A Structuralist Approach for Personal Knowledge Exploration Systems on Mobile Devices

Stefan Bordag, Christian Hänig, and Christian Beutenmüller

Abstract We describe the reasons and choices we made when designing an architecture for a multilingual Natural Language Processing (NLP) system for mobile devices. The most tangible limitations and problems are limited processing power of mobile devices, strong influence of idiolect (or generally personal language usage differentiation between individual users in their personal communication), effort required to port the NLP system to multiple languages, and finally the additional processing layers required when dealing with real-world data as opposed to *controlled* academic set-ups. Our solution is based on a strict differentiation between server-side preprocessing and client-side processing, as well as maximized usage of unsupervised techniques to avoid the problems posed by personal language usage variations. Hence it represents an adequate combination of solutions to provide robust NLP despite all these limitations.

1 Introduction

Natural Language Processing is a field that has evolved dramatically over the past fifty years. Some of the early core areas of study were syntax, morphology, and semantics of languages. The main goal is to provide methods to let computers do at least some of the work that us humans do with information embedded in unstructured data.[1] By now there is a wide range of use cases for which at least practically acceptable solutions exist, such as Machine Translation, Information

[1] Assuming the dichotomy between structured and unstructured data to be loosely defined as "information or pieces of data stored in explicit relations with each other (cf. relational data bases) to be structured data and information stored without explicit relations within unanalyzed texts to be unstructured". Another possible definition is that "any data base is structured if it is possible to use a simple and precise query system which guarantees to retrieve a particular piece of information if it exists". No current system is able to achieve that fully with textual information.

S. Bordag (✉) • C. Hänig • C. Beutenmüller
ExB Research & Development GmbH, Seeburgstr. 100, 04103 Leipzig, Germany
e-mail: bordag@exb.de; haenig@exb.de; beutenmueller@exb.de

© Springer International Publishing Switzerland 2014

C. Biemann, A. Mehler (eds.), *Text Mining*, Theory and Applications of Natural Language Processing, DOI 10.1007/978-3-319-12655-5_6

Retrieval, Spell Checking, and Speech processing, amongst others. Most of these approaches are based on explicitly distinguishing individual languages. This makes it possible to build dedicated modules for each language such as for English versus a module for Spanish or French, etc. It is often assumed implicitly that a module built for English will be more or less uniformly valid for the language and all possible texts written in it. Yet it is obvious that there are distinguishable subdomains such as medicine versus technical documents within languages. Accordingly, much of the effort in NLP goes into solutions to the domain dependency problem to either reduce that dependency or find easier ways to port a solution from one domain to another.

In its entirety, any approach based on building language modules can be described as top-down, because at the core it will be about some particular language or language family. It splits the problem of building an NLP system top-down into solutions for particular languages of a language family and finally for domains inside a language. But what would a bottom-up approach be in this case? The most reasonably extreme form of that would be to treat the production and perception of language by a single person as distinguishable from that of any other person, which is very close to the one sense per discourse idea, but not as extreme.

A single person typically speaks one or more languages, is a specialist in one or more subdomains of each language she/he speaks, and also uses words, expressions, and even syntactic constructs in a highly idiosyncratic way. So much so, in fact, that it is even possible to detect original authorship of texts purely based on word frequency differences with a certain degree of confidence [33], distinguish individual persons based on their language production patterns even within the same location, profession, and topics [5], or conduct linguistic forensics pertaining to individual language usage [31]. The idiolect, the difference in language use amongst individual language users, may not seem very large. However, typical inter-annotator agreement figures show that these differences are far from negligible. For example, [21] show correlation scores ranging from 0.71 to 0.84 (Kappa) between human annotators while [11] show that even a task such as the detection of chemical Named Entities (which appears relatively easy) achieves 93 % F-score agreement between human annotators.

The relatively recent rapid adoption of new communication channels such as short messaging services (SMS) has brought about entirely new interaction limitations, which in turn spawned entirely new language domains, which additionally interact with existing domains. For example, lawyers and researchers are likely to exhibit different kind of language usage when they use SMS on smartphones or email on PCs. This leads to a further dispersion of personal language usage into larger numbers of different subdomains, cross-domains, and complex domain interaction phenomena. So much so that trying to build domain specialized NLP solutions for each of these possible interactions becomes more and more intractable. But is there any other way? Is it even possible to think of a solution that solves or circumvents these problems? In order to tackle these and other problems related to building NLP solutions for mobile devices, we formulate the following hypotheses and show that incorporating experimental results and observations arising from

these hypotheses improves user-centric performance levels of our NLP solution significantly.

Language usage variation hypothesis We assume that language usage between different users differs so strongly that any additional mechanism of adapting to user-specific language usage will easily outperform any pre-built NLP systems that do not adapt.

Limited language usage variation hypothesis We assume that the language usage variation is stronger (weaker) in some areas of language. Hence it makes sense to preprocess some parts of the language models on a large corpus on a server and only compute the more variable parts directly on-device.

Principle of minimal language dependence We assume that a solution to an NLP problem which requires substantial professional input[2] will be inferior to a solution which provides slightly inferior results in an evaluation but requires significantly less or no professional input on real-world data in order to adapt it to a new language or domain.

Additional processing layers Real-world data contains many more structural levels compared to theoretical models of language usage which often focus on morphemes, words, sentences, paragraphs, and larger text units. We assume that any system that provides even basic support for dealing for additional levels (e.g. lists, tables, log files) will provide noticeably improved quality levels on real-world data.

Our solution is a combination of traditional NLP methods with structuralist methods as extensions of statistical algorithms and in some occasions simple word lists. This approach extends many of the more basic approaches described in [24]. We particularly want to show that such solutions can be built in a sufficiently efficient way so that they can even be employed on the weakest possible computing devices, that is, mobile devices. One reason to do this is that in the past, excessive computing power requirements of structuralist approaches have been used as a counterargument—as we think—erroneously.

