PaloPro: a platform for knowledge extraction from big social data and the news

Nantia Makrynioti*

Department of Informatics,
Athens University of Economics and Business, Greece
Email: makriniotik@aueb.gr
*Corresponding author

Andreas Grivas

Institute of Informatics and Telecommunications,
NCSR 'Demokritos', Greece
Email: andreasgrv@gmail.com

Christos Sardianos

Department of Informatics and Telematics,
Harokopio University of Athens, Greece
Email: sardianos@hua.gr

Nikos Tsirakis

Palo Services Ltd.,
Corinthia, Greece
Email: nt@paloservices.com

Iraklis Varlamis

Department of Informatics and Telematics,
Harokopio University of Athens, Greece
Email: varlamis@hua.gr

Vasilis Vassalos

Department of Informatics,
Athens University of Economics and Business, Greece
Email: vassalos@aueb.gr

Vassilis Poulopoulos and Panagiotis Tsantilas

Palo Services Ltd.,
Corinthia, Greece
Email: pv@paloservices.com
Email: pt@paloservices.com

Abstract: PaloPro is a platform that aggregates textual content from social media and news sites in different languages, analyses them using a series of text mining algorithms and provides advanced analytics to journalists and social media marketers. The platform capitalises on the abundance of social media sources and the information they provide for persons, products and events. In order to handle huge amounts of multilingual data that are collected continuously, we have adopted language independent techniques at all levels and from an engineering point of view, we have designed a system that takes advantage of parallel distributed computing technologies and cloud infrastructure. Different systems handle data aggregation, data processing and knowledge extraction and others deal with the integration and visualisation of knowledge. In this paper, we focus on two important text mining tasks, named entity recognition from texts and sentiment analysis to extract the sentiment associated with the corresponding identified entities.

Keywords: text mining; social media analysis; named entity recognition; NER; sentiment analysis; opinion mining.

Reference to this paper should be made as follows: Makrynioti, N., Grivas, A., Sardianos, C., Tsirakis, N., Varlamis, I., Vassalos, V., Poulopoulos, V. and Tsantilas, P. (2017) 'PaloPro: a platform for knowledge extraction from big social data and the news', *Int. J. Big Data Intelligence*, Vol. 4, No. 1, pp.3–22.

Biographical notes: Nantia Makrynioti is currently a PhD student at the Department of Informatics of Athens University of Economics and Business. She received her MSc in Information Systems from the same university (2013). Prior to that, she studied computer science at the Department of Computer Science of the University of Ioannina (nowadays, Department of Computer Science and Engineering). Her research interests lie in the areas of sentiment analysis, distributed machine learning and large scale analytics, as well as the application of machine learning in query optimisation.

Andreas Grivas received his BSc degree from Harokopio University of Athens, Department of Informatics and Telematics in 2014. His thesis was on the problem of text alignment using string similarity extracted from synonym graphs. He enjoys applied machine learning and at the time of writing has worked on named entity recognition, relation extraction and author profiling. He is currently a member of the PerSoNA group at NCSR Demokritos, Greece.

Christos Sardianos holds a Bachelor in Electronic Engineering and a Master of Science in Informatics and Telematics. Currently, he is a Research Associate and a PhD candidate in the research area of 'knowledge extraction from large scale social networks', under the supervision of Prof. Iraklis Varlamis, at the Department of Informatics and Telematics at Harokopio University of Athens. His main topics of interest are data mining, big data, recommender systems, social network analysis and mining, big graph analysis, business intelligence, databases, etc.

Nikos Tsirakis graduated from the Computer Engineering and Informatics Department, University of Patras, Greece in 2004 and then graduated Masters program 'Science and Technology of Computers' by 2006 in the same department. He received his PhD in Computer Science from the Department of Computer Engineering and Informatics of the University of Patras, Greece, in 2010. His research interests are focused in information retrieval, design and analysis of data mining algorithms and applications (mainly for huge data manipulation ex. data bases, data streams, XML data), social networks, hypertext modelling and searching, software quality assessment, and finally, web technologies. Currently, he is technical leader in Palo Services, a company that provides news and social media monitoring services in Greece, Serbia, Romania, Cyprus and Turkey.

Iraklis Varlamis is an Assistant Professor at the Department of Informatics and Telematics of Harokopio University of Athens. He received his PhD in Computer Science from Athens University of Economics and Business, Greece, and his MSc in Information Systems Engineering from UMIST UK. His research interests vary from data-mining and the use of semantics in web mining to social network analytics and knowledge extraction from social media and the news. He has published several articles in international journals and conferences, concerning web document clustering, the use of semantics in web link analysis and web usage mining, word sense disambiguation using thesauruses, etc. He holds a patent from the Greek Patent Office for a system that thematically groups web documents using content and links. More information is available at http://www.dit.hua.gr/~varlamis.

Vasilis Vassalos is an Associate Professor at the Department of Informatics of the Athens University of Economics and Business. Prior to that, he was an Assistant Professor at the Department of Information Systems in the Stern School of Business of NYU (1999–2003), and at AUEB (2003–2009). He has published more than 50 papers in international peer-reviewed journals and conferences. He holds two US patents for work on information integration, and regularly serves on the program committees of the major conferences in databases, and as a reviewer for the major journals. He was a co-founder of Enosys Software, a successful innovative startup company in enterprise information integration (acquired by BEA systems in 2003). He was a Visiting Professor at UCSD and a Marie Curie Outgoing International Fellow in 2007–2008, and a Visiting Professor at EPFL in 2013. He is currently a PI of the FET-Flagship Human Brain Project, working on medical data integration.

Vassilis Poulopoulos obtained his diploma from Computer Engineer and Informatics Department of the University of Patras in 2005, and in 2007, he completed his MSc degree. In 2010, he obtained his PhD degree by creating an innovative platform for multi-lingual worldwide article collection including data mining, data analysis, text extraction, text categorisation, text

summarisation and web personalisation. His basic fields of interest include: data mining, web technologies, web data integration, dynamic processing of web content, information extraction (from web content), web content summarisation, web content categorisation, website construction, web personalisation, OLPC, code integration, and databases. He has more than 35 publications in international journals, conferences and encyclopedias, and he obtained the best paper award twice. Currently, he is the Research and Development Director of Palo Services a company that provides news and social media monitoring services in Greece, Serbia, Romania, Cyprus and Turkey.

Panagiotis Tsantilas received his degree in Physics and his MSc in Computer Science and Business Administration at the University of Glasgow. His articles have been published in several Greek magazines and newspapers, and in 1995, he published his first book. He is the founder and CEO of palo LTD, which runs palo.gr, PaloPro, http://www.palo.rs, palo.com.tr, palo.com.cy and palo.ro. Palo.gr is the leading news search engine in Greece already holding two awards distinctions in e-volution awards 2013 and in Ermis awards 2012. PaloPro awarded the evolution awards 2014 for the innovative technology used in managing the corporate reputation online.

1 Introduction

The enormous advances in social media and their power to reflect and influence public opinion made them a domain of great interest for marketeers, communication specialists, journalists and entrepreneurs who want to invest in knowledge extraction from them. In this content-rich environment people report or comment on individuals, brands, products, services, etc. by providing references to named entities, polarised opinions about them and ratings about different aspects of the same entity. This huge amount of information is mainly unstructured text addressing human readers and hence, the only way to extract useful knowledge from it is by using natural language processing (NLP) techniques.

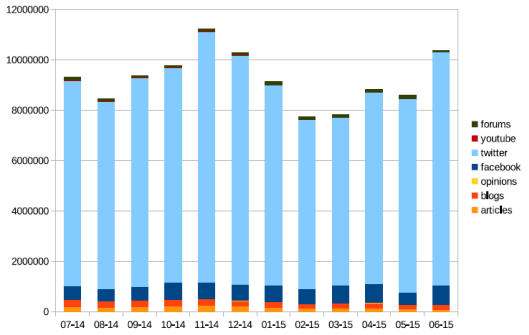
Despite the recent advances in NLP research that led to producing artificially intelligent behaviours, e.g., Google, IBM's Watson, and Apple's Siri, there are still many challenges to be faced in order to allow knowledge extraction from social media content to scale to big data. Popular NLP algorithms are tested for their performance in small-sized, properly curated corpora and have not been evaluated their ability to handle abundant social media content. Existing text mining and text processing solutions are tuned and tested only for English and a few more popular languages, but can hardly adapt to any language, especially languages with minimum linguistic resources. Finally, NLP researchers face the challenge to jump from the syntactic to the semantics curve (Cambria and White, 2014) in text representation and analysis.

Big players from the web and databases domains invest in social media analytics with generic frameworks and platforms (e.g., IBM social media analytics) that emphasise on the analytics part but do not focus on text mining, or with extensions of their existing platforms (e.g., Google news lab and Google analytics) that incorporate content from specific social media using associate data hubs and plugins (e.g., Google's social data hub). Social media and news platforms on the other side, provide comprehensive social media data and libraries of tools for analytics (e.g., Thomson Reuters Machine Readable News, Radian 6, Lexalytics, Synthesio). They use Twitter, Facebook and other social media APIs to collect data in streams and provide commercial archives/feeds and associated analytics. Finally, social media monitoring tools such as Brandwatch (http://www.brandwatch.com), Sysomos (http://www.sysomos.com/), Trackur (http://www.trackur.com/) and Engagor (https://engagor.com/) focus on the monitoring of popular social media, target central European and US markets and their primary users are market analysts with good technological background, since they offer complex visualisation interfaces which are intended to be used by experienced users.

In this dynamic environment, we have developed a flexible infrastructure, which allows to quickly expand to new markets, to collect, analyse and visualise social media pulse about companies and products. The company behind PaloPro platform was founded in Greece, in 2008 and launched the first news search engine for Greece, which offers news clustering and news summarisation services. At the end of 2012 the company launched an innovative platform for monitoring, measuring and analysing all web mentions of a company, brand, person or product that was introduced in the Greek market. As part of its expansion strategy, the company has already launched the platform in Greece, Serbia, Cyprus, Romania and Turkey since it targets the market of South Eastern Europe where the competition is not so mature and the technological challenges, due to multi-linguality and content size are even harder. The platform uses a unique crawling and data analytics mechanism, which can be quickly adapted to any new language thus allowing to expand to more EU countries At that point of expansion, PaloPro platform will have a unique advantage against competitors, which will be the pan-European market coverage.

As presented in Figure 1, the crawler module is able to collect a large number of data which is estimated at 1,000 articles per minute during the rush hours leading to an increase of 2.5 Gb per month of compressed data. The data in absolute numbers are more than ten million records per month from all the possible data sources while as it is obvious most of them derive from twitter.

