, *w*, is generated by randomly sampling from a topic-specific multinomial distribution Φ_z. The robustness of the model is greatly enhanced by integrating out uncertainty about the per-document topic distribution *Θ*.

4.2. Usage

The service is designed to classify incoming documents (text) into latent document categories discovered by analysis of LYCOS iQ dataset. This dataset it used only for testing and demonstration purposes. The service functionality does not depend on it and can be directly applied into WeKnowIt system

The service usage is straightforward. The service exposes single method "*findLatentCategories*" that accepts a document text as an input and, optionally, which latent topic model to use for document categorization. Currently, we calculated and provided for this prototype 3 latent topic models. They are characterized by different levels of topic granularity: 500, 1000 and 1500 topics. Calculated models are based on source documents containing about 64000 questions and 202000 answers. If no model is selected, by default the model with 1500 topics is used.

Britney Jean Spears (born December 2, 1981) is a Grammy Award-winning American pop singer and entertainer. Raised in Kentwood, Louisiana, Spears first appeared on national television in 1992 as a contestant on the Star Search program, and went on to star in Disney Channel's television series The New Mickey Mouse Club from 1993 to 1994. In 1997, Spears signed a recording contract with Jive, releasing her debut album ...Baby One More Time in 1999 [...]

Topics			
204	**102**	**1201**	**47**
70%	32%	9%	2%
music	female	television	international
artist	actress	radio	national
pop	movie	noise	country
singer	award	commedy	birth
album	club	horror	men
record	singer	channel	woman

Figure 6 Example of topics relevant for document

Response time should be relatively quick, but it depends on the size of an input document (number of words) and size of the topic model used for categorization. All information required for the categorization must be transferred from the topic model stored in the database, as the whole model together with appropriate documents and user mappings is too large to completely fit in memory.

As a result the user receives a ranked list of topics that are relevant to the provided text. Each latent topic is identified by a unique ID and the topic model used for categorization. Each topic is defined as a set of relevant terms accompanied by their probability distribution over all topics. To get further explanation, the user can request a ranked list of terms defining the topic, ordered by their importance. In this implementation, as default behavior, the list is limited to the 20 most important terms. As latent topics differ across models, different topics can be assigned for the same document while categorizing it using different latent topic model s.

4.3. Implementation

During runtime the service uses pre-calculated LDA models stored in the database (in this implementation we use an Oracle database). It is due to the fact, that for the massive data, like the one used from the LYCOS iQ, calculations required to generate the complete model from underlying documents can take up to few days. Recalculation of new model for a new, single incoming document would be unacceptable for an on-line service. Therefore models are created off-line and later used in read-only mode for categorization purposes.

Off-line calculations of latent topic models use Latent Dirichlet Allocation implementation from LingPipe 3.7.0 library [24]. Text importers specific for the LYCOS iQ data and associated database exporters have been created locally. Calculation of latent topics executes in main memory and its results (partial and final) are stored in the database.

The latent topic detection service utilizes topic-specific multinomial distribution Φ_z to associate appropriate word probabilities w_p with a given topic. Each of the documents stored in the system has its own distribution of topic probabilities, calculated by the LDA model. Analyzed documents are treaded as bags-of-words, and the goal of the service is to find the

most relevant topics in the system. The natural approach is to use a Bayes classifier for finding relevant topics, as appropriate word probability distributions are already present in the LDA model. Categorization returns a list of related latent topics to the document content together with their probability. This describes the mixture and distribution of the latent topics relevant for the document.

4.3.1. Deployment

Currently, the service has been packaged and delivered in the form of a Java library (jar). The library is bundled together with its dependencies (e.g. LingPipe jars) and a short readme file describing how it can be used from command-line (for demonstration purposes). Currently, the service operates by reading required data from the database. Before proper execution, information like database access, username and password must be updated in the configuration (properties) file.

Additionally, an SQL dump of three different pre-computed LDA models derived from LYCOS iQ data is provided for testing purposes. To run this service, appropriate tables in the database must be created, so a classifier can access the model. We provide a script appropriate for the Oracle database, which enables creation of required tables and import of the data. Before running the script, information about database access like address, username and password must be updated in the configuration file.

The LDA models were generated from the LYCOS iQ data. The code for generating models off-line is included in the distribution. It contains a customized document importer, the LDA calculation package and the database upload module. Note, that for the used data, generation of a new model can take up to a few days.

Soon (by M14 of the project), the service will be packaged as an OSGi module, suitable for deployment within the WKI runtime platform.