1.1 The Structuralist Approach and Personal Data

In structuralism signs (e.g. words) get their meaning from their relationships and contrasts with other signs [12]. Related to this, an unsupervised approach is loosely defined as a solution to an NLP problem that relies on finding structure within the input text on its own, rather than assuming structure through pre-annotated training data or explicit rules and assignments [23]. Such algorithms are based on measuring the observable patterns of usage of, for example, words with other words. An unsupervised part-of-speech tagger, for example, clusters words according to

[2]Input by people educated in a relevant field of linguistics.

their syntactic usage as measured by occurrence with other nearby words, instead of assuming rigid categories such as nouns or verbs. This typically requires a large amount of raw data in order to produce good results, and there is a logarithmic dependency between data size and quality, as observed in Chap. 3 in [6]. Hence, the first step is to assess what kind of input data can be used as input and whether it is sufficient. Personal data can be found in and between the realms of structured and unstructured data. Examples for structured personal data include:

- address books across various device/application combinations such as email client, mobile phone, or notebook
- calendar applications
- equipment or book rental lists in custom data base designs (often, this is given just by a single Excel table)
- music and video files
- bookmarks of favorite websites and histories of visited websites

Beyond that, a large bulk of real information is stored in unstructured data (or semi-structured data):

- documents the user wrote or received (note that usually information about the creator of a particular document does not exist explicitly)
- emails the user wrote or received
- short messages the user wrote or received on a multitude of devices (mobile phone, notebook, PC, various websites such as Facebook).

The level of detail available in personal data from both data sources is already staggering. With enough personal data, a highly accurate map of a given user's daily life can be achieved. No current systems are able to come even close to enabling the user to work with all her/his own knowledge in a seamless, fully integrated way. The main obstacles that need to be overcome include, among other things, the following ones:

- The individual structured data bases are scattered among a great variety of completely different and incompatible non-standardized custom data bases.
- There exists no software that is able to reliably extract clean bits of knowledge from all the unstructured data that the user has. There are only large-scale projects such as NELL[3] [7], or Google knowledge graph[4] or IBM Watson[5] [15] which cannot be applied on the personal data and needs of a single user. Other solutions such as Apple's Siri, Google's Google Now are typically server based and do not possess capabilities yet to analyze the unstructured data of the user.

But what kinds of real-world use cases would such a software enable. Is it possible to solve some of these use cases with existing technology?

[3] rtw.ml.cmu.edu/rtw, retrieved on 24.03.2014.

[4] www.google.com/insidesearch/features/search/knowledge.html, retrieved on 24.03.2014.

[5] www-05.ibm.com/de/watson, retrieved on 24.03.2014.

1.2 Mobile Devices

Natural language processing on mobile devices poses a particularly hard combination of challenges, such as extremely limited computing power or very limited interaction with the device due to small screens and tiny virtual keyboards. Excluding spoken language from the scope of this contribution still leaves us with an immense variety of communication types such as SMS, emails, and web pages, amongst others. The amount of data available for each of these types will be small compared to what computational linguists are used to work with, but not too small to consider statistical approaches. It is reasonable to expect a single user to handle on the order of a thousand SMS, emails or visited websites per year. Additionally, each of these sources of different text type poses problems which, at first, seem to be peripheral from the point of view of traditional NLP.

For example, SMS messages feature and encourage strongly abbreviated language patterns. Emails contain huge amounts of seemingly non-linguistic components, such as signature text parts, quoted email parts, tables, ASCII art, emoticons etc. Websites typically devote very small areas for content proper and use most of the screen estate for ancillary data such as menus, advertisements, and (un)related content. When viewing emails or websites, the human eye quickly catches all relevant parts and discards irrelevant parts even quicker. However, simulating even this task as an algorithm proves to be surprisingly difficult [28] and is only solved using indirect tricks, such as comparing different websites from the same provider with each other and tracking the similar parts.

Finally, mobile devices are sold and used on a global scale now, so any NLP solution will also have to be useful for a wide variety of languages, from English through Finnish to Korean and Japanese. This means that even very basic traditional components such as tokenizers, sentence boundary detectors and part-of-speech taggers can become obstacles in some cases. It also means that the effort of adapting a NLP system to a new language must be as cost-efficient as possible in order to make it feasible to cover several dozen languages.

2 Our Solution

Considering all the problems and limitations described above, we have built and tested in a commercial product[6] a solution which offers a robust NLP capability for a variety of use cases, including information extraction, text similarity measurement, and various classification tasks. This solution is a hybrid system including rule-based parts as well as purely statistical parts. The system is split into two parts, one for pre-processing on the side of the server and a second for

[6]The product is an email client with enhanced NLP capabilities; see http://mailbe.at.

on-device computation. We employ rule-based systems for basic NLP tasks such as tokenization or sentence boundary detection as well as for tasks of information extraction system as, for example, time extraction. We further employ rule-based systems for detecting macro-structures such as detecting quoted text, bulleted lists and tables. We use self-learning classifiers for signature detection in emails, website scraping, email classification and other on-device learning components. We employ server-side bootstrapping algorithms for our generalized named entity recognition framework and client-side trained and compressed models. Finally, we use entirely unsupervised solutions for text similarity estimations, text summarization, inducing semantic networks and other components which are strongly influenced by personal language variation. Some of the learning components, in particular all classifiers, run exclusively on the mobile device. Some components, such as the pre-computation of a part-of-speech tagger model (clustering and training) run on the server and produce compressed language models for client-side deployment.

2.1 Pledge for Additional "Language" Layers

When developing NLP solutions, one of the most critical steps is to introduce an evaluation framework which measures success and error rates. Our main finding of the study reported here was that the primary source of errors is not partial intelligence in the NLP solution (even if it is just a baseline solution) but that most errors are introduced by applying the solution in the wrong way or on the wrong data. For example, even a very simple sentence boundary detection algorithm usually produces very few mistakes (typically around abbreviations). This is demonstrated, for example, in [27] which show that the difference between the best and worst system is only 1,5 % (error rate) on the Brown corpus. However, when applied to the raw text of an email, it is not uncommon to see that error rates jump up to nearly 100 %. This is because an email may contain a single sentence and a long log printout with lots of dots scattered around (none of which signify sentence boundaries), and no dots at the end of each log entry (which would constitute sentence boundaries). In an academic setting, it is easy to discard such examples as *invalid* or *unfitting data*. When it comes to real-world applications, it is impossible to use such an argument because real-world applications need to detect and make automatic decisions whether to use or ignore data of any kind. Taking into account all of these issues and challenges, the architecture of our NLP system for mobile devices can be seen in Fig. 1. The basic idea underlying this architecture is to design in a way that the traditional NLP modules (such as time and named entity recognition) are applied only after extensive pre-processing steps. These steps detect text types and blocks, and apply the analytic algorithms only on data where and when it makes sense. This approach reduces error rates to levels comparable to the ones reported in the literature and also keeps the error levels relatively constant.