Figure 1 Statistics on data collected within a year (July 2014–June 2015), (a) number of data collected per type (b) data distribution percentage per type (see online version for colours)



(a)

Type	Percentage
articles	1.82%
blogs	2.40%
opinions	0.06%
facebook	6.81%
twitter	87.71%
youtube	0.14%
forums	1.06%

(b)

Social media analytics involves a three-stage process: capture, understand, and present (Fan and Gordon, 2014). In the capture phase, a crawling engine is responsible for monitoring or 'listening' to various social media sources, archiving relevant data and extracting useful metadata. Since the collected data are not useful in their entirety, the understand phase is responsible for selecting relevant data, removing noisy and low quality data, and analysing the collected data using NLP and data mining techniques. The last phase (present) is responsible for displaying the findings of phase 2 to the end user in the form of dynamic reports. In this article we focus on certain aspects of the second and third phase. More specifically, we present our language agnostic solution for the recognition of named entities and the detection of opinion polarity about these entities. We present the pipeline we developed and provide initial evidence on its performance against news and social media post streams. In addition, we present the architecture that supports the analytics dashboard, which allows us to develop real time analysis components based on predefined templates and content dynamically collected from social media.

The advantages of the proposed architecture can be summarised in the following:

- a language agnostic solution for named entity recognition (NER) and sentiment analytics, which is based on machine learning techniques and a combination of automatically annotated corpora, manual annotations and crowdsourced annotated data (such as Wikipedia and DBpedia)
- a set of tools for automatic training-corpus creation and model training and a set of annotation tools that allow the fine tuning of our NER and opinion mining models
- a methodology, which has been tested against different language corpora comprising both formal (e.g., news articles) and informal texts (e.g., tweets and blog posts)
- a modular and multithreaded pipeline, which can easily take advantage of all available resources and perform on a cloud infrastructure.

In Section 2 that follows, we present an overview of related research works on NER and sentiment analysis (SA), which are the main social media analytics tasks presented in this work. In Section 3 we present the overall architecture of the platform and in Section 4 we provide the pipeline and main implementation details of the NER and SA modules. Section 5 performs a first evaluation of the two modules in terms of accuracy and Section 6 describes the challenges text analytics systems face. Finally, Section 7 concludes the paper.

2 Related work

2.1 Named entity recognition

The term NER refers to the task of recognising information units like persons, companies, organisations and locations and identifying references to these entities in structured or unstructured texts such as news articles, social media posts, comments, etc.

During the last 25 years, NER has attracted great research focus and a lot of work has been devoted to the analysis of English texts. Another large proportion of work addresses multilingualism problems in the field of NER. In this context, apart from English, languages such as German, Spanish, Dutch, Italian, Japanese, Chinese and French have been well studied. Moreover, many other languages (including Balkan languages) received some attention, such as Greek, Bulgarian, Catalan, Danish, Romanian, Russian and Turkish. One main approach for addressing NER is supervised learning, an approach wherein a system is trained on a corpus of annotated texts with intent to extract rules that maximise the probability of correctly identifying named entities and distinguishing between their different types in unseen texts.

A lot of multilingual or language agnostic NER approaches have been presented in the CoNLL-2003 shared task for language-independent NER (Sang and De Meulder, 2003), where the training dataset contained both English and German texts and the challenge was to label entities in untagged texts in both languages (test dataset). All the systems developed for the

CoNLL-2003 shared task used a wide range of machine learning techniques, such as maximum entropy models, hidden Markov models, AdaBoost.MH, support vector machines, conditional random fields (CRFs) and memory based models. Florian et al. (2003) tried to combine the results of four systems and reported that the robust risk minimisation model is the best option. According to this algorithm, the model was estimated on training data, by selecting the optimum class (entity type) for each word based on the words that appear in a window around the target word and a risk function which must be minimised for the selected class. One participant used the output of externally trained NER systems, though the rest of the systems attempted to use information from gazetteers and untagged data.

Johannessen et al. (2005) described a work on NER for Scandinavian languages (Norwegian, Swedish, and Danish). During their research project, named Nomen Nescio, they used rule-based and statistical methods to develop named entity recognisers. To develop the six different NER systems that form the NN project, they used rule-based methods based on constraint grammars, as well as shallow parsing with context sensitive finite-state grammars, OpenNLP as a named entity recogniser based on maximum entropy, TiMBL as a memory-based learning mechanism and gazetteers. In order to handle multi-membered names, they used pattern matching, lexicons and context rules.

An interesting approach was proposed by Szarvas et al. (2006) who introduced a multilingual NER system that identifies and classifies named entities in the Hungarian and English languages by applying AdaBoost.M1 and C4.5 decision tree learning algorithm. Essentially, they handled the NER problem as a classification of separate tokens taking into account the relationship between consecutive words as well using a window of appropriate size. They used a pre-annotated corpus of business news articles with an inter-annotator agreement rate of 99.8%.

Another interesting approach for annotating large corpora with NER tags is proposed by Richman and Schone (2008) who utilised the multilingual characteristics of Wikipedia in order to create a NER system that requires minimal human intervention. The proposed system identified words and phrases within the text that were potential entities, by using Wikipedia links. With the use of category links and/or inter-article links they categorised the entities found in one of the entity categories. For categorising non-English terms that had an entry in its language's Wikipedia, they proposed two techniques: first, the title of an associated English language article is found by searching for a Wikipedia link beginning with 'en:'. If such a title was found, then the English article was categorised and the assumption that the non-English title was of the same type as well was made. In the opposite case the category information and their English equivalents were used to decide.

Part of the limited research on NER for Balkan languages is a preliminary investigation for the Turkish language by Celikkaya et al. (2013). Their research focused on NER in Turkish texts from different domains (twitter, forums and speech-to-text). The proposed system tokenises the data and then creates training/testing instances using the features extracted from an automatic morphological analysis process. The features employed are the stems of the tokens, their main POS-tags, the case marker and the availability of the proper noun tag. In order to deal with spelling errors, they created a text normaliser that pre-processes the input texts. The CRFs algorithm they use, combines hidden Markov models, stochastic grammars and maximum entropy Markov models. Initial results were promising but also uncovered the difficulty of NER in real (structured and unstructured) data and in Turkish and Balkan languages in general.

Apart from modifying or extending methods that worked well in English to other languages, a significant amount of effort has been invested in extending NER systems to work on different types of text. Most systems were initially trained on corpora containing structured texts, where many NLP problems have already been solved manually [e.g., Stanford NER (Finkel et al., 2005) has been trained on the CoNLL-2003 corpus comprising of formal texts, where sentence splitting and tokenisation has been carried out by human annotators].

However, the rise of social media has brought forward a gargantuan amount of small snippets of casual noisy text such as Twitter posts and short messages. In this novel domain, systems trained on news articles and other types of structured text display an important drop in accuracy (Ritter et al., 2011). This drop can be attributed to the fact that important distinguishing features of named entities in formal texts, are not credible discriminating factors in informal texts. For example, capital letters in tweets are often used for emphasis and do not denote entities with the same probability as in formal texts. In order to tackle the problem of texts with informal language, several NER extensions have been proposed, for tweets (Li et al., 2012; Ritter et al., 2011; Locke, 2009; Liu and Zhou, 2013; Liu et al., 2012), e-mails (Minkov et al., 2005), etc.

2.2 Entity level SA

SA is defined as the task of classifying texts into categories depending on whether they express positive or negative sentiment, or whether they enclose no emotion at all. It has attracted considerable attention in recent years due to its direct applicability in real-world businesses, such as brand monitoring or prediction of election results.

Researchers study the task at different levels of granularity, e.g., document level (Pang et al., 2002) or sentence level (Wiebe and Riloff, 2005; Wilson et al., 2005), or for different kinds of text (Barbosa and Feng, 2010; Hu et al., 2013; Boiy and Moens, 2009), e.g., tweets, movie reviews or forum posts. There are also studies in other languages apart from English (Boiy and Moens, 2009; Zhao et al., 2012; Atteveldt et al., 2008; Abbasi et al., 2008; Abdul-Mageed et al., 2011). Current approaches for SA fall under two main categories. The first category uses supervised machine learning and trains classifiers based on

features extracted from text (Engonopoulos et al., 2011; Mohammad et al., 2013; Pak and Paroubek, 2010; Socher et al., 2013). Then, the trained models predict sentiment classes for unseen text data. Supervised machine learning algorithms (e.g., support vector machines, naive Bayes and maximum entropy) have been proven effective for sentiment classification, but need a considerable amount of manually annotated training data. In unsupervised approaches sentiment lexicons (Hu et al., 2013; Turney, 2002) are created or used in order to determine the sentiment of a text. They use a scoring scheme that considers the number of subjective words in the text and their polarity strength. These methods usually need minimal human effort, but are either only based on words included in lexicons and do not take into account the position of a word, or they need linguistic resources, such as WordNet, which may be language-dependent.

Concerning SA at the entity level, we present in this section a number of proposed methods. ELS (Engonopoulos et al., 2011) is a method for entity-level sentiment classification, which uses CRFs to identify the sentiment of each word in a document and then determine the sentiment for each entity, based on where it appears in the text. The method aims at a fine-grained sentiment classification at the segment level and it takes into account the position of the words in text rather than only their appearance. Given that a sentence may mention more than one entities, a segment is considered a part of a sentence that mentions a single entity. The authors consider the entities mentioned in text and the corresponding segment known. Godbole et al. (2007) assign positive and negative opinions to entities with the use of sentiment lexicons, which are created by small seed lists and expanded through WordNet synonyms and antonyms. The authors assume cooccurence of an entity and a sentiment word in the same sentence to mean that the sentiment is associated with that entity. The prototype system by Nasukawa and Yi (2003) assigns sentiment expressions to subjects of interest, which may be entities as well, and classify the polarity of these expressions as positive/negative. Thus, the proposed method operates at the level of text fragment including a subject and not at the document level. The sentiment expressions associated with a subject are recognised with the use of a dictionary of sentiment expressions. Sentiment expressions consist of subjective words (positive/negative), part-of-speech tags of these words and a few syntactic patterns determining the subject and the object of the expression. Hence, the dictionary includes such sentiment expression patterns and the authors attempt to match these patterns with new text segments.