4.4. Short evaluation

For testing and evaluation we used questions and answers from the LYCOS iQ dataset. It consisted of 64.745 questions, 202.424 answers, 45.432 unique tags that were used 134.601 times. For the LDA analysis for each question we created document that consisted of question text, all relevant answers and all associated tags with it. Such created document was used to generate LDA model.

Unfortunately, documents in the LYCOS iQ dataset do not have originally associated topics with them, so it is not possible to directly compare document groups associated with detected latent topics with the original document topics. Instead, we performed a manual evaluation of detected

latent topics and their practical usability for grouping documents into human-understandable categories.

This evaluation included naming latent topics and inspecting individual document associated with detected topics. Assigning human-readable names to topics (like: action movies, pop music) was to check if detected topics can be meaningful to future users and if using them in the system to categorize documents can be helpful for users.

We performed a short evaluation of 20 randomly chosen topics, and used 5 documents for each topic, which were the most representative, to check if the document content reflects the assigned topics. Each topic is represented by a distribution of words. In case of 17 analyzed latent topics, we were able to manually name the topic based on the most contributing terms. In case of 3 topics labeling was inconclusive, as terms covered multiple areas of interest. Inspection of individual documents only confirmed that content of documents reflected assigned topics. Such results encourage future research in automatically finding labels for detected latent topics based on their term distribution. We plan to investigate the possibility of using a general interest ontology for this purpose.

5. Service: Topic Expert Finder

When posting a question on the social site, we would like to find experts for relevant subjects to answer it, as we trust that these people should have the most appropriate knowledge. While reading a provided answer, we would like verify if its author is an expert in the core topic of the document. It indirectly ensures the trustworthiness of an answer. This service tries to address these issues. Goal of this service is to find the users with the best expertise on the topics defined by the content of a given document, whether it is a question or an answer.

We concentrate on finding users with expertise in topics defined by the content of a given document. A user is considered an expert in a selected topic if she or he publishes documents that are highly relevant for the topic. In the context of the LYCOS iQ system, this means providing answers that users evaluate as relevant and helpful for the posted question. Using the model of latent topics, we can indirectly associate user expertise with a specific topic with the quality of documents published by her or him, which are relevant to the latent topic.

5.1. Method description

The original LDA model uses only information about document and terms. Authors can be incorporated into this model by adding the additional layer and by associating them with documents, as it was proposed in [25]. In such a model, latent topics, and relations between authors and topics are calculated simultaneously. This is an extension of the original LDA method, which has been described in more details in the previous service (see Section 4).

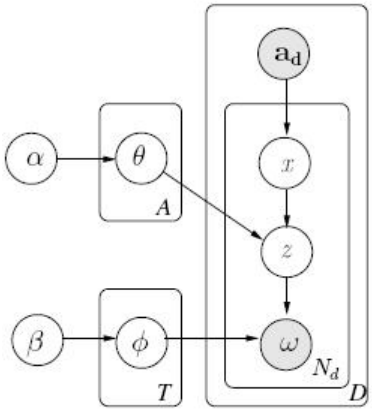

Figure 7 Author-topic LDA model.

The Author-Topic model is a Bayesian network, in which each author's interests are modeled with a mixture of topics. In its generative process for each document *d*, a set of authors, a_d, is observed. To generate each

word, an author x is chosen uniformly from this set, then a topic z is selected from a topic distribution Θ_x that is specific to the author, and then a word w is generated from a topic-specific multinomial distribution Φ_z. For improving LDA efficiency for author-topic model Gibbs sampling was used for parameter estimation [26].

5.2. Usage

The service is used for identifying most expert users in topics related to the content of a given document. The service is based on information learned from the LYCOS iQ dataset. This dataset it used only for testing and demonstration purposes. The service functionality does not depend on it and can be directly applied into WeKnowIt system.

The service consists of a single method "*findExperts*" that accepts document text as an input and, optionally, which pre-calculated latent topic model should be used for expert identification. Currently, we provide models including 500, 1000, and 1500 latent topics. Models are based on the source documents containing about 64000 questions and 202000 answers that were posted by over 23000 users. If no latent topic model is selected, by default model with 1500 topics is used to identify the expert users.