Another aspect of our architecture is that it divides between server-side processing of language models and client-side processing for auxiliary modules. There

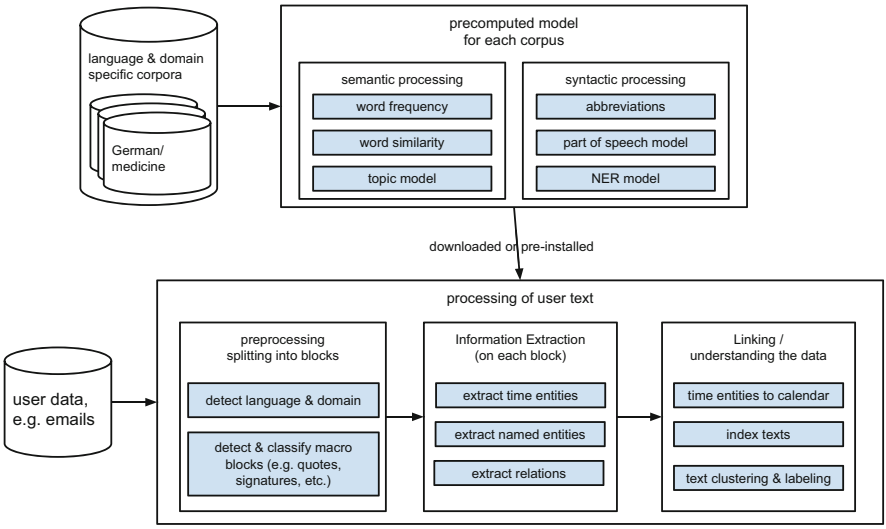

Fig. 1 An overview of the architecture of our NLP solution for mobile devices

are some algorithms that require a huge amount of input data in order to deliver acceptable results. Examples of such algorithms include part-of-speech tagging [29] and the differential analysis used for key word extraction [39]. On the other hand, there are algorithms that require relatively little data. However, they expect data which fits the problem very well. A good example of that is signature detection (based on observing similarities between emails from the same sender) or quote detection (where a helpful feature is whether some part of text has been seen previously) (cf. [8]).

In the following two sections we show how different parts of our architecture can be combined to achieve optimal performance levels. The next section describes the influence of the various parts and limitations of a mobile device on text similarity measurement followed by a detailed section on information extraction.

3 Text Similarity Measurement

Measuring semantic text similarities is a topic of NLP with a large variety of use cases. Finding semantically related content is an extension of the core use case of search in information retrieval. We can simply reformulate the typical information retrieval use case using arbitrary documents as a query, for example in the vector space model [34]. However, apart from information retrieval, many other applications that deal with textual data may benefit from text similarity algorithms and associated index structures. For example, in an email application, it can be used to show emails related to the one the user is currently reading or writing. It can also

be used as a basis for text clustering, automatic filtering, automatic classification of emails into folders, and even recipient prediction or verification.

On the other hand, it is a computationally intensive problem because the worst case time complexity of exhaustive comparisons amongst texts is quadratic. One possible technique to reduce the number of comparisons is a pruned inverse index that associates a term with a limited set of documents. But any kind of pruning has the implied risk of missing relevant information by simply skipping it. This might lead to a reduction of the number of actually found similar items. A good pruning, however, could even increase the quality of the found items in terms of semantic similarity by inspecting only *relevant* terms. The central problem hereby is the computational representation of *relevancy*. Often, this is done by employing measures of statistical significance. The differential analysis formula presented in [40] (which is in turn based on the log-likelihood significance test [14]) is used in the following experiments to build a proprietary document index based on the frequency of a word in a document in comparison to its frequency in a pre-processed corpus as well as the size of the document.

Another aspect of measuring semantic similarities is that, even without any pruning, there is a risk of missing documents of the same topic(s) that use different words of close, similar or related meanings. For example, the two sentences "That tank fired a round" and "Armor fire detected" would be missed by an index that is based only on words. Additional knowledge (as provided, for example by a thesaurus) is needed to allow the text similarity system for matching words of related meaning. This knowledge can be provided by a lexical database like WordNet [32, 38]. The availability or size of such manually compiled lexical databases is rather limited for some languages. In some cases even their applicability beyond academic projects is limited. However, as discussed under the distributional hypothesis earlier [6], a corpus in a language and a domain is sufficient to automatically compute a knowledge base of distributional semantics that may serve the same purpose.

As a result, text similarity systems consist of a variety of components. Figure 1 shows the main building blocks of our NLP solution. Pre-processing starts with parsing of raw documents like textual emails, HTML, XML, or Office documents, and then continues with language detection, tokenization and text cleaning. The text cleaning part is highly dependent of the source and text type of the data. For generic web pages, for example, we employ a self-learning module that learns to skip advertisements and structural data like menus. In the email use case, processing and correct handling of quotations and signatures is central to the quality of the text similarity algorithm and other parts of the NLP processing chain. We have discovered that when our system does not recognize signatures and hence cannot exclude them from any processing, the performance of the text similarity system degrades severely for users who share many messages on different topics (e.g. co-workers or friends). Although in theory there is a *convention* regarding how to separate signatures from the main body (`- \n`), many if not most users choose not to follow the convention. Signature detection has thus been defined as a classification task [8] with the conventional pattern as just one out of many

possible features or as one part of a generalized email structure classifier as outlined above [30].

Typically, a complex information retrieval system also includes a lemmatization module which maps, for example, "fired" and "fire" on the same lemma. It may also contain the aforementioned thesaurus module which would allow for matching *tank* with *armor*. We assume that usage differences of personal language affect some areas of language more than others. For example, differences in pronunciation or choice of words tend to be stronger than differences in grammar or morphology. As a consequence, we state that some modules might profit only very little or not at all from a perfect fit to the data of the user. Other modules, however, would profit much more. In order to show the potential influence of personalizing NLP systems through self-learning techniques we now discuss several experiments.

3.1 Evaluation Method

Traditionally, evaluation of *Information Retrieval* (IR) systems has been centered around the so-called Cranfield Experiments using gold standards (e.g. TREC[7]). These gold standards offer various beneficial properties for the research community like comparability and repeatability. They are limited in scope and availability, however. Apart from that personalized information retrieval systems are hard to evaluate using standardized gold sets, since gold sets cannot easily capture the applied personalization. This might explain why the evaluation of personalized IR systems has focused on user studies [13]. While these user studies can clearly capture the personalization, they are limited in terms of repeatability and comparability. Gold standards require higher up-front costs but may be reused any number of times. User studies on the other hand lead to continuous costs for each experiment or improvement cycle. Simulated user studies or a living lab as described in [3, 26] might be a solution to combine the desired repeatability of experiments with the ability to capture personalization.