Aspect SA is quite similar to SA at the entity level as it aims to assign a sentiment class to a specific target, even if this is a product feature. Hu and Liu (2004) propose a method for identifying product features in product reviews and then classifying opinion sentences about them into positive and negative. They define opinion sentences as the sentences that contain one or more product features and one or more opinion words. Opinion words are adjectives near the identified product features. To classify the polarity of these adjectives into positive and negative, the method uses a small seed list of words and WordNet. The seed list consists of known positive/negative words, whereas WordNet is used to provide synonyms and antonyms of words. The polarity of an entire opinion sentence is determined by the polarities of its opinion words. Popescu and Etzioni (2005) present OPINE, a system that also extracts product features from product reviews, recognises opinion phrases concerning these features and evaluates the polarity of the phrases as positive or negative. In order to identify the beginning of opinion phrases in sentences where product features are found, the authors apply extraction rules based on syntactic dependencies. The polarity of the identified opinion phrases is determined through a relaxation labelling technique that finds the semantic orientation of words in the context of given product features and sentences. A paper by Blair-Goldensohn et al. (2008) aims at recognising aspects of services, e.g., for restaurants or hotels, and then summarising the positive/negative sentiments expressed about these aspects. So the task of the paper is divided into three subtasks:

1 identification of subjective sentences and classification of these sentences as positive/negative

2 identification of aspects of services

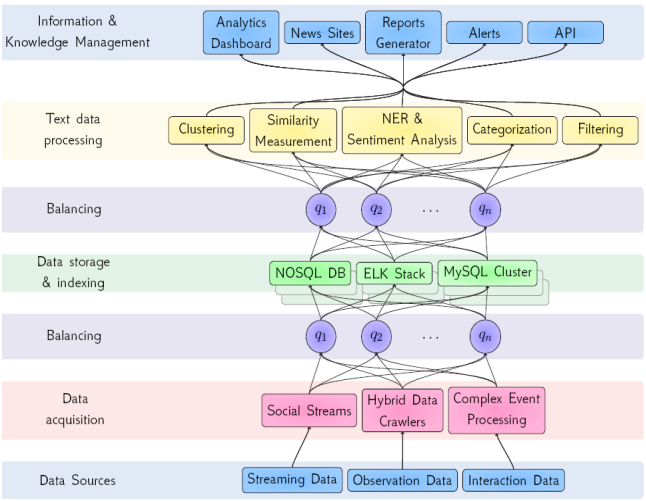
3 summarisation of opinions for the aspects of the service.

The first subtask combines both the creation of a sentiment lexicon based on basic seed lists as well as synonyms/antonyms of WordNet, and the training of a Maximum Entropy classifier with features based on the aforementioned lexicon and the overall review rating provided by the user.

3 System design

The architecture of the platform (see Figure 2) consists of a set of mechanisms that implement the whole text processing pipeline starting from data fetching from the distributed sources, storage, summarisation and analysis of textual data and finally synthesis and representation of the extracted knowledge back to the end user.

Figure 2 Platform architecture (see online version for colours)



On the bottom layer of this architecture, we have the data acquisition modules. A large number of distributed crawlers accesses and processes content from different type of sources (news sites, blog feeds, social media APIs, etc.) in order to collect the desired data (comments, article content, etc.). The crawlers differ from source to source, because the data sources can vary from social media to news portals and forums. For example, data acquisition from social media (streaming data) is mainly done using the 'streams' they provide via their APIs. A similar strategy is followed for feeds and channels of cloud based CMS or blog services (e.g., blogspot) that helps us get a large amount of 'clean' data from web sources. In the case of news sites and portals, data fetching is performed by a complex system of hybrid web crawlers. The news crawlers combine manually defined parsing rules and automatic extraction of page wrappers (data extraction rules), which allow quick collection of specific content (e.g., article title, body, media) from multiple sources (Varlamis et al., 2014) (observation data). Finally, we analyse information concerning how our users interact with the collected content and services (interaction data), and deliver personalisation services and advanced analytics and alerts to the users. Personalisation is achieved through a combination of usage and content analytics and personalised alerts through a combination of NER, SA and complex event processing based on rules.

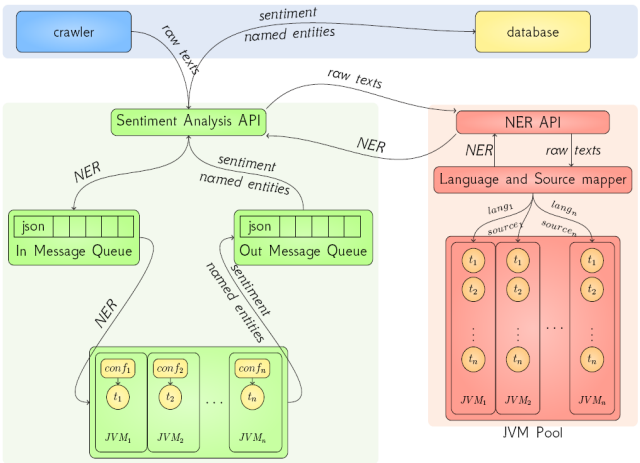
The next layer comprises the data storage mechanisms. Because of the different nature of information we collect and extract (i.e., full text, metadata, entities, sentiment), information is stored and indexed on multiple different types of databases. This allows us to get the maximum out of each DBMS and be able to quickly deliver the appropriate information services. For this reason, we utilise both SQL and no-SQL databases. In both cases we use database clusters in order to support the huge amount of data that we collect on a daily basis. The SQL databases (RDBMS cluster) are used for storing the collected data and all related metadata. The noSQL database is used for storing and indexing all the information necessary for supporting fast search and fast data analysis. With the Elasticsearch, Logstash and Kibana (ELK) Stack we are able to define complex analytics, which are applied in real-time on the data we collect. At the same time, we provide full access to the complete set of information we collect using the RDBMS storage.

One layer above, are the language-agnostic text mining modules (NER, SA, text clustering and classification), which undertake the detailed analysis of the collected textual data and the filtering and association of extracted information. The processing pipeline applies a set of algorithms to the texts of a stream (e.g., filtering, data cleaning and validation, similarity measurements, clustering and categorisation) and extracts a lot of useful information that is stored in the RDBMS and NOSQL cluster. The first two procedures, are done together as we expect the results of one of them (SA) to be applied to the results of the other (NER). Text clustering and classification is critical for our platform, since it allows removal of redundant information (e.g., repeated/copied news articles) and provides a better organisation of news content. After the completion of this procedure, we expect to have combined, connected and 'clear' data, which can be used for presentation or for analytics.

The top layer of the system comprises the modules that perform an even deeper analysis of the extracted and organised information and produce more qualitative results for the end-users of the system. These modules deliver the information and services to be presented to the end-users, such as named entity mentions analytics and reports, alerts, an API for news and related content, which is used by our platform but also by third party services.

This paper focuses on the NER and SA module, so from this point forward we continue with its description. In the visualisation of the SA and named entity extraction module presented in Figure 3 we can distinguish three main components. At the top of Figure 3 we have the main infrastructure, in blue background, comprising the subsystems that crawl the web for content, perform the necessary preprocessing (cleaning, filtering, clustering and classification) of text streams' contents and store the results to the data storage level.

Figure 3 The pipeline for NER and SA (see online version for colours)



On the bottom right of Figure 3 (in red background) is the NER module. The system accepts a batch of raw texts combined with metadata about the batch, such as language and source, in JSON format. Subsequently, it performs sentence splitting, tokenisation and NER in this order. The model contains information about which model to use for each language and task, since different languages and types of text are better processed using different models (e.g., trained with documents in the same language and for the specific task). Each model is loaded once, runs on a separate Java virtual machine (JVM) and employs a configurable number of threads. The results are returned in JSON format and contain information about sentences, their positions in the text, the entities and their positions in the sentences.

On the bottom left of Figure 3 (in green background), we have the SA module. The SA module takes as input a batch of raw texts, as well as their language and source, in JSON format. It first forwards the input data to the NER module, in order to obtain the sentences of the text, the entities mentioned in each sentence and the types of the entities (e.g., person, location). After the response by the NER module, the SA module identifies the sentiment expressed for each entity mentioned in the sentence. Its output is also in JSON format and contains the sentences of the input text, the entities mentioned in each sentence along with their types, the sentiment class for each entity in a sentence and the overall sentiment class of each sentence.

The architecture of the SA module consists of two main components. The first component is the web service that receives requests by users and communicates with the NER module. The second component is an application that implements the SA method and provides sentiment predictions for input data. The communication between these components is done through messaging. Specifically, the web service publishes messages, including input text and the response by the NER module, to a message queue. The application threads act as listeners to the queue and process the messages to produce sentiment predictions. Finally, the output JSON is inserted in another queue and is returned back to the user.

4 Implementation

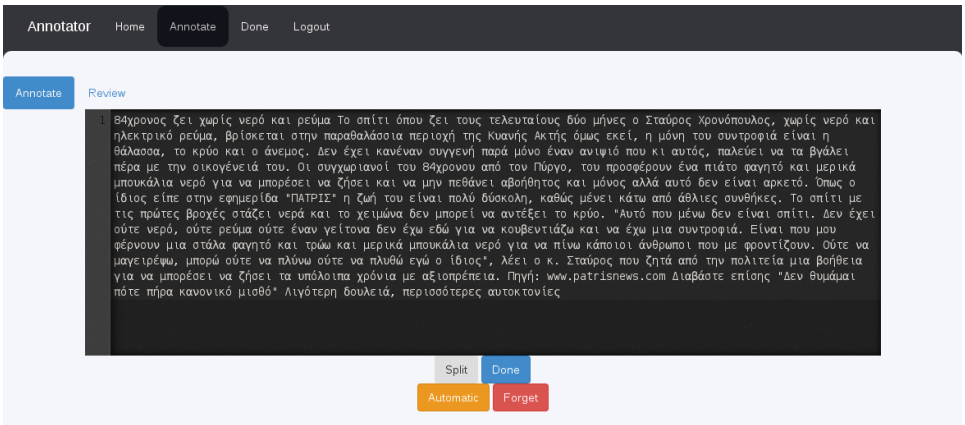
In this section we elaborate on the methods, algorithms and models that we employ for detecting named entities in texts and assigning sentiment labels to them based on text segments associated with them. We also provide details about the software, linguistic tools and resources that we use to implement the two pipelines.

4.1 Implementation of NER approach

For the extraction of named entities from text, we capitalise on the use of an open source library for natural language processing, namely OpenNLP. Compared to other solutions such as Stanford NER or NLTK NER, OpenNLP better fits the architecture, since it is written in Java, it is open source and provides a common suite for many text analysis tasks. OpenNLP is a machine learning based toolkit for natural language applications. It supports the most common NLP tasks, such as sentence segmentation, tokenisation, part-of-speech tagging, named entity extraction, etc. OpenNLP provides two alternative machine learning methods, maximum entropy models and perceptrons (neural networks) depending on the NLP task.

In simple words, both methods solve complex problems as classification tasks. They use an extended set of features that is extracted from the textual content and for this reason, for each task, they require a training set which comprises properly annotated documents or sentences. For example, the sentence segmentation model is trained using a set of sentences, one sentence per line, and trains a probabilistic model that learns whether a punctuation mark is most probably a sentence splitting mark or not. A similar approach is followed in other tasks.

Figure 4 Online application for creating the training corpus for the sentence splitter and the tokeniser (see online version for colours)



In the case of news processing, we use OpenNLP in three different tasks: sentence splitting, tokenisation and NER. Since we want to quickly deploy our solutions for different languages, it is important for us to be able to quickly create the appropriate (annotated) training corpora for any new language. We attempt to solve this issue, using massive open source data and collaboratively created text corpora, which are usually semi-structured and rich in semantics, so we can easily transform them to training data for our models. For the cases that these data are not adequate, we have created a set of annotation tools, that allow human annotators to create additional training samples. In Section 5 we provide more details on the performance of models which have been trained using automatically and manually annotated datasets.