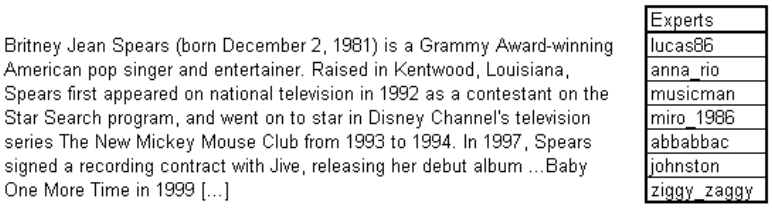

Figure 8 Example list of expert users

Response time should be relatively quick, but it depends on the size of an input document (number of words) and size of the used topic model. All information required for the topic detection and finding expert users must be transferred from the topic model stored in the database, as the whole model together with appropriate documents and user mappings is too large to completely fit in memory.

As a result the user gets a ranked list of experts that are relevant to the topics defined by the provided text. Each expert is uniquely identified by a login name, which is independent from the selected topic model. In this implementation, as default behavior, the list of experts is limited to the 20 most relevant users. As latent topics differ across models, different expert users can be suggested for the same document while using different latent topic models.

5.3. Implementation

During runtime the service uses pre-calculated LDA models stored in the database (in this implementation we use an Oracle database). It is due to the fact, that for the massive data, like the one used from the LYCOS iQ, calculations required to generate the complete model from underlying documents can take up to few days. Recalculation of new model for a new, single incoming document would be unacceptable for an on-line service. Therefore models are created off-line and later used in read-only mode for categorization purposes.

Off-line calculations of latent topic models use Latent Dirichlet Allocation implementation from LingPipe 3.7.0 library [24]. Text importers specific for the LYCOS iQ data and associated database exporters have been created locally. Calculation of latent topics executes in main memory and its results (partial and final) are stored in the database.

The topic expert finder service utilizes a topic-specific multinomial distribution Φ_z to associate appropriate word probabilities w_p with a given topic, and later find the most expert users for the topic. Goal of the service is to find authors that have the best expertise in the topics defined in the document. Experts are discovered by the analysis of documents they published in the relevant topics. Each of the documents stored in the system has its own distribution of topic probabilities, calculated by an LDA model. The analyzed document is treated as a bag-of-words, and we use a Bayes classifier for initially finding relevant topics in the model. To limit the search space, we consider only existing documents that belong to the most relevant latent topics discovered in the analyzed document. Using most relevant documents for given topics, expert authors are identified and provided to the user in the form of a ranked list. Note that identified users can be experts in multiple latent topics, as the document in LDA is modeled by a mixture of topics.

5.3.1. Deployment

Currently, the service has been packaged and delivered in the form of a Java library (jar). The library is bundled together with its dependencies (e.g. LingPipe jars) and a short readme file describing how it can be used from command-line (for demonstration purposes). Currently, the service operates by reading required data from the database. Before proper execution, information like database access, username and password must be updated in the configuration (properties) file.

Additionally, an SQL dump of three different pre-computed LDA models derived from LYCOS iQ data is provided for testing purposes. To run this service, appropriate tables in the database must be created, so a classifier can access the model. We provide a script appropriate for the Oracle database, which enables creation of required tables and import of the

data. Before running the script, information about database access like address, username and password must be updated in the configuration file.

The LDA models were generated from the LYCOS iQ data. The code for generating models off-line is included in the distribution. It contains a customized document importer, the LDA calculation package and the database upload module. Note, that for the used data, generation of a new model can take up to a few days.

Soon (by M14 of the project), the service will be packaged as an OSGi module, suitable for deployment within the WKI runtime platform.

6. Service: Answer Quality Measure

Information quality and its relevance is essential for providing strong building blocks for the Collective Intelligence. In every scenario, when user searches for information (like e.g. asking for activity recommendation in CSG), she or he would like not only to receive relevant information, but also of high quality.

In the question and answer-based systems users post questions and answers. Users who answer questions invest their time and knowledge, and in the ideal case, they want to provide the best possible answer according to their knowledge. However, the quality of the answer and its relevance to the originally posted question can vary. For people who post the question is most important to select the answer that contains the most relevant and complete information. This service would allow automatically measuring of the answer quality with relevance to a given question and could help in distinguishing good answers from the bad ones.

6.1. Method description

The proposed method is based on finding if and to what extent the answer covers topics (or categories) defined in a stated question. Due to the low number of terms in a question, the quality of an answer cannot be evaluated using only the co-occurrence of terms in both documents. To overcome the sparsity of data, we propose to use a general purpose ontology to identify categories with the help of recognized named entities in the document. Using an encyclopedic-like, general knowledge-based ontology such as the one based on Wikipedia, allows us to recognize entities from numerous domains. As a result, we can extract categories assigned to these entities in the ontology and use them as expected question categories.