In order to quickly measure the impact of certain improvements, a specialized gold standard may provide a good estimate of the impact in the final product. Due to licensing restrictions and the previously mentioned scope and limitations of availability, we decided to create our own gold standard data sets. We distinguish gold standards by their language, source and content type. Each gold data set consists of a document collection with a list T of topics from a defined text source. Each topic contains a number of associated documents. We make a conscious simplification by assuming that a document is related to exactly one topic. Additionally we evaluated our system using the data from the SemEval 2012 and 2013 tasks on *Semantic Text Similarity* [1, 2].

[7] trec.nist.gov.

3.2 Experiments on News and Email Text Collections

For evaluation we add the entire collection to the index and then iterate over all documents d in the current document collection D. For each d we query a ranked top n list of similar documents d'. Each d' is then compared to the content of T (a list of topics, see above) of the original document d. From these counts of matching and non-matching documents we calculate the precision, recall and F_1-score (i.e., the harmonic mean of precision and recall). We use two different English gold standards which encompass:

- A collection of 26 topics in news texts where exactly 20 texts are given per topic
- A collection of 18 topics in emails from a single user with an average of 12 emails per topic.

The following experiments show the impact of domain dependence and advanced domain dependent filtering on our semantic text similarity index. To show the influence of domain dependence we compare pre-trained models with models trained directly on the evaluation data. Additionally we show the impact of signature detection to the email use case.

For the following experiments we used word frequencies and distributional semantic clusters pre-calculated on a 1 million lines newspaper corpus comparable to those available from the Wortschatz project [18]. In addition, we created a small corpus from the email gold data set with about 6,000 lines of text and computed distributional semantic clusters based on that as well.

E_0 is the baseline experiment evaluated on the news gold standards using our pruned index, distributional semantics, and word frequencies calculated from our newspaper corpus. E_1 evaluates the email gold standard using the same parameters as E_0. For E_2 we employed adapted models by using the distributional semantic clusters computed on the email corpus. For E_3, E_4 and E_5 we additionally executed a signature detection module to filter the emails. E_3 hereby uses the same default semantic clusters as employed in E_1 and a generic pre-trained version of our custom signature detection classifier. E_4 refines the signature detection classifier by training it directly on the emails. Finally E_5 is a combination of both the adapted semantic clusters as in E_2 and the trained signature detection classifier. For each of the experiments we show the aggregated precision, recall and F_1-score as computed by querying the index for the top 10 similar items to each given text. The results are summarized in Table 1.

With an F_1-scores of 72.43 % Experiment E_0 clearly outperforms Experiment E_1 (56.73 %). There are two main reasons for this. First of all, the real-world email data of E_1 is noisier than the manually cleaned news text. Among others, the email dataset contains large signatures, complicated quotes, bad spelling, ASCII art and even partial email headers in quotes, tables or stack traces. Secondly, we assume that the trained newspaper model is a better match for the news gold standard than for the emails. Further evidence for this assumption can be observed by using the computed distributional semantics of the email gold set as described above. By comparing the

Table 1 Experiments on source-dependent text cleaning and domain-dependent distributional semantics in text similarity measurement

E	Semantic clustering	Signature detection	Data set	Top N	Precision (%)	Recall (%)	F_1-score (%)
E_0	Default	n/a	News	10	72.51	72.36	72.43
E_1	Default	None	Email	10	55.94	57.54	56.73
E_2	Email	None	Email	10	59.03	58.01	58.51
E_3	Default	Pre-trained	Email	10	42.28	41.99	42.13
E_4	Default	Trained	Email	10	53.65	54.88	54.26
E_5	Email	Trained	Email	10	58.31	55.45	56.84

F_1-scores of E_1 versus E_2 and of E_4 versus E_5, we note a rather small but persistent difference of about 2 % (F_1). The small factor is partly due to the fact that we only fitted the distributional semantics. As described earlier, the distributional semantics module only works as an extension to the general pruned index which is similar to a thesaurus. All other parameters like the very important word frequencies are kept identical. Considering the size of the two corpora (one with 6,000 lines of email text and the other with 1 million lines of generic newspaper text), this effect is nonetheless surprising, especially since prior work [6] on the effect of corpus size and the quality of semantic relation extraction showed a direct logarithmic dependency of both. From these findings it seems that domain adaptation is at least as important as corpus size. Hence it can be predicted that, on larger personal email collections, the performance will increase for the personalized semantic model. At the same time, performance is likely to stay the same for the trained model or even degrades if the discourse of the personal communication moves away from general topics. While the distributional semantic model benefits from on-device adaptation our generic signature detection algorithm needs to be trained on specific emails to actually perform well. This can easily be seen by means of Experiment E_4 and E_5 which clearly outperform E_3.

In summary, we have shown that for a task like signature detection per-user adaptation is a requirement in order to provide acceptable performance. Furthermore, even for algorithms like semantic clustering, we have initial evidence showing that domain adaptation and on-device training can be more important than corpus size and design.

In regard to our discussion so far it seems strange that E_4 and E_5 do not outperform E_1 and E_2 as clearly as E_3. This effect is, however, due to the relatively small size of our email gold standard. As noted earlier, there is a strong correlation between senders and topics in the gold standard. Since many topics in the email set contain only a discussion of a few selected people, signatures turn out to be good features for association. On a more complete email inbox a user will most probably have discussions on different topics with the same persons over the course of time. This is clearly an example of how challenging it is to scope gold standards correctly to reflect the final use cases.

3.3 (Unofficial) Semantic Text Similarity Experiments

In an effort to provide evaluation data on standard datasets we also ran our system against the shared *Semantic Text Similarity* task (SemEval workshops of 2012 and 2013 [1, 2]). The STS evaluations consist of sentence pairs associated with scores between 0 and 5 to indicate their observed semantic relatedness. Here 0 is not related and 5 is identical on a semantic level. The evaluation pairs were gathered from different sources such as video subtitles, statistical machine translation, news headlines or glossaries (cf. [1, 2]). In 2012 35 teams submitted a total of 88 system runs, in 2013 34 teams participated in the similar CORE sub-task submitting 89 system runs in total.