4.1.1 Sentence splitting and tokenisation

Sentence splitting and tokenisation along with part-of-speech tagging are the most popular pre-processing tasks that precede any other text mining or NLP task. They play an important role in the complexity of the solution that follows them, since they limit the context to a sentence level and they affect the quality of any following task since they split each sentence into lexical tokens (words) and allow for word-based approaches to be applied.

Although OpenNLP provides pre-trained models for sentence splitting and tokenisation in English and a few more languages, it still lacks support for Balkan languages or even Turkish. One may think that sentence splitting is similar across languages. This is not the case, since for example the semicolon in English (;) works as a question mark in Greek. Even worse, the period symbol (.) can denote sentence termination or not depending on the text that appears in front of it; for example in Mr., Dr. or B.B.C. it is simply part of the abbreviation and not a sentence termination mark. The list of abbreviations differs across languages and is hard to compile one list for each language. Finally, creating a single model for a language is not always correct, since the same language (e.g., English) may have different syntactic rules in different contexts. This is the reason that the Stanford POS Tagger, recently released a second POS tagging model for informal English (e.g., for twitter texts) which has been trained on different text corpora.

The advantage of machine learning is that it allows us to create any model using a properly annotated training corpus. For example we can build a sentence splitting model by creating a set of sentences (one per line – in the case of OpenNLP) that contain punctuation marks. Finding or automatically creating such a dataset for a language is not an easy task. For this reason, we developed a module that takes as input *Html* content from news articles and creates a training dataset for sentence splitting, taking advantage of HTML template information and the use of specific tags (e.g., breaks and paragraph marks). We also developed a semi-automatic solution, an online annotator for creating training examples for sentence splitting and tokenisation from raw text. The online annotator (see Figure 4) takes a raw text as input and allows users to add sentence splitting and tokenisation markup. It also offers an auto-annotate mode, which employs the models trained so far and allows users to correct any errors.

4.1.2 Named entity recognition

OpenNLP takes an annotated text corpus as input and creates a NER model as output. Any raw text must first be converted to the OpenNLP name finder training format, which is one sentence per line. The sentence must be tokenised and contain spans which mark the entities. If the training file contains multiple types then the created model will be able to detect multiple types. However, it is recommended to only train single type models. The most widely supported entity types in NER are person, location and organisation. In the case of the analytics services, we also examine products as entities. As a result, we have to create four sets of training texts for each language (one model per entity type).

For the creation of training corpora we implemented different alternatives. Others use crowdsourcing open data and manually assigned semantics, others are based on regular expressions and others are based on manual annotations. In the first two cases, our input is Wikipedia and DBpedia. The advantage of this combination is that a Wikipedia corpus is available in many languages and comprises many semi-structured texts, where named entities are properly annotated within the text. For example in the following sentence, the double braces denote a link to a page for this entity, while the pipe symbol separates the actual entity from the string which represents it in the text.

> "iPhone" is a line of [[smartphone]]s designed and marketed by [[Apple Inc.]] They run Apple's [[iOS]] mobile operating system. The [[iPhone (1st generation)|first generation iPhone]] was released on June 29, 2007

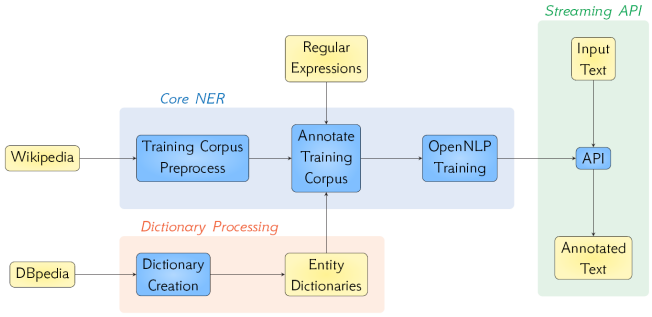
In addition to this, DBpedia provides information on the type of these entities. More specifically, DBpedia contains a long list of persons, locations and organisations for many languages and each entity can be traced to Wikipedia sources. In the third case, manual annotation is performed for languages that do not have a Wikipedia and DBpedia corpus or for the case when we want to add more training instances to the model.

4.1.2.1 Training corpus creation

When the NER mechanism is developed for a new language, we care for a quick deployment with minimum resources spent on text annotation. For this reason, the mechanism that creates the *wikimodel* fetches the DBpedia entities for each of the types of person, location and organisation and adds to these lists a list of products manually created for the target language. The mechanism then processes all sentences of Wikipedia that contain these entities. With a set of regular expressions and text formatting instructions we are able to automatically convert

the Wikipedia corpus to an input annotated training corpus for OpenNLP. Then using OpenNLP training we can create the four models for the target language. The whole pipeline is depicted in Figure 5.

Figure 5 Detailed view of the NER training pipeline (see online version for colours)



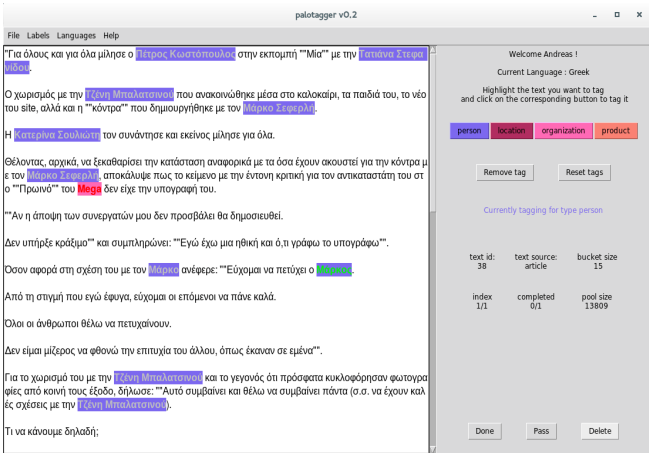
The advantage from the use of Wikipedia manual annotations is that they provide entity mentions with high accuracy, since all the entities are referenced with a specific syntax in wiki markup language. However, the recall is lower. For example, while an entity may be mentioned dozens of times within a Wikipedia page, some of the mentions are not marked properly in the Wikipedia source code. As a result we have many false paradigms (false negatives) in the training set, which affect the model performance. Since we only keep the sentences that contain explicitly marked entities, we reduce the number of false negatives. In order to increase the amount of training data from Wikipedia we used two alternatives:

- *Regex model:* The first alternative increases the recall of entities in Wikipedia pages by using regular expressions in order to markup texts. In simple words, the method locates consecutive words that start with a capital letter, e.g., *Simon P. Laplace* or word sequences in which stopwords may occur between names, e.g., *Joan of Arc*. The method creates a list of potential entities and then examines this list against the lexicons. We perform a case sensitive exact search in the lexicons using any accents supported by the language. This is important since in some languages the use of accents may result in different words. For example Αθήνα is the name of the city of Athens in Greek, whereas Αθηνά stands for the female name Athena.
- *Ngram model:* The second alternative performs a looser matching of potential entities against lexicon entries. It uses a hybrid trie-index structure for the lexicon entries, in which each multi-keyword term in the lexicon is split into the words that it contains. In the first level of the index we put all the words that appear in the beginning of lexicon terms. In the second level, the words that appear second in a lexicon term and so on. In a word at any level in the trie-index, we keep a Boolean that indicates whether the word is the last word of a term in the lexicon (terminal word). The ngram matching algorithm looks at the first level of the index and if it finds a match then continues to the lower levels until the longest matching is found. The last word matched must also be marked as a terminal word. For example, if we have 'John Doe' and 'Mary Brown' in the lexicon the ngram model will also match 'John Brown' and 'Mary Doe'.

  We must remind here that the method is used for marking entities in Wikipedia texts and thus creating an automatically annotated text corpus. So it is reasonable to mark entities that do not necessarily correspond to Wikipedia entries but resemble to proper names for locations, person or organisations. As a result, the method further increases recall of proper names detection in Wikipedia, since it positively detects more entity references, but also introduces some false positives in the training dataset.
- *Manual annotator:* In order to further improve the quality of our training dataset, and consequently the quality of the NER model, we developed a manual annotator.

  The annotation tool, illustrated in Figure 6, automatically performs sentence splitting and tokenisation, using the models we presented in Section 4.1.1 and allows users to quickly tag entities of different types in the text. It also allows users to assign sentiment (positive, negative or neutral) to the tagged entities. The annotator offers an automatic annotation option, which employs the models we trained with Wikipedia documents and further accelerates the annotation process. As a result, we are able to annotate a few hundreds of articles and create an annotated corpus for NER and SA comprising thousands of sentences and entities within a few hours.

Figure 6 The NER and sentiment annotator application (see online version for colours)



4.1.2.2 NER model usage

Any new text fetched by the crawling mechanisms is processed by OpenNLP and annotated for entities of all types. Since OpenNLP does not create a single model for all entity types, we created a single interface that integrates the answers of the different entity models. We modified the

OpenNLP code in order to apply the four models in a row and consequently merge the resulting entities. When two entities of different type overlap, we keep the one with the highest probability or in the case of a tie, the lengthiest one. All the implementation was wrapped in a jar file which was further accessed using a the Python library *jnius* by the NER service which was implemented in Python.

In order to further increase the recall of the machine learning approach, we passed the texts through a second filter which automatically marks any entity found in the text that belong in the lexicon (using exact match). This method adds an overhead to the pipeline since every text is scanned a second time in a token-by-token manner, but it significantly adds to the NER model recall.

4.2 Implementation of SA approach

SA is treated as a classification problem into three classes: positive, negative or neutral. Neutral is used for factual references of an entity, whereas positive and negative classes are used for opinions. Although the method receives sentences as input, it operates at the entity level, meaning that if the sentence mentions more than one entities, the system will assign a sentiment class for each entity and not just for the entire sentence. The approach followed by the SA module consists of two main steps:

1 split of the sentence to segments, with each segment mentioning a single entity

2 assignment of each segment to a sentiment category.