On the other hand, we can utilize the same ontology and approach presented in [27] to detect the most important categories that describe the document content. The proposed approach relies on using the ontological knowledge to convert an unstructured text document into a semantic graph, and later find categories in the ontology that closely describe the constructed graph. In terms of coverage of the entities in the graph. Such ontological categories cover areas defined by related named entities in the graph, and as a result, they describe topics of the original document. The most important categories should focus on the most important discovered named entities (core entities), cover a significant portion of the created graph, and be rather specific than general.

To illustrate the method we present the category extraction process on a sample document:

> **Barack Obama** won the **presidency** and made history on a steady message of bringing change to a country that has been hungering for it.
>
> The first **black** to be elected **U.S. president**, **Obama** ran a nearly flawless **campaign** and proved to be a cool **customer** under fire on the **campaign** trail and during three head-to-head **debates** with defeated **Republican** **John McCain**.
>
> He ran for **president** during the best possible conditions for a **politician** running against the incumbent **party** -- a country saddled with two **wars** and teetering on the brink of a **recession**, including a Wall Street **meltdown** that amounted to an **October surprise**. […]
>
> **Obama** linked **McCain** to **Bush** at every turn, even though the two **Republicans** are not particularly fond of each other and **McCain** had broken with **Bush** on several important issues. It did not **matter**. **Obama** had only to keep bringing up the fact that the Arizona **senator** had voted with **Bush** 90 percent of the time.
>
> **Obama** was able to straddle the middle ground, managing to escape **McCain**'s attacks on his **liberal** **voting** record as a **senator** from Illinois, and raised vast sums of money for **television advertisements** that essentially turned him into an **iconic figure**. […]

Detected named entities from the ontology are underlined. Here, using the background information from the Wikipedia ontology, we can induce relationships among them. This leads to the creation of an initial semantic graph, which is later analyzed, so non-essential entities can be removed. The remaining entities (bold underlines) are the important part of the graph and create a connected component. Such entities are semantically related in the ontology and represent the major areas covered in the document content. They can be seen at the dominant thematic graph of the document, shown in Figure 9.

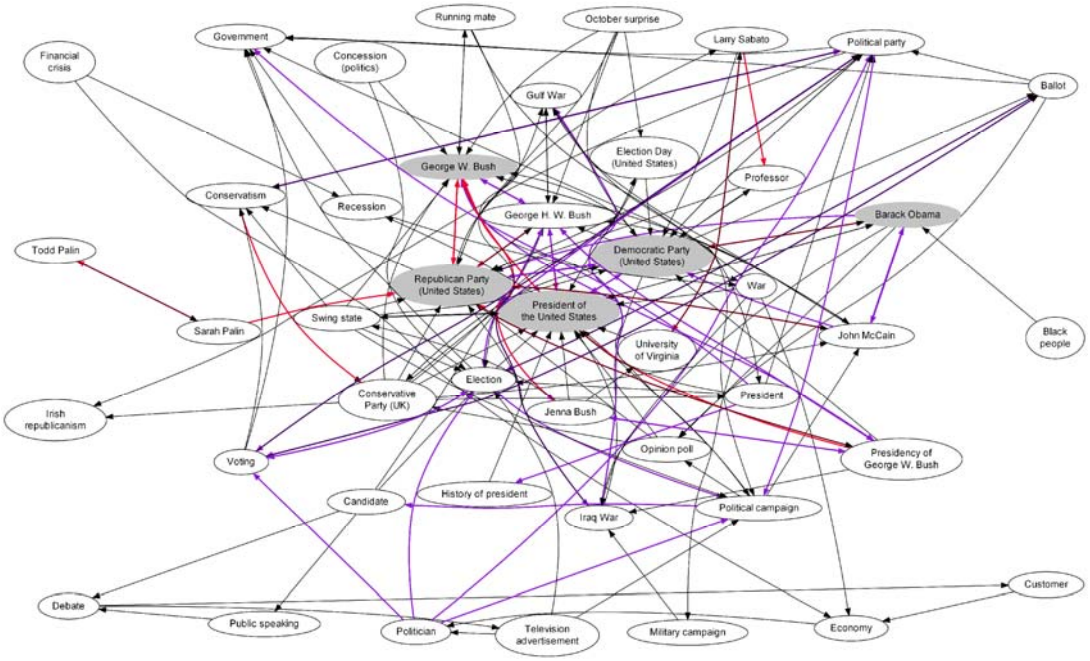

Figure 9 Thematic graph of a sample document

Analysis of the created thematic graph, especially concentrating on core entities (most central, most important) allows assigning categories that provide the best explanation and description of the graph. The top 5 assigned categories for the above graph, as defined in the ontology, are presented below:

- Presidents of the United States
- Presidency of the United States
- Political Parties in the United States
- Bush family
- Republican Party (United States)

Comparison of categories initially assigned to the question and discovered from an analysis of the answer content, measures how well answer covers topics defined by the question.