Before considering the results, it is necessary to stress the following facts. Neither did we optimize our system to perform well on sentence-level comparisons (it is optimized to compare entire emails with each other), nor do the STS evaluations take the amount of CPU cycles to compute the similarity, RAM usage or resources disk space usage into account when comparing the quality of the algorithms. Our system, for example, has to have response times for computing the 10 most similar documents in a collection of 1,000 documents of less then several hundreds of milliseconds (exact numbers depending on specific phone models). The linguistic processing also has to never exceed more than a couple of megabytes of RAM usage (irrespective of other factors, e.g., document size), because any android process using more than 35 megabytes will be terminated by android without warning and the application has to perform many other operations such as user interface processing or data base handling at the same time. Additionally, download sizes of applications still typically range within a few dozen megabytes, with anything bigger than that considered to be too large. Hence, we were forced to optimize our entire language models to be around one to three megabytes per language.

In 2012 most algorithms with decent results relied on a mixture of different string- or word-based similarity algorithms combined with external knowledge sources (e.g. WordNet), or complex linguistic tools (e.g. parsers, PoS-taggers, machine learning and sometimes even machine translation (an overview is provided in [1, p. 392])). UKP, the best performing system in 2012, which also served as a baseline for 2013 and still performed very well, uses a combination of many different string and word based features, explicit knowledge from WordNet and Wikipedia and a distributional thesaurus leading to more than 300 different vectors [4]. We do not have access to exact numbers, but it is likely a computationally rather complex approach to the problem.

Our results were matched to the gold standard using the WEKA toolkit [22] to calculate a simple supervised linear regression for each dataset. To work around one issue of our system when dealing with very short sentences consisting solely of stopwords, we defined a return value of 5 in cases where the lowercased words between both sentences all matched. The results are summarized in Table 2. Our unofficial results show us at rank 57 of 88 in the 2012 challenge. A run on the freely available parts of the 2013 gold sets using regression parameters obtained on 2012

Table 2 Pearson correlation on STS 2012 and 2013 datasets including the rank

Dataset	Year	Correlation	Rank
MSRpar	2012	0.37072	77
MSRvid	2012	0.69851	57
SMTeuroparl	2012	0.49825	17
OnWN	2012	0.57392	59
SMTnews	2012	0.42229	7
Weighted mean	2012	0,52431	57
Headlines	2013	0.62939	52
OnWN	2013	0.63346	26
FNWN	2013	0.14607	78

data shows us again at a mid range position. Due to the distribution restrictions on one 2013 dataset, we cannot provide an overall mean.

Our system appears to have a particular weakness regarding the headline paraphrases of the MSRpar dataset of 2012 and the FrameNet-WordNet glossary pairs in the FNWN dataset, which we will look into in future improvement rounds. Presently, our system also does not contain several modules yet, because they are still under development or because they are excluded for reasons of efficiency, such as unsupervised lemmatization, our time extraction module, or the extraction module for measurement units.

4 Information Extraction

The extraction of information out of textual data is a complex and challenging task in NLP. The field of information extraction has evolved over the last few decades. Starting with early conferences focusing on message understanding [19, 20], information extraction was firstly defined as the task of extracting entities, events, and relations. These entity types range from names of locations, persons and organizations, temporal expressions and monetary values; the defined relation types cover logical relations between those entities (e.g. *Located-In(Org, Loc)*).

Today, research efforts in this area are closer to a deep understanding of language [10, 37]. This includes detection and extraction of manifold entities which are not necessarily restricted to named entities (e.g. entities from the biomedical domain [25] or arbitrary expressions for practical purposes such as shopping list items). For many use cases extraction of simple logical relations is only a minor step towards more complex structures. We consider detecting and understanding flight confirmations as one representative example. This task includes extraction of numerous entities (e.g. airports, departure / arrival time and flight durations) and logical relations (e.g. departure airport / time, arrival airport / time), sanity constraints (e.g. departure time must be earlier than arrival time, time zone(s)), and additional information (e.g. baggage restrictions, flight booking references).

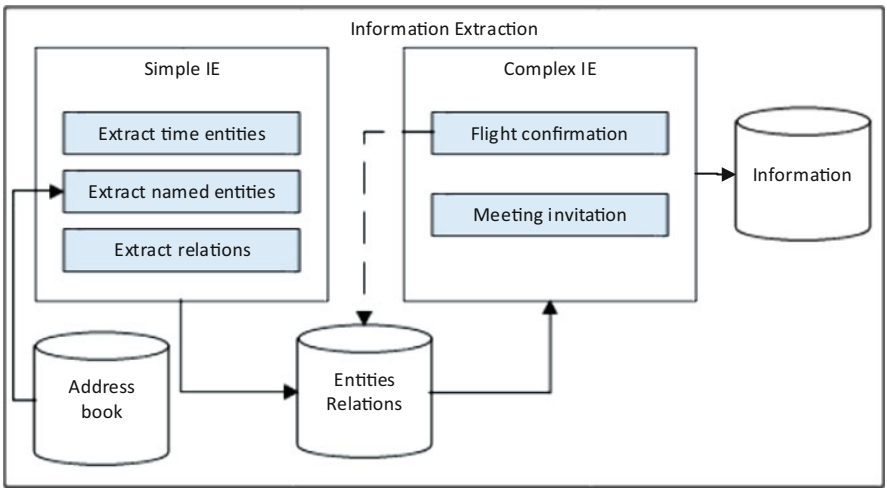

Fig. 2 An overview of the information extraction system

In order to master information extraction for these and similar complex tasks, we created a hybrid information extraction system (see Fig. 2) combining the strength of established statistical approaches for *Named Entity Recognition* (NER) with rule-based approaches to ensure special properties of the information to be extracted without neglecting characteristics of personal data and restrictions of applying this system on mobile devices. Furthermore, the information extraction system should be able to pass extracted information arbitrarily from one layer to another (e.g. from the layer of complex information extraction back to the layer of simple information extraction). This leads to an iterated information extraction process capable of using features which were not there during a first run through the text (e.g. after the text type extractor found out that the current text is a flight confirmation then the normal NER module could use this information to increase the possibility of three letter abbreviations to be airport codes).

4.1 (Named) Entity Recognition

We divide the extraction of (named) entities into two approaches: the rule-based and the statistical one. We developed flexible and robust rule-based solutions for detecting temporal expressions (e.g. 04/12/2014, yesterday in the evening, today) and measurements (e.g. $100, 12 km).