4.2.1 Split of sentence to segments

We assume that the entities of the sentence are known and in this case provided by the NER module of the pipeline. We split each sentence into segments spanning from entity to entity, with each segment containing a single entity. Suppose we have the sentence 'Robbie Williams was impressive in Athens yesterday, whereas the Black Keys concert we went to a month ago was mediocre'. The system will automatically split this sentence to the segments displayed in Figure 7. We observe that the first segment spans from the beginning of the sentence to the second identified entity. The second segment starts at the second identified entity and extends until the third entity, whereas the third part is from the third identified entity up to the end of the sentence. We perform SA to each of the generated segments in order to predict a sentiment class for each entity, as displayed in Figure 8. If the sentence involves only one entity, we perform SA on the whole sentence. The idea behind this technique is that subjective words concerning an entity are probably in small proximity from the entity reference. The same concept is behind techniques that use a fixed window size to define phrases around a target. Because we deal with different types of documents, it is hard to choose a window size that gives good approximate results for both short and longer sentences. Thus, we prefer to use the distance between entities to set approximate boundaries between the segments concerning different entities in a sentence. Syntantic analysis is probably the most accurate way to discover dependencies between words and entities. Nevertheless, it requires syntactic parsers, which are not available in every language and may need different tuning to address both formal and informal text effectively. Given the multilingual perspective of the platform and the heterogeneity of the processed documents, the requirement of syntactic parsers could be quite restrictive.

Figure 7 Segments generated from a sentence

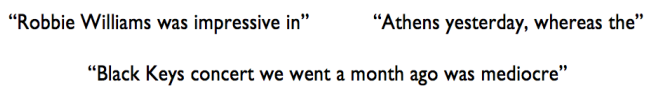
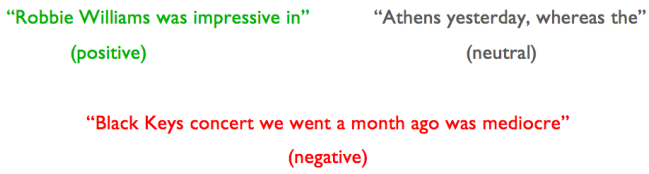


Figure 8 Sentiments generated for each segment (see online version for colours)



4.2.2 Prediction of sentiment

The presented approach for sentiment classification uses a supervised machine learning algorithm and therefore it needs training data to be initialised. However, text data cannot be processed directly by machine learning algorithms and we need to transform them to numeric feature vectors. In many cases to achieve good performance, we also need to perform preprocessing or normalisation techniques on data. We describe the techniques/algorithms used in each step of our system in the subsections below.

4.2.2.1 Data preprocessing

Negation is an important aspect of the semantics of a text. We aim to identify negated context using patterns, which start with a negation word and end with a punctuation mark. The definition of negated context is based on Mohammad et al. (2013). For every sentence we store whether a negated context is identified or not. At the end of the process, we use this information to determine whether to alter the sentiment label from positive to negative or vice versa. We also tried more sophisticated techniques, which involved patterns of negation words and part-of-speech tags [e.g., (do not)<verb>], and reversed sentiment polarity only if the pattern negated a subjective word found in training data or lexicons used for feature extraction described below. Although these techniques increase average F1 by about 2%, they also require three times more processing time. As time performance is critical for the platform, especially during rush hours, we prefer not to increase processing time considerably for negation identification and use a more coarse-grained method.

Articles from news websites are usually well-structured and without typos, unlike more informal text, such as social

media and forum posts, which may be terse or contain specific abbreviations and misspellings. Prior to creating numeric feature vectors, we preprocess data to remove noisy elements and reduce the counter effects of inflection and omission of stress marks. Most of the noisy elements are met in informal text, such as social media or forum posts, and are less present in well-written documents, such as news websites' articles. We consider the following elements as noise:

1 URL links, because we do not follow them and analyse their content.

2 Mentions of users and the abbreviation RT in case of Twitter. RT means that a tweet is a retweet of another one.

3 Stop words, which are extremely common words including articles and pronouns.

In many languages, such as Greek, accent marks are used over vowels to denote where a word is stressed. However, users of social networks and forums often forget to add these marks or they add them at the wrong syllable. To avoid misspellings due to this reason, we prefer to also remove accent marks from words. Moreover, repetitive vowels are reduced to one and repetitive consonants are reduced to two. Another preprocessing step is the grouping of emoticons in two categories, positive and negative, and the replacement of the members of each category with a single emoticon. Positive emoticons are replaced with ':)', whereas negative are replaced with ':('.

Last but not least, we use stemming to address the inflective nature of languages. In some languages, such as Greek, verbs and adjectives are inflected for person, number and gender, which affects mostly the suffixes of the words. This variance in suffixes affects the effectiveness of produced features and as a consequence the classification performance also decreases. Therefore, we assume that stems of words are usually enough to indicate the sentiment of a text. We use the implementations of stemmers provided by Apache Lucene (https://lucene.apache.org/).

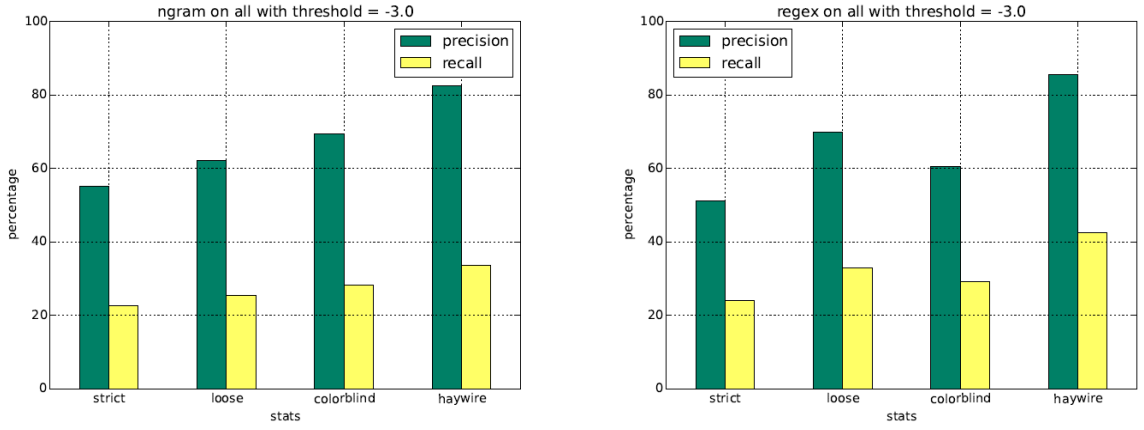
4.2.2.2 Feature vectors extraction

Numeric feature vectors consist of bag-of-words and lexicon-based features. Bag-of-words features are unigrams extracted by the vocabulary of the training data. Each feature of this type has value 1 or 0 depending on whether the unigram is present in the instance or not. The values for the lexicon features are determined using two lists of subjective terms. The first one includes positive words, whereas the second contains negative words. Two lexicon features are extracted based on the number of words in the instance from the positive and negative list. A third feature of this type is also added storing the difference between the values of the former lexicon features. As the number of bag-of-words features is usually quite large, we use information gain to keep a subset of the most informative ones. The implementation of information gain algorithm is provided by the Weka Data Mining software (Hall et al., 2009).

4.2.2.3 Sentiment classification algorithm

After preprocessing of data and extraction of feature vectors, we train a classification model. Training examples are sentences or short texts, such as tweets, labelled by human annotators with the dominant sentiment class. By dominant sentiment class, we mean the strongest opinion expressed for an entity in the sentence. For example if we have a sentence with two entities, a positive opinion for the first entity and factual information for the second, the annotators should label this sentence as positive. In cases where all expressed sentiments are of equal strength, we base the sentiment class arbitrarily on the first entity. As a result training instances are annotated at the sentence level. However, the sentiment predictions generated by the trained model for unseen data are at the entity/segment level, as each sentence has been first split to as many segments as the number of entities identified in it by the NER module.

Figure 9 Results from the use of the automatically trained models, (a) recall and precision for the ngram automatic annotation model (b) recall and precision for the regex automatic annotation model (see online version for colours)



(a) (b)

For the SA module we chose SVM and specifically the liblinear implementation (Fan et al., 2008). Liblinear is a linear classifier. We define the regularisation parameter (*C*) using cross validation. When the classifier finishes giving predictions for the input data, we check for each instance whether negation patterns were identified during preprocessing. If negation was identified in an instance and its sentiment prediction is positive, we reverse it to negative and vice versa.

As the prior polarity of words in lexicons is verified manually, we also train a model in which the weights for lexicon features are equal to two times the maximum weight of the rest of the features. The purpose of this amendment is to give lexicon features a more important role in sentiment prediction, since the polarity of unigram features extracted from training corpus may be biased due to specific examples. We will refer to the model described above as SA model and to this one as SA_boost model.

4.2.2.4 Language configuration

Given the preprocessing and feature engineering techniques, which were described above, specific resources are important when the SA module needs to handle a new language. The only requirement is the existence of training data, in order to create a trained model for SVM. All other resources are optional, as they may be unavailable for a language and difficult to construct from scratch. Optional resources include stemmers, subjective lexicons, preprocessing code and code for negation identification. Preprocessing and negation identification code should be straightforward, nevertheless the user is still able to omit them. The user configures these settings through a configuration file. Then it maps this file to a language in another file that acts as an index for languages and configuration files. Stemmers, preprocessing and negation identification must be Java classes that implement a corresponding interface, and are stored in different packages, with one package per language. Lexicons are text files including one word per line and possibly a score assigned to each word separated with a tab. These scores denote the strength of the sentiment expressed by a word. In a simple scheme where there is no information about the strength of the sentiment, the scores may have only two values, 1 for positive words and –1 for negative ones.

When texts are submitted, the SA module checks the language metadata of the input and uses the index to map the language to a configuration file. By reading the respective configuration file, the module is able to apply the correct preprocessing and feature engineering techniques on input texts. The applicability of an earlier version of the SA module to other languages is also presented in experiments in Makrynioti and Vassalos (2015).

5 Evaluation

In this section, we present some initial results from the application of our modules to validation datasets.

5.1 NER performance

This section tests the performance of the NER module against an evaluation dataset, which comprises texts in Greek collected from news sites (100 texts), twitter (102 tweets) and Facebook (113 posts). This resulted to 1,366 sentences in total. The performance of the models, which have been created using the automatically annotated Wikipedia corpus and DBpedia (regex and ngram models) are depicted in Figure 9. The four different pairs of columns in the results correspond to the matching method we used for evaluation. The *strict* matching method counts a hit when the correct entity was marked exactly in the text and with the correct type. The *loose* method counts a hit when the entity was partially marked (e.g., only the last name of the person, or the entity and a neighbouring word) and the type was marked correctly. The *colourblind* method counts all exact entity matches that probably are assigned to wrong entity types. This metric is useful, since it is easy to confuse between a company and a product or in some cases between a name and a location. Finally, the *haywire* method counts all partial matches even when the entity type detected was not correct. From the results, we can see that in all cases, both methods have a fair precision but significantly low recall. The low recall is mainly due to the partial annotation of Wikipedia entities, which was not fully treated by the regex and ngram annotation methods.