6.2. Usage

This service compares overlap of categories assigned to a question, with categories discovered from the answer. The service uses a RDF ontology based on Wikipedia as a source of named entities, relationships, categories and the category hierarchy.

The service consists of a single method "*evaluateAnswerQuality*" and accepts two text parameters: question and answer. Each text is analyzed separately to extract the most important categories.

As a result, the user gets a float value from 0 to 1, which represents the quality of the answer with respect to the given question. It is calculated as percentage of categories defined by the question that are covered by the answer.

6.3. Implementation

The initial implementation uses BRAHMS system [28] for ontology storage and querying. We use the RDF ontology created from the English version of Wikipedia, using a slightly modified DBpedia approach [29]. For the need of better entity recognition and construction of the semantic graph, we have created separate named relationships to distinguish among the direct names of entities (page names in Wikipedia), redirections (redirection pages), and entity names included in the disambiguation pages (disambiguation and contextual information in entity name). Additionally, we simplified the relationship scheme for some templates.

In the current version the service runs using a pre-calculated snapshot of the Wikipedia ontology and utilizes the algorithm described in [27] for category identification. Categories are identified separately for each input text. The time required for analysis of each text depends on its length and

number of identified named entities in it. Expected runtime for a short document (few Kb) is in order of a few seconds.

The final result representing answer quality is computed as percentage overlap of initial categories assigned to the question with the categories discovered within the answer.

6.3.1. Deployment

The service is originally written in C++ for *nix platforms. Complete source code together with required libraries is packed in tar.gz file. To install it on new machine, code must be unpacked, configured, compiled and locally installed. Only standard development tools are needed, like g++ and make utility. Additionally, a new snapshot of Wikipedia must be calculated from RDF files, so it is compatible with a selected architecture (32/64 bit, little or big endian). Unfortunately, the snapshot created by BRAHMS is not always portable between different systems and architectures. BRAHMS provides speed for accessing and searching statements at the cost of portability.

Currently we are working on rewriting the whole package to Java and using Sesame [30] as ontology storage with database backend. Such component will be fully portable and offered as a complete jar library. Java version of the service will be ready soon (expected M14 of the project) and packaged as an OSGi module, suitable for deployment within the WKI runtime platform.

7. Conclusions

In this document we presented five services relevant for the mass question answering prototype. All of them were motivated by the specific user needs extracted from the ER and CSG scenarios in WeKnowIt. The presented services were implemented and tested using the LYCOS iQ dataset. All of the services are able to operate as stand-alone, but specifically the local community detection and the latent topic discovery services were designed to cooperate with services from WP2 and WP4, respectively. Presented services include:

- local community detection,
- spam detection,
- latent topic discovery,
- topic expert finder,
- answer quality measures.

Description of each service includes short observation and motivation, explanation of used methods and algorithms, usage instruction, and details about the provided implementation.

8. References

[1] JUNG, Java Universal Network/Graph Framework, http://jung.sourceforge.net/

[2] Newman M. E. J., & Girvan M., *"Finding and evaluating community structure in networks"*, Physical Review E, Vol. 69, 026113, 2004.

[3] Fortunato, S., & Castellano, C., *"Community Structure in Graphs"*, Tech. report, **arXiv:0712.2716**, December 2007.

[4] Danon L., Diaz-Guilera A., Duch J., & Arenas A., *"Comparing community structure identification"*, Journal of Statistical Mechanics: Theory and Experiment, P09008, 2005.

[5] Bagrow J.P., & Bollt E.M., *"Local method for detecting communities"*, Physical Review E, Vol. 72, 046108, 2005.

[6] Clauset A., *"Finding local community structure in networks."* Physical Review E, Vol. 72, 026132, 2005.

[7] Palla G., Derenyi I., Farkas I., & Vicsek T., *"Uncovering the overlapping community structure of complex networks in nature and society"*, Nature, Vol. 435, pp. 814-818, 2005.