Statistical approaches are used for detecting person names, places and company names. Our system is based on the well-established Maximum Entropy classifier architecture [9] rather than on *Conditional Random Fields* (CRFs) due to lower resource usage. It is trained using many of the features that have achieved

competitive results such as lexical features, PoS-tags, affix information, and word structure information [35].

In contrast to common NER systems that operate on PCs, we have to deal with a number of limitations:

- Mobile devices have slow CPUs and limited amount of memory which poses serious limitations on the model size and, thus, on number of features that can be processed.
- Supervised PoS-taggers do not perform well on noisy data [36] and require large models.
- User-generated data (e.g. SMS, emails, Twitter messages) are very noisy: they contain a high number of (personal) abbreviations, no reliable case information, and highly personalized personal language patterns.

We developed a number of strategies to circumvent these major challenges. We incorporated a fast, unsupervised PoS-tagging producing a small model size, and which does not require huge amounts of memory (our model is based on [29]). Furthermore, unsupervised PoS-tagging adds a semantic differentiation to the PoS which provides additional information to the NER model (e.g. there is no class containing all normal nouns, there are multiple semantic classes containing normal nouns separating words like weekdays, cities, countries, first names, last names etc.). Additional semantic classes for all words within a sentence provide more information about the content of a sentence to the classifier which is more robust against idiosyncratic language structures and case information (which is sometimes missing and sometimes completely broken due to automatic corrections made by T9 (text on 9 keys)). Most importantly, our unsupervised PoS-tagging naturally adapts itself to any domain by virtue of being completely unsupervised. Hence, domain adaptation for our system is reduced to the (still sometimes challenging) task of collecting a sufficient amount of text samples of the domain in question. For example, when we trained a PoS-tagging model on a biomedical domain corpus, it produced POS-tags with highly useful distinctions between genes as one "POS-tag" vs. diseases and medical ingredients as another cluster instead of lumping everything together into one huge "noun" cluster.

In addition to these restrictive adaptations, our NER system benefits from the usage of personal resources that are available on-device.

4.2 Integration of Personal Resources

Although there are massive restrictions regarding running NER systems on mobile devices, is it possible to use their personal resource storage to improve the information extraction system. The NER classifier can be backed up by various approaches as exemplified subsequently.

4.2.1 Address Book

What kind of information is available on all mobile phones? The obvious answer is: an address book. It contains the names and telephone numbers of the most frequent communication partners of a person. It may also contain addresses and user names of this person in social networks or communication platforms like Facebook, Twitter and Skype. Especially the latter ones are typically very hard to detect with traditional name extraction methods. Merely adding all the contained names and places to the personal name list of the NER system easily increases the perceived performance of the entire system.

Nevertheless, the disambiguation of names is still a necessary step since people do not always use full names when writing messages, for example when the receiver knows which *Christian* or *Mr. X* the sender refers to due to context or previous messages. Even the address book might contain multiple persons sharing the same first or last name (typical due to family name sharing). Since this disambiguation step is not important during the NER step, it is postponed to a semantic disambiguation module.

Another important benefit is the automatic linking to alternate names of the people stored in the address book. Most pre-trained NER modules are unable to detect nicknames used in social networks or communication apps like Skype or Twitter unless they are trained on respective pre-annotated corpora with adapted feature sets that do not exist for multiple languages. They are even less able to resolve nicknames to real names without a complex system which collates and keeps track of information across different sources. Using the information of an address book that contains real-world names linked with various nicknames of various social networks enables our NER module to use matches of these nicknames as on-device training instances.

4.2.2 Exploiting the Personal Corpus

The personal corpus contains many documents of personal communication. Although it is not possible to train a complex classifier directly on-device, it is possible to exploit the personal corpus in several ways. The main challenge of a NER system is to classify unseen entities. It also has to rely on the context within neighboring tokens, the sentence or even the document. This context can be extended to the complete personal corpus, too. In many cases, the direct context does not contain enough information to classify an entity into its correct class. Most approaches take context tokens occurring within a window of two tokens around the target token into account. Such contexts seldom contain sufficient information to decide the target word's type (e.g. "[quiet around] X [for the]"). Looking at the complete sentence unveils more valuable context and the decision becomes easier: "After months of hype, it's been pretty quiet around X for the last few weeks." But still, X could be a person, company, place or even a fancy new candy.

In news texts the document would most likely contain X in other positions with more context information than in this sentence and it should be possible to classify the token(s) based on the additional information. Short messages, emails, and tweets are very short because in personal communication explanations of common knowledge is left out as discourse participants are relatively familiar with each other. Hence, further references to entities might not exist in the document.

This is where the personal corpus can be used to find more evidence for entities that the user might be interested in. The personal corpus contains all communication data and thus, the entities which are important / interesting for the user. Our assumption is that crawling previously extracted entities can yield a high chance to find more contexts for the token(s) in question that makes it easier to extract proper name type (or conversely to prove that no name is present in a given text).

4.2.3 Combining Precomputed NER Models with Personal Models

Precomputed models work well when they are applied on data that is similar to the training corpus used. When applied to personal communication the personal language of the user becomes a non-negligible factor. Complete on-device NER model creation is not possible due to hardware restrictions and the lack of annotated data. Furthermore, the model would need a very long time to collect enough instances to make confident decisions. Thus, we propose a combined model consisting of a pre-computed model and a less complex personal (e.g. a Naïve Bayes classifier [16]) model which is iteratively trained directly on-device.

The user should get the possibility to revise extractions made by the NER system. A convenient GUI should be able to provide inconspicuous feedback loops for extracted names (e.g. the possibility to cross out false positives, verify true positives and add false negatives upon result list presentation). In addition, the personal model can be trained using this user feedback additionally to boost instances like address book matches that are not recognized by the general model and confident classification results made by the general model in order to provide more training instances.

Both models are members of an ensemble [17] and are weighted differently. Regular evaluation runs that validate the personal model against the general model provide a reliability score for the personal model. This score is used to weight the personal model and to account for its steadily improving classification performance. The combined model then applies both classifiers and combines the resulting scores according to the respective weights of the classifiers.