In order to compare the performance of automatically created and trained models to that of manually annotated corpora, we trained the so-called *human* NER models using a manually annotated set comprising approximately 2,000 texts from all types of sources. Finally, we managed to boost recall without losing in precision, by adding the lexicon-based annotation, which marks any entity that has not been marked by the models but appears in the lexicon. We took advantage of this increase in recall and were more strict in the use of the machine learning by adjusting a threshold parameter from –3 (default) to –1. This threshold defines which assignments will be kept in each iteration of the MaxEnt algorithm. Less negative values correspond to more confident assignments. Figure 10 presents the results of the human model with and without the addition of a lexicon. From the results we can see that the use of manually annotated corpora raised the precision above 80% in some cases and at 70% in the case of strict matching. The use of lexicons helped us improve the recall of our methods by almost 10% in all cases.

A comparison across the different types of text sources reveals that the performance on Facebook posts and Twitter texts is worst than that of articles, which can be explained by the frequent use of informal language in the two types of media.

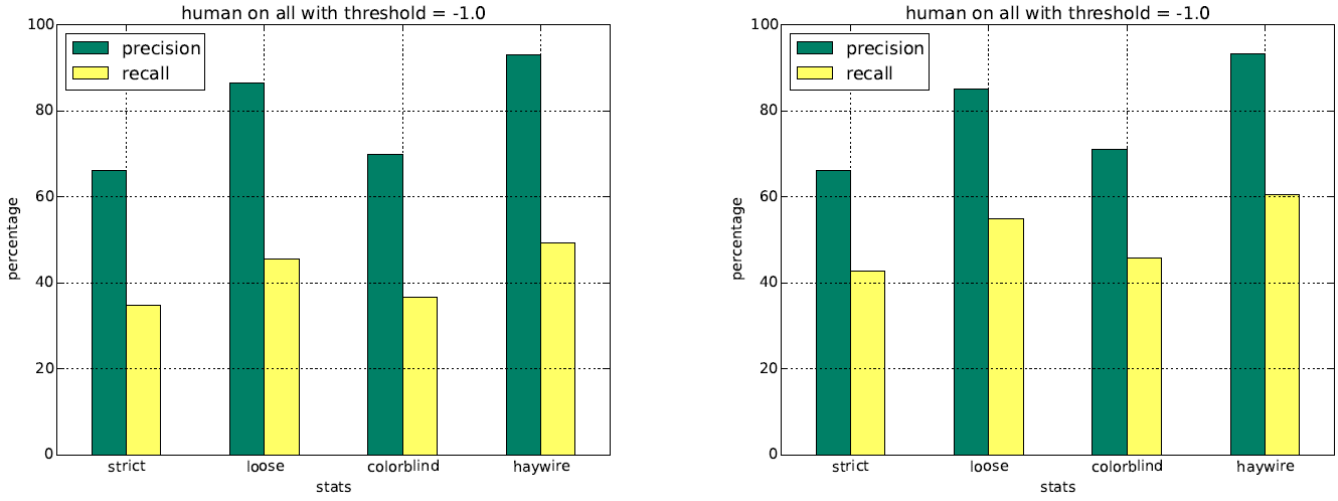
5.2 SA performance

We also present some initial results from the application of the SA module on short texts, such as tweets, and larger texts extracted from sites and blogs. We use two validation datasets, one consisting of 100 sentences from tweets

and short Facebook posts, and another comprising of 97 sentences from news articles and blog posts. The former dataset mentions 141 entities, whereas the latter includes 145 entity references. Note that the number of sentiment predictions produced by the classifier corresponds to the number of entities mentioned in the dataset, so in these experiments it would be 141 for the dataset of sentences from short posts and 145 for the dataset of sentences from larger texts.

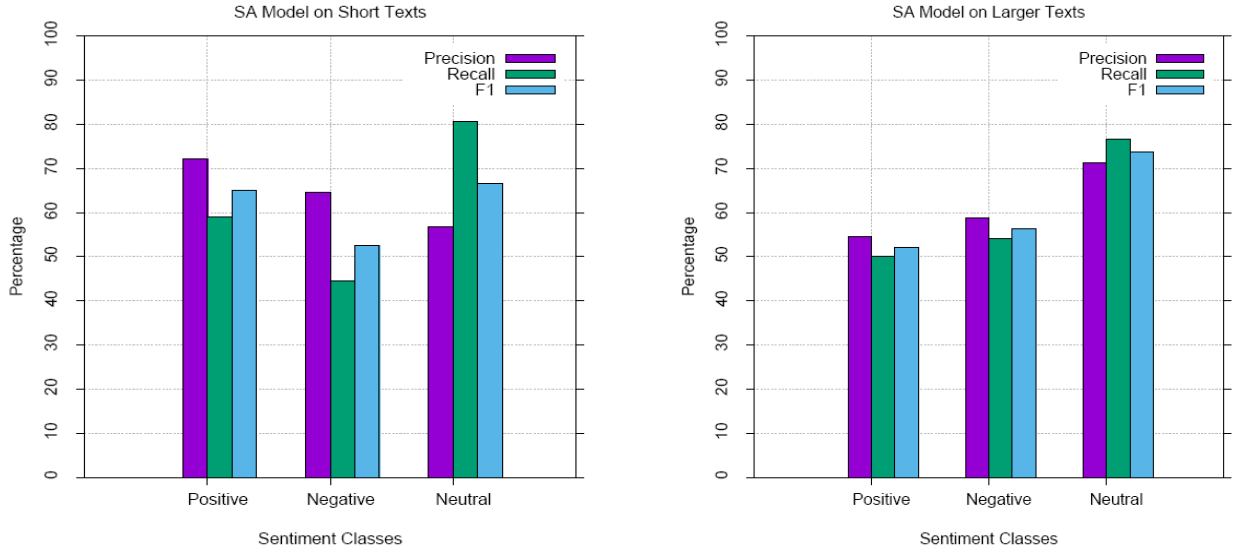
The performance of the SA model with equal weights for all features regarding short texts from Twitter and Facebook is depicted in Figure 11(a). The three different columns in the results correspond to precision (purple), recall (green) and F1 (light blue) evaluated for each sentiment class. We can see that precision of positive and negative classes is much higher than neutral class, whereas the results for recall are the exact opposite. This behaviour indicates that many positive and negative predictions of the module are precise, whereas a number of subjective tweets are recognised mistakenly as neutral. Experiments on sentences from sites articles and blog posts are presented in Figure 11(b). Here we can observe a drop in precision for positive and negative classes, as sentiment is expressed more implicitly in sites and blogs than in social media.

Figure 10 Results from the models trained on human annotated corpora with and without lexicons, (a) recall and precision without lexicons (b) recall and precision with lexicons (see online version for colours)



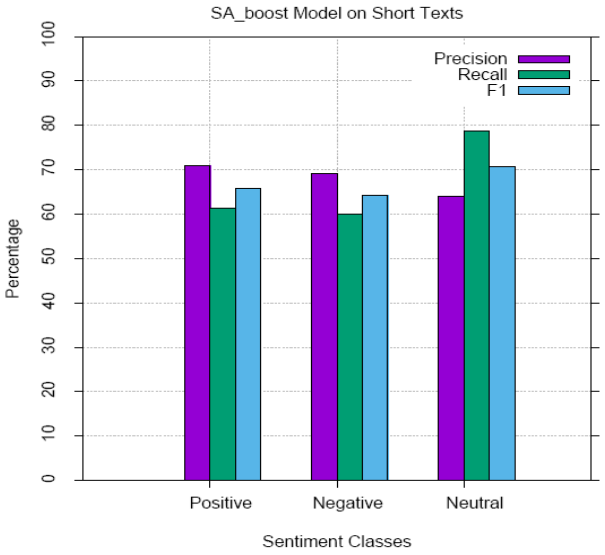
(a) (b)

Figure 11 Results on short and larger texts with the SA model (all weights are equal), (a) results on tweets and short Facebook posts (b) results on site articles and blog posts (see online version for colours)

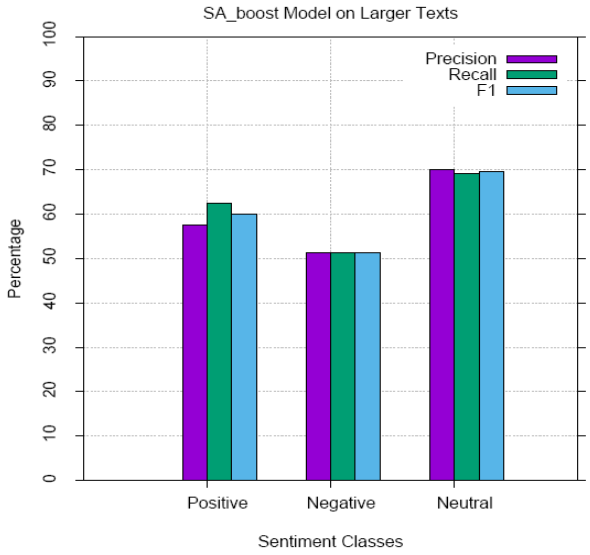


(a) (b)

Figure 12 Results on short and larger texts with the SA_boost model (increased weights for lexicon features), (a) results on tweets and short Facebook posts (b) results on site articles and blog posts (see online version for colours)



(a)



(b)

In Figure 12(a) we present the performance of the SA_boost model with boosted weights for lexicon features on tweets and short Facebook posts. The increase in weights of lexicon features results in significantly better precision and recall for negative class. The F1 for the other classes is also a bit higher than in Figure 12(b). On sentences from articles the boost of lexicon features does not affect performance so much. We can see an increase in all metrics for positive class, but this does not apply for the other two classes. Since sentiment is usually more implicit in such types of text, it is expected that words with known prior polarity may not help that much.

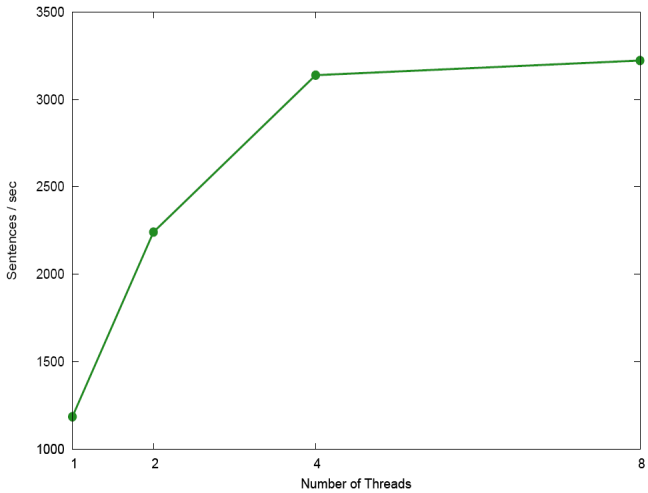
5.3 Time performance

In this section we measure the time performance of our pipeline, providing evidence separately for the NER and SA modules. We report on the time needed for training the model and the time needed for processing an increasing amount of content. The datasets that we used are in different languages (i.e., English, Greek, Serbian and Turkish) in an attempt to demonstrate the ability of our platform to handle multilingual content and also measure the effect of the size of models in the performance of the system. Finally, we ran our experiments on a Linux server with Intel Xeon CPU at 3.50 GHz with four cores, and 32 GB memory, and we use different setups that either exploit or not the multithreading capabilities of the pipeline.