[8] Luo F., Wang J. Z., & Promislow E., *"Exploring Local Community Structures in Large Networks"*, Proceedings of the IEEE/WIC/ACM international Conference on Web intelligence. IEEE Computer Society, Washington, DC, pp. 233-239, 2006.

[9] Bagrow J.P., "Evaluating local community methods in networks", Journal of Statistical Mechanics: Theory and Experiment, P05001, 2008.

[10] Papadopoulos S., Skusa A., Vakali A., Kompatsiaris Y., & Wagner N., *"Bridge Bounding: A Local Approach for Efficient Community Discovery in Complex Networks"*, Tech. report, **arXiv:0902.0871**, February 2009.

[11] Christopher D. Manning, Hinrich Schütze, Foundations of Statistical NLP, MIT Press: 1999. ISBN 0-262-13360-1.

[12] G. Salton, A. Wong, and C. S. Yang (1975), "A Vector Space Model for Automatic Indexing," Communications of the ACM, vol. 18, nr. 11, pages 613–620.

[13] Jakub Piskorski , Marcin Sydow , Dawid Weiss, Exploring linguistic features for web spam detection: a preliminary study, Proceedings of the 4th international workshop on Adversarial information retrieval on the web, April 22-22, 2008, Beijing, China

[14] Zoltán Gyöngyi, Hector Garcia-Molina: Web Spam Taxonomy. AIRWeb 2005: 39-47

[15] F. Menemenis, S. Papadopoulos, B. Bratu, S. Waddington, and Y. Kompatsiaris. "AQUAM: Automatic Query Formulation Architecture for Mobile Applications". In Proceedings of the 7th international Conference on Mobile and Ubiquitous Multimedia (Umea, Sweden, December 3 - 5, 2008). MUM '08, ACM, New York, NY.

[16] V. Hatzivassiloglou and K. R. McKeown. "Predicting the Semantic Orientation of Adjectives". In Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the European Chapter of the ACL, pages 174–181, Madrid, Spain, July 1997. Association for Computational Linguistics.

[17] Andrea Esuli and Fabrizio Sebastiani, "SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining", In Proceedings of LREC-06, 5th Conference on Language Resources and Evaluation, Genova, IT, 2006, pp. 417-422.

[18] F. Christiane , "WordNet: an electronic lexical database" - Cambrige, MIT Press, Language, Speech, and Communication, 1998

[19] S. W. Carrington P.J., J. Scott. "Models and Methods in Social Network Analysis." Cambridge University Press, 2005.

[20] M. McPherson, L. Smith-Lovin, and J. M. Cook. "Birds of a feather: Homophily in socialnetworks." Annual Review of Sociology, 27(1):415-444, 2001.

[21] D. Gruhl, R. Guha, R. Kumar, J. Novak, and A. Tomkins. "The predictive power of online chatter". In KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pages 78-87, New York, NY, USA, 2005. ACM.

[22] [9] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. "Information diffusion through blogspace". In WWW '04: Proceedings of the 13th international conference on World Wide Web, pages 491-501, New York, NY, USA, 2004. ACM.

[23] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent dirichlet allocation". J. Mach. Learn. Res., 3:993-1022, 2003.

[24] LingPipe - suite of Java libraries for the linguistic analysis of human language, http://alias-i.com/lingpipe/

[25] Steyvers, M., Smyth, P., Rosen-Zvi, M., Griffiths, T.L. "Probabilistic author-topic models for information discovery." Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 306-315, Seattle, USA, August, 2004.

[26] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. "An introduction to MCMC for machine learning." Machine Learning, pages 5-43, 2003.

[27] Janik, M. and Kochut, K.J., "Wikipedia in Action: Ontological Knowledge in Text Categorization", Second IEEE International Conference on Semantic Computing, ICSC 2008, Santa Clara, CA, USA, August 2008

[28] Janik, M. and Kochut, K.J., "BRAHMS: A WorkBench RDF Store And High Performance Memory System for Semantic Association Discovery." in Fourth International Semantic Web Conference (ISWC 2005), (Galway, Ireland, 2005).

[29] Auer, S. and Lehmann, J., "What have Innsbruck and Leipzig in common? Extracting Semantics from Wiki Content." in European Semantic Web Conference (ESWC'07), (Innsbruck, Austria, 2007), Springer, 503-517.

[30] Broekstra, J., Kampman, A. and Harmelen, F.v., Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. in International Semantic Web Conference 2002, (Sardinia, Italy, 2002).