In order to demonstrate the crucial information gain obtained by including the address book as a list of known names we sampled some statistics from an authentic email box. This email box contains 913 emails in which the trained NER model

Table 3 Experiments on incorporating user feedback during iterative training of a NER module

	Precision(%)	Recall(%)	F-Score(%)
Without user feedback	77.67	56.99	65.74
With user feedback	**89.68**	**74.40**	**81.33**

tagged about 16.8 k tokens as person names.[8] More than 8.6 k of them are tokens covered by matching against the address book (which only contains 65 different person name tokens). Although the address book is not very comprehensive, this experiment shows that the most common names in personal communication are covered by address books. Furthermore, it provides a considerable amount of positive training instances, even in fragmentary contexts like signatures or in SMS that lack proper case patterns. We thus expect further work to prove our hypothesis that even lightweight classifiers which are trained on-device can improve the performance of pre-trained models significantly.

We also performed an experiment to evaluate the impact of user feedback on iterative NER module training. Iterative training uses a precomputed model and applies it to the user's data. An intuitive GUI provides the possibility of manual corrections (e.g. deny false positive matches, annotate missing matches or correct the boundaries of detected entities) to the user. This feedback is used to improve the training data of the iteratively trained classifier which afterwards achieves superior result to the precomputed one. In our experiment on English data (mixture of user-generated data and web texts with 1,139 annotated person names) the user feedback resulted in an increase of more than 15 % (F1) (see Table 3), although the user was told to only focus on the most frequent mistakes. This also means that each user may improve the performance of the NER module until the quality fulfills the user's expectations.

5 Conclusions

We have built a robust and efficient NLP system for mobile devices. We have shown, partially demonstrated, and briefly explained the most important design decisions which make any mobile-centric NLP system different from typical NLP systems that have access to virtually unlimited CPU and RAM resources, and which deal with clean data.

Even though our on-going task is to continually develop our system, there are already several clear conclusions that can be drawn as follows:

[8] This equals an average of more than 18 person name tokens per mail. Besides many false positives (classified incorrectly due to broken contexts in log file texts, quoted mails and signature parts), most of them can be verified as correct.

Personal language variation Our work shows that the influence of personal language variation on the performance of an NLP system is indeed very strong and goes very deep through many layers of language processing. The influence appears to be about as strong as that of corpus size. This also implies that optimal performance can only be achieved if sufficient amounts of everyday communication content from a particular user can be input into the system. Doing so can optimize both the analysis of personal language usage as well as corpus size.

Personal data Personal data constitutes an extremely rich and varied domain in terms of the data types present and the amount of data available. Large amounts of personal data can easily be used for automatic, on-device refinement of various NLP subsystems.

On-device processing Since personal data is very sensitive in terms of data protection and privacy issues, the best way to build trust with the users is to not send their data anywhere but to analyze it directly on the device. Thus, the results of the analyses belong to the user and remain on her/his device. We observed that many users displayed a certain amount of uneasiness when they discovered the high quality of some (from their perspective) more advanced analyses such as the *compelling* automatic semantic recipient suggestion (or alert in case the user clearly added the wrong recipient) based on the contents of the email they are writing. These emotions did not develop further into an outright rejection because the users were always fully aware of the fact that their personal data would not be transmitted anywhere.

At least baseline It is common knowledge that many NLP solutions require context for their decisions. However, in real-world applications many contexts are completely misleading. For example, attempting to run a PoS-tagger on the textual lines found in a flight ticket confirmation will result in very low hit rates because in this case the algorithm degenerates to plain vocabulary look-up performance levels at best. Hence, a robust real-world NLP solution will provide at least baseline solutions for all levels of processing. Even just detecting a flight ticket as non-valid input into the PoS-tagger and NER system will prevent many false detections.

Robustness and language scale-out Real-world applications have to deal with very diverse and noisy data. Besides employing advanced adaptive preprocessing techniques, we focus our efforts on robust unsupervised algorithms such as an unsupervised PoS-tagger or automatically computed semantic clusters. Although these algorithms are known to produce output of lower quality when being applied under artificial conditions of scientific evaluations, we emphasize robustness, performance and model size over the last percent of theoretical performance. If the employed algorithm is good enough in terms of user acceptance, we simply use it rather than that we focus solely on theoretical numbers. Another benefit of using unsupervised algorithms is the possibility to scale up the number of supported languages and domains with minimal manual effort. This is a central aspect for a rather small company addressing an international market.

We found that the increased processing power of the mobile devices and the possibilities of compressing NLP subsystems converged sufficiently to allow ubiquitous NLP systems to be pre-installed on even mid-range mobile devices without hurting their core performance in any noticeable ways. Nevertheless, the processing power of current mobile devices is still not sufficient to run all possible or necessary core analyses directly on-device. Moreover, some analyses also require initial background corpus processing in order to produce good results. The most obvious example in this regard is the background corpus-based word frequency computation for keyword extraction where the deviation of personal language usage from default language usage patterns is estimated. Another example is the PoS-tagger clustering and training which definitely requires a corpus larger than what would could be compiled from limited personal language usage samples, and which also requires considerable CPU and space resources that are not yet available on current mobile devices. We have found a good balance between server-side and client-side processing which is facilitated by the fact that all server-side processing can be done as a pre-analysis that results in compressed models eventually shipped to the user.

References

1. Agirre E, Cer D, Diab M, Gonzalez-Agirre A. Semeval-2012 task 6: A pilot on semantic textual similarity. In:*SEM 2012: The first joint conference on lexical and computational semantics – vol 1: Proceedings of the main conference and the shared task, and vol 2: Proceedings of the 6th International workshop on semantic evaluation (SemEval 2012), pp 385–393, Montréal, Canada, 7–8 June 2012
2. Agirre E, Cer D, Diab M, Gonzalez-Agirre A, Guo W (2013) *Sem 2013 shared task: Semantic textual similarity. In: Second joint conference on lexical and computational semantics (*SEM), vol 1 Proceedings of the main conference and the shared task: semantic textual similarity, pp 32–43, Atlanta, Georgia, June 2013
3. Azzopardi L, Balog K (2011) Towards a living lab for information retrieval research and development: a proposal for a living lab for product search tasks. In: Proceedings of the 2nd international conference on multilingual and multimodal information access evaluation, CLEF'11. Springer, Berlin, pp 26–37
4. Bär D, Biemann C, Gurevych I, Zesch T (2012) UKP: Computing semantic textual similarity by combining multiple content similarity measures. In: *SEM 2012: The first joint conference on lexical and computational semantics – vol 1: Proceedings of the main conference and the shared task, and vol 2: Proceedings of the 6th international workshop on semantic evaluation (SemEval 2012), pages 435–440, Montréal, Canada, 7–8 June 2012
5. Barlow M (2013) Individual differences and usage-based grammar. Int J Corpus Linguist 18(4):443–478
6. Bordag S (2007) Elements of knowledge-free and unsupervised lexical acquisition. Phd, University of Leipzig, Leipzig
7. Carlson A, Betteridge J, Kisiel B, Settles B, Hruschka ER, Mitchell TM (2010) Toward an architecture for never-ending language learning. In: Proceedings of the conference on artificial intelligence (AAAI), pp 1306–1313, AAAI Press
8. Carvalho VR, Cohen WW (2004) Learning to extract signature and reply lines from email. In: Proceedings of the conference on email and anti-spam