5.3.1 SA time performance

In order to evaluate the performance of the SA module in training models of various sizes, we run experiments using datasets of different sizes in three different languages: English, Greek and Serbian. We used a small manually annotated dataset in Serbian of 1,572 sentences and two much larger datasets for the Greek and English languages comprising 1,000,000 and 1,600,000 instances respectively. The Greek dataset was automatically created to be used only in time performance tests by assigning a sentiment class randomly to each instance, whereas English data were downloaded from the Sentiment140 (http://www.sentiment140.com/) API and include only positive and negative tweets. Table 1 depicts the time required for creating SA models from the respective training data in the three languages, which is less than a minute for the small corpora, and reaches a few hours in the case of the large training corpus in English and Greek.

Figure 13 Response time per number of threads (see online version for colours)



In the second experiment, we test the time performance of the SA module in predicting sentiment for new data. We used a Greek model extracted from 8,000 manually annotated instances and tested on 30,694 sentences. In Figure 13 we present the number of sentences processed by

the SA module per second (module's throughput). This number is strongly affected by the number of available processing cores, so we display results using a varying number of threads. In the results, we observe that even in the single-threaded mode, the SA module is able to process a large number of sentences per second (more than 1,000 sentences per second). We observe that throughput is more than three times higher when using four threads on a machine of four CPU cores compared to using a single thread. However, depending on the available CPU cores, after a number of threads the overhead of creating and managing new ones increases considerably. Thus the number of sentences/sec processed by eight threads increases a little compared to four threads on a machine with four CPU cores, but the difference is significantly higher when going from one to two threads or from two to four threads.

Table 1 Training time on data from three languages

Language (number of instances)	*Time (hours)*
Serbian (1,572)	0.004
Greek (1,000,000)	2.78
English (1,600,000)	4.78

5.3.2 NER time performance

In order to evaluate the performance of the NER module, we ran similar experiments using corpora of different sizes for training and annotation. In the case of the NER module, we train four separate models for each language in order to identify locations, organisations, products and person names. For the experiments we use Wikipedia as a text source and DBpedia as a lexicon source as explained in Section 4.1.2 and we evaluate the system performance for the English, Greek and Turkish language. All the times reported in Tables 2 and 3 are measured on the same machine as before but using the single thread version. This means that the throughput can be three to four times bigger if we take advantage of the four CPU cores.

The process for the creation of NER models as explained in Section 4.1.2 begins with the annotation of the Wikipedia corpus with the entities of a certain type as listed in DBpedia. Then from all the sentences in the Wikipedia corpus we keep only the sentences that contain at least one entity of this type and using this set of sentences we train our model, which is specific for this language and the respective entity type. This process is repeated for the four entity types of interest.

It is obvious from the numbers presented in Table 2 that the size of the English Wikipedia corpus is much bigger than that of other languages. Furthermore, we see that the number of entities and sentences extracted as well as the time it takes to label the sentences is proportional to the size of the corpus. As a consequence, the processing time and the size of the models for the English language are much bigger. Table 2 presents the number of entities extracted from DBpedia for each type of entity and each language, the number of Wikipedia sentences that contained an entity of that type, the time needed for training the respective model using OpenNLP and the size of the binary model that was created. We trained the model on all the sentences extracted to check scalability, however, in practice especially for the English dataset, it would be wise to feed the training algorithm with an increasing subset of the data until the model performance no further improves.

Table 2 Statistics for NER models built from Wikipedia for English, Greek and Turkish

	Location	*Organisation*	*Person*	*Product*
Number of entities added to lexicon	617,537	202,636	544,497	41,847
Number of sentences extracted	3,574,287	2,520,794	2,684,434	145,227
Model train time in seconds	9,499.9	7,264.5	8,155.1	366.9
Model size in megabytes	68	48	64	16
(a) English				
	Location	*Organisation*	*Person*	*Product*
Number of entities added to lexicon	5,736	1,724	5,259	877
Number of sentences extracted	103,522	14,209	19,484	2,051
Model train time in seconds	327.9	35.5	55.2	4.5
Model size in megabytes	5.3	0.94	1.5	0.15
(b) Greek				
	Location	*Organisation*	*Person*	*Product*
Number of entities added to lexicon	10,511	6,071	22,361	643
Number of sentences extracted	102,086	56,960	73,154	1,041
Model train time in seconds	242.6	103.9	148.8	1.3
Model size in megabytes	4.7	2.5	4.1	0.06
(c) Turkish				

Table 3 Wikipedia statistics for English, Greek and Turkish Wikipedia

	English	*Greek*	*Turkish*
Corpus size (gigabytes)	51	1.2	1.7
Corpus size (sentences)	50,560,419	2,053,992	4,900,147
Total NER annotation time (seconds)	13,481	198	326
Average sentence length in chars	132	142	111
Throughput (sentences per second)	77	265	294
Throughput (kbytes per second)	9.94	36.76	31.88

In order to evaluate the performance of the NER module in annotating new corpora, we used once again the raw Wikipedia corpus in the same language. This time, the annotation was done using the trained NER models from the previous steps. Results in Table 3 show the size of the corpus we annotated and the total time we needed for the annotation. It is indicative that we need three and half hours to annotate the full English Wikipedia corpus for named entities using only one CPU. Using the four CPUs and more threads, it is possible to process the 51 Gbytes dataset in less than one hour. From the total NER annotation time and the corpus size in sentences we get an estimate of the module throughput performance, which varies significantly between languages. It is obvious that the size of the Wikipedia corpus varies significantly between languages. We also notice a difference in the average sentence size, which also affects the NER module throughput performance. The differences in performance are also due to the larger number of features used in OpenNLP when trained over a much larger corpus, as well as the larger lexicon that is used.

6 Challenges

6.1 Multilingual content

The biggest challenge for the solutions that we develop for detecting entities and sentiment is that they must be language agnostic.

As far as it concerns NER, the most promising language independent technique is the statistical learning technique of maximum entropy (Sang and De Meulder, 2003; Curran and Clark, 2003) and the hidden Markov models and their variations (Zhou and Su, 2002). In our approach, we employ the maximum entropy model, which is a supervised technique that requires a training dataset for each language annotated with entities of different type. It is important to annotate all entity occurrences within the training set and provide a balanced set with equal number of entities from each category. It is also important, to train different models for each different language style (for example a different model for tweets and a different model for news articles). The large number of different language styles (even newspapers do not all use the same writing style) and the continuous domain shift in the sphere of news (new topics and new entities appear every day) make it important for a NER system to be able to adapt to domain changes (Wu et al., 2009). In order to achieve this adaptability, we have implemented a hybrid annotation technique, which combines the trained NER model and domain lexicons that contain verified named entities. On top of this annotation, we created a bootstrapping mechanism, which suggests new named entities to be added to the lexicon, when they are frequently detected within the articles. The use of lexicon guarantees that all the occurrences of these words will be detected in the text, even when the NER model fails, thus increasing recall. And this process runs continuously for every new language corpus.

The need for extensive training also exists in the case of SA. The techniques that use word vectors (Wu et al., 2009), latent Dirichlet allocation (LDA) (Boyd-Graber and Resnik, 2010) or any supervised algorithm require large training corpora or parallel linguistic resources in all languages (thesauri, lexicons, etc.) in order to operate. In all cases, the human resources required for training the models and the processing power for applying them to new texts is huge. Once again, we developed tools that allow the quick and semi-automatic creation of training corpora. The annotation tools support an Auto-annotate feature, which employs existing models in order to annotate the texts and then allows human annotators to correct any possible errors, thus reducing the overall effort required. When applying the models, we take advantage of the multiprocessor infrastructure and balance the load for the different languages.

6.2 Entity type ambiguity

One of the biggest challenges in the recognition of named entities in texts is the resolution of ambiguity for certain entity names and types.

One type of ambiguity refers to the names of companies or products (mainly), which can be simple words that we use in the everyday life, e.g., Windows, Apple, Amazon, to name a few. A solution to this ambiguity can be to examine the context of the word in order to decide whether it corresponds to an entity or not. The algorithm that we use and the most popular probabilistic NER algorithms (i.e., CRFs, hidden Markov models) take into account the neighbourhood of a candidate entity word in order make a decision. However, a side problem that arises in this case is that a large set of annotated documents is needed in order to cover as many entity context cases as possible. Another solution is to create specific entity lexicons for each country

and language, but this is a tedious task which requires many resources. In our approach, we are able to suggest words to be added (semi-automatically) to such lexicons, when they are frequently detected by the NER algorithm.

Another type of ambiguity refers to terms that correspond to more than one entity types. For example, it is quite common that companies (or organisations) take the name of the place they originate (e.g., Liverpool soccer team, Aegean airlines, Oxford university) and we have to search for a word in the context that determines the entity's type. Once again probabilistic NER algorithms perform well when available training instances exist.

A third type of ambiguity, which is more evident in morphologically rich languages such as Arabic or Greek, is to detect all the different writings of the same entity. Although this does not affect the named entity detection algorithm, it is important for the analytics engine, where we need to group all the different mentions under the same entity. Once again, our system provides a semi-automatic solution that suggests pairs of words that potentially correspond to the same entity. These pairs may appear frequently within the same context (the same article, within a set of articles that are on the same topic), may have high similarity in character-level or both.

6.3 Coreference resolution

An even more difficult task than detecting the different writings of the same entity can become coreference resolution. Coreference resolution is the task of grouping all the mentions of entities in a document into equivalence classes so that all the mentions in a given class refer to the same discourse entity. This assumes that even relative pronouns in a sentence can refer to an entity mentioned in a previous sentence. This problem is not so critical, when we only need the mentions to a named entity in article level. However, when we need fine-grained analytics that locate references to entities in sentence level and we are also interested to the opinion or sentiment assigned to the entity, then it is more important to detect such implicit references. For example, when a sentence mentions the name of the product and the sentence that follows expresses a negative sentiment on *this product*, without mentioning its name again, the resolution of coreference is important.

Despite the advances done for the English languages, the resolution of coreference for many languages is still far from being solved and several linguistic analysis resources are necessary in order to tackle it. Since we focus on content in multiple languages we have not yet incorporated a coreference resolution module into our architecture.

6.4 Content size

Another important issue when dealing with news and social media content is the volume and volatility of data. For example, when a major event happens in a country, the volume of tweets and articles multiplies (Giannakoulopoulos and Varlamis, 2009; Anstead and O'Loughlin, 2011) and the load for the system too. Although short term analysis of content is useful for sentiment monitoring purposes, the content itself may be of little use after the event. So it is important in such cases to process the content on the fly and summarise the sentiment or entity references. However, the content itself may be useful for post analysis or even when the users want to drill down to each specific mention to an entity instead of looking at the broad picture (mentions and overall sentiment).