9. Chieu HL, Ng HT (2002) Named entity recognition: a maximum entropy approach using global information. In: Proceedings of the 19th international conference on Computational linguistics, pp 1–7, Morristown, NJ
10. Collobert R, Weston J (2008) A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on machine learning, Helsinki, Finland
11. Corbett P, Batchelor C, Teufel S (2007) Annotation of chemical named entities. In: Proceedings of the annual meeting of the ACL, pp 57–64, ACL
12. De Saussure F (1916) Cours de linguistique générale. Payot, Lausanne/Paris
13. Dumais S, Cutrell E, Cadiz JJ, Jancke G, Sarin R, Robbins DC (2003) Stuff I've seen: a system for personal information retrieval and re-use. In: Proceedings of the 26th annual international ACM SIGIR conference on research and development in informaion retrieval, SIGIR '03, pp 72–79, New York
14. Dunning T (1993) Accurate methods for the statistics of surprise and coincidence. Comput Linguist 19(1):61–74
15. Ferrucci D, Brown E, Chu-Carroll J, Fan J, Gondek D, Kalyanpur A, Adam L, J William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, Chris Welty (2010) The ai behind watson - the technical article. AI Mag 31
16. Fleischman Michael, Hovy E (2002) Fine grained classification of named entities. In: Proceedings of the 19th international conference on Computational linguistics, pp 1–7, Morristown, NJ
17. Florian R, Ittycheriah A, Jing H, Zhang T (2003) Named entity recognition through classifier combination. In: Proceedings of the seventh conference on natural language learning at HLT-NAACL 2003, vol 4, pp 168–171, Edmonton
18. Goldhahn D, Eckart T, Quasthoff U (2012) Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In: Proceedings of the 8th international conference on language resources and evaluation (LREC'12), pp 759–765
19. Grishman R (1995) The NYU system for MUC-6 or where's the syntax? In: MUC6 '95: Proceedings of the 6th conference on Message understanding, pp 167–175, Morristown, NJ
20. Grishman R, Sundheim B (1995) Design of the MUC-6 evaluation. In: MUC6 '95: Proceedings of the 6th conference on message understanding, pp 1–11, Morristown, NJ
21. Grouin C, Rosset S, Zweigenbaum P, Fort K, Galibert O, Quintard L (2011) Proposal for an extension of traditional named entities: From guidelines to evaluation, an overview. In: Proceedings of the 5th linguistic annotation workshop, LAW V '11, pp 92–100, Stroudsburg, PA, 2011
22. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: An update. SIGKDD Explor Newsl 11(1):10–18
23. Heyer G, Bordag S (2007) A structuralist framework for quantitative linguistics. In: Alexander Mehler and Reinhard Köhler, editors, Aspects of Automatic Text Analysis / Series: Studies in Fuzziness and Soft Computing. Springer, Berlin, New York
24. Heyer G, Quasthoff U, Wittig T (2008) Text mining: Wissensrohstoff text – konzepte, algorithmen, ergebnisse. W3L-Verlag, Herdecke
25. Kim JD, Pyysalo S, Ohta T, Bossy R, Nguyen N, Tsujii J (2011) Overview of BioNLP shared task 2011. In: Proceedings of the BioNLP shared task 2011 workshop, pp 1–6. ACL, 2011
26. Kim J (2012) Retrieval and evaluation techniques for personal information. PhD thesis, Graduate School of the University of Massachusetts, 2012
27. Kiss T, Strunk J (2006) Unsupervised multilingual sentence boundary detection. Comput Linguist 32(4):485–525
28. Kushmerick N (2000) Wrapper verification. WWW 3(2):79–94
29. Lamar M, Maron Y, Johnson M, Bienenstock E (2010) SVD and clustering for unsupervised pos tagging. In: Proceedings of the ACL 2010 conference short papers. Uppsala, pp 215–219
30. Lampert A, Dale R, Paris C (2009) Segmenting email message text into zones. In: Proceedings of the 2009 conference on empirical methods in natural language processing: vol 2 - vol 2, EMNLP '09. Stroudsburg, PA, pp 919–928

31. McMenamin GR (2002) Forensic linguistics: Advances in forensic stylistics. CRC Press, London
32. Richardson R, Smeaton AF, Murphy J (1994) Using WordNet as a knowledge base for measuring semantic similarity between words. In: Technical Report, Proceedings of AICS conference, 1994
33. Rudman J (1997) The state of authorship attribution studies: Some problems and solutions. Comput Hum 31(4):351–365
34. Salton G (1989) Automatic text processing: The transformation, analysis, and retrieval of information by computer. Addison Wesley, Reading
35. Tjong Kim Sang EF, Meulder FDe (2003) Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In: Daelemans W, Osborne M (eds), Proceedings of CoNLL-2003, volume pages, pages 142–147
36. Schierle M (2011) Language engineering for information extraction. Phd, University of Leipzig, Leipzig
37. Schuetze H, Scheible C (2013) Two svds produce more focal deep learning representations. CoRR, abs/1301.3, 2013
38. Varelas G, Voutsakis E, Euripides, Petrakis EG, Milios EE, Raftopoulou P (2005) Semantic similarity methods in WordNet and their application to information retrieval on the web. In: 7 th ACM international workshop on web information and data management (WIDM 2005), pp 10–16, ACM Press, 2005
39. Witschel Hf (2004) Terminologie-extraktion – möglichkeiten der kombination statistischer und musterbasierter verfahren. Ergon Verlag, Würzburg
40. Witschel Hf (2007) Multi-level association graphs - a new graph-based model for information retrieval. In: Proceedings of the HLT-NAACL-07 Workshop on Textgraphs-07, New York, 2007