In order to tackle this, we have two processing pipelines: one that processes content stream on the fly and feeds our real-time analytics and one that aggregates the content and stores it for future analysis or review. The first pipeline has to be really fast and offer a high throughput, whereas the second has lowest priority and uses the available resources in order to run various analytics on the raw text data.

7 Conclusions

In this article we presented the overall architecture of our platform, which is designed for the collection and analysis of social media and the news. We focused on the modules that perform text mining and analytics in order to extract useful knowledge for entities and sentiment associated to them. An initial analysis of the performance of the NER and SA modules shows that the solution achieves a performance which is comparable to related systems. There is still enough room for improving the quality of results and reach the levels of state-of-the art NLP methods, which however are evaluated in smaller test corpora and mostly focus on quality of results and neglect processing speed.

Acknowledgements

Part of the research in this paper was conducted for the ICT-000651 proposal, in the context of the ICT4Growth action.

References

Abbasi, A., Chen, H. and Salem, A. (2008) 'Sentiment analysis in multiple languages: feature selection for opinion classification in web forums', *ACM Trans. Inf. Syst.*, June, Vol. 26, No. 3, pp.12:1–12:34 [online] http://doi.acm.org/10.1145/1361684.1361685 29.

Abdul-Mageed, M., Diab, M.T. and Korayem, M. (2011) 'Subjectivity and sentiment analysis of modern standard Arabic', in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, *Ser. HLT '11*, Association for Computational Linguistics, Stroudsburg, PA, USA, Vol. 2, pp.587–591 [online] http://dl.acm.org/citation.cfm?id=2002736.2002851.

Anstead, N. and O'Loughlin, B. (2011) 'The emerging viewertariat and BBC question time: Television debate and real-time commenting online', *The International Journal of Press/Politics*, Vol. 16, No. 4, p.1940161211415519.

Atteveldt, W.V., Ruigrok, N. and Schlobach, S. (2008) 'Good news or bad news? conducting sentiment analysis on Dutch text to distinguish between positive and negative relations', in *Inf. Technology and Politics*, Vol. 5, No. 1, pp.73–94.

Barbosa, L. and Feng, J. (2010) 'Robust sentiment detection on twitter from biased and noisy data', in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, Association for Computational Linguistics, pp.36–44.

Blair-Goldensohn, S., Neylon, T., Hannan, K., Reis, G.A., Mcdonald, R. and Reynar, J. (2008) 'Building a sentiment summarizer for local service reviews', in *NLP in the Information Explosion Era*.

Boiy, E. and Moens, M.F. (2009) 'A machine learning approach to sentiment analysis in multilingual web texts', *Information Retrieval*, Vol. 12, No. 5, pp.526–558.

Boyd-Graber, J. and Resnik, P. (2010) 'Holistic sentiment analysis across languages: Multilingual supervised latent Dirichlet allocation', in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ACL, pp.45–55.

Cambria, E. and White, B. (2014) 'Jumping NLP curves: a review of natural language processing research', *Computational Intelligence Magazine, IEEE*, Vol. 9, No. 2, pp.48–57.

Celikkaya, G., Torunoglu, D. and Eryigit, G. (2013) 'Named entity recognition on real data: a preliminary investigation for Turkish', in *Application of Information and Communication Technologies (AICT), 7th International Conference on*, IEEE, pp.1–5.

Curran, J.R. and Clark, S. (2003) 'Language independent NER using a maximum entropy tagger', in *Proceedings of the Seventh Conference on Natural Language Learning*, HLT-NAACL, Association for Computational Linguistics, Vol. 4, pp.164–167.

Engonopoulos, N., Lazaridou, A., Paliouras, G. and Chandrinos, K. (2011) 'ELS: a word-level method for entity-level sentiment analysis', in *WIMS*, ACM, p.12.

Fan, R-E., Chang, K-W., Hsieh, C-J., Wang, X-R. and Lin, C-J. (2008) 'Liblinear: a library for large linear classification', *Journal of Machine Learning Research*, Vol. 9, pp.1871–1874.

Fan, W. and Gordon, M.D. (2014) 'The power of social media analytics', *Communications of the ACM*, Vol. 57, No. 6, pp.74–81.

Finkel, J.R., Grenager, T. and Manning, C. (2005) 'Incorporating non-local information into information extraction systems by Gibbs sampling', in *Proceedings of the 43rd Annual Meeting of ACL*, pp.363–370.

Giannakoulopoulos, A.P. and Varlamis, I. (2009) 'Developing a civic journalism social medium on the web: Technological methods and constraints', in *Proceedings of the Colloque International Enjeuxet Usages des TIC (EUTIC)*, Bordeaux, France.

Godbole, N., Srinivasaiah, M. and Skiena, S. (2007) 'Large-scale sentiment analysis for news and blogs', in *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. (2009) 'The weka data mining software: an update', *SIGKDD Explor. Newsl.*, November, Vol. 11, No. 1, pp.10–18.

Hu, M. and Liu, B. (2004) 'Mining and summarizing customer reviews', in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Ser. KDD '04*, ACM, New York, NY, USA, pp.168–177 [online] http://doi.acm.org/10.1145/1014052.1014073.

Hu, X., Tang, J., Gao, H. and Liu, H. (2013) 'Unsupervised sentiment analysis with emotional signals', in *Proceedings of the 22nd International Conference on World Wide Web*, *Ser. WWW '13*.

Johannessen, J.B., Hagen, K., Haaland, Å., Jónsdottir, A.B., Nøklestad, A., Kokkinakis, D., Meurer, P., Bick, E. and Haltrup, D. (2005) 'Named entity recognition for the mainland Scandinavian languages', *Literary and Linguistic Computing*, Vol. 20, No. 1, pp.91–102.

Li, C., Weng, J., He, Q., Yao, Y., Datta, A., Sun, A. and Lee, B-S. (2012) 'Twiner: named entity recognition in targeted twitter stream', in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, *Ser. SIGIR '12*, ACM, New York, NY, USA, pp.721–730.

Liu, X. and Zhou, M. (2013) 'Two-stage NER for tweets with clustering', *Information Processing & Management*, Vol. 49, No. 1, pp.264–273.

Liu, X., Zhou, M., Wei, F., Fu, Z. and Zhou, X. (2012) 'Joint inference of named entity recognition and normalization for tweets', in *Proceedings of the 50th Annual Meeting of the ACL*, *Ser. ACL '12*, ACL, Stroudsburg, PA, USA, Vol. 1, pp.526–535.

Locke, B.W. (2009) *Named Entity Recognition: Adapting to Microblogging*, Computer Science Undergraduate Contributions, Paper 29.

Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y. and Potts, C. (2011) 'Learning word vectors for sentiment analysis', in *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies*, Association for Computational Linguistics, Vol. 1, pp.142–150.

Makrynioti, N. and Vassalos, V. (2015) 'Sentiment extraction from tweets: multilingual challenges', *DaWaK 2015*, pp.136–148.

Minkov, E., Wang, R.C. and Cohen, W.W. (2005) 'Extracting personal names from email: applying named entity recognition to informal text', in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, *Ser. HLT '05*, Association for Computational Linguistics, Stroudsburg, PA, USA, pp.443–450 [online] http://dx.doi.org/10.3115/1220575.1220631.

Mohammad, S., Kiritchenko, S. and Zhu, X. (2013) 'NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets', in *Second Joint Conference on Lexical and Computational Semantics (*SEM)*, *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval)*, June, Vol. 2, pp.321–327.

Nasukawa, T. and Yi, J. (2003) 'Sentiment analysis: capturing favorability using natural language processing', in *Proceedings of the 2nd International Conference on Knowledge Capture*, *Ser. K-CAP '03*, ACM, New York, NY, USA, pp.70–77 [online] http://doi.acm.org/10.1145/945645.945658.

Pak, A. and Paroubek, P. (2010) 'Twitter as a corpus for sentiment analysis and opinion mining', in *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, European Language Resources Association (ELRA), May.

Pang, B., Lee, L. and Vaithyanathan, S. (2002) 'Thumbs up?: sentiment classification using machine learning techniques', in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, ACL, Vol. 10, pp.79–86.

Popescu, A-M. and Etzioni, O. (2005) 'Extracting product features and opinions from reviews', in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, *Ser. HLT '05*, Association for Computational Linguistics, Stroudsburg, PA, USA, pp.339–346 [online] http://dx.doi.org/10.3115/1220575.1220618.

Richman, A.E. and Schone, P. (2008) 'Mining wiki resources for multilingual named entity recognition', in *ACL*, pp.1–9.

Ritter, A., Clark, S., Mausam, and Etzioni, O. (2011) 'Named entity recognition in tweets: An experimental study', in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, *Ser. EMNLP '11*, ACL, Stroudsburg, PA, USA, pp.1524–1534.

Sang, E.F.T.K. and De Meulder, F. (2003) 'Introduction to the Conll-2003 shared task: Language-independent named entity recognition', in *Proceedings of the Seventh Conference on Natural Language Learning*, HLT-NAACL, Vol. 4, Association for Computational Linguistics, pp.142–147.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A.Y. and Potts, C. (2013) 'Recursive deep models for semantic compositionality over a sentiment treebank', in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Stroudsburg, PA, October, pp.1631–1642.

Szarvas, G., Farkas, R. and Kocsor, A. (2006) 'A multilingual named entity recognition system using boosting and C4.5 decision tree learning algorithms', *The Ninth International Conference on Discovery Science 2006, DS 2006*, Barcelona, Spain, pp.267–278.

Turney, P.D. (2002) 'Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews', in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, *Ser. ACL '02*, pp.417–424.

Varlamis, I., Tsirakis, N., Poulopoulos, V. and Tsantilas, P. (2014) 'An automatic wrapper generation process for large scale crawling of news websites', in *Proceedings of the 18th Panhellenic Conference on Informatics*, ACM, pp.1–6.

Wiebe, J. and Riloff, E. (2005) 'Creating subjective and objective sentence classifiers from unannotated texts', in *CICLing*, pp.486–497.

Wilson, T., Wiebe, J. and Hoffmann, P. (2005) 'Recognizing contextual polarity in phraselevel sentiment analysis', in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, *Ser. HLT '05*, pp.347–354.

Wu, D., Lee, W.S., Ye, N. and Chieu, H.L. (2009) 'Domain adaptive bootstrapping for named entity recognition', in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Vol. 3, pp.1523–1532.

Zhao, J., Dong, L., Wu, J. and Xu, K. (2012) 'Moodlens: an emoticon-based sentiment analysis system for Chinese tweets', in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Ser. KDD '12*, pp.1528–1531.

Zhou, G. and Su, J. (2002) 'Named entity recognition using an hmm-based chunk tagger', in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, pp.473–480.