

Using language-independent rules to achieve high multilinguality in Text Mining

Ralf STEINBERGER¹, Bruno POULIQUEN and Camelia IGNAT
European Commission – Joint Research Centre
21020 Ispra (VA), Italy

Abstract. A lot of information is hidden in foreign language text, making multilingual information extraction tools – and applications that allow cross-lingual information access – particularly useful. Only a few system developers offer their products for more than two or three languages. Typically, they develop the tools for one language and then adapt them to the others. Information on guidelines how to produce highly multilingual applications with the least possible effort is scarce. Due to the multinational character of the European institutions, the Text Mining applications developed at the *Joint Research Centre* (JRC) had to be planned with the multilinguality requirements in mind. The purpose of this chapter is to present the design principles behind the in-house developments and to show a number of different applications that are built according to these guidelines. The results of the text analysis applications presented here are visible on the publicly accessible multilingual news gathering, analysis and exploration tool NewsExplorer.²

Keywords. Design principles; multilinguality; language-independence; geo-tagging; named entity recognition; morphological name variation; name variant mapping; quotation recognition; date recognition; cross-lingual document similarity calculation.

1. Introduction

Text Mining generates valuable meta-information for texts that can be used for a variety of applications. These include the highlighting of relevant information in documents and storing the extracted meta-information to provide users with powerful search functionality that goes much beyond that of full-text search. As a lot of information is available in some languages and not in others, there is a clear benefit to multilingual information extraction. However, the development of text mining tools for new languages is typically rather time-consuming, which explains why most systems only cover one or a small number of languages. The purpose of this chapter is to present some simple, successfully applied design principles that make it easier to extend text analysis tools to many languages. They can be roughly summarised as follows: By combining universal, i.e. language-independent rules with relatively simple language-specific resource files, it is possible to extend the language coverage of text analysis applications quickly. Secondly: when *cross-lingual* applications need to be developed for a large number of language pairs, it is important to represent monolingual contents in a language-neutral way so as to avoid the need for language pair-specific resources.

¹ The email address of the corresponding author is Ralf.Steinberger@jrc.it.

² NewsExplorer currently covers 19 languages. It is publicly accessible at <http://press.jrc.it/NewsExplorer>.

Before describing these design principles in a bit more detail (Section 1.3), we will first remind the reader of the various benefits of using text mining tools (1.1) and give some concrete examples of cases where multilingual information extraction and aggregation tools provide more information than monolingual ones (1.2). Section 2 on related work will give an overview of existing multilingual and cross-lingual applications and of the few design principles mentioned by other system developers regarding multilinguality. This is followed by short descriptions of some Text Mining applications developed according to the principles presented here: the recognition and disambiguation of references to geographical places (3) and of persons and organisations (4); a light-weight approach of dealing with inflection and other regular word variations (5); a method to identify name variants across many languages and writing scripts (6), a tool to recognise quotations (7) and one to recognise and disambiguate dates (8). Section 9 then shows how various monolingual text analysis tools can be combined to link related documents across languages without the need of language pair-specific resources. Avoiding language pair-specific components is the essential requirement when the objective is to link related documents in many different language pair combinations. Due to the wide variety of linguistic phenomena, identifying language-independent rules and applying them to many languages is not trivial. Section 10 lists a few linguistic phenomena that made it necessary to adapt the generic rules, or to allow exceptions. A short summary concludes the chapter (11).

1.1. Benefits of using text mining tools

The usage of software for named entity recognition and classification (NERC), which can identify references to persons, organisation, locations and other entities, has a number of benefits for the information-seeking user, including the following:

- Detection of relevant information in large document collections and – on demand – highlighting of the found words or phrases in the text;
- Display of all the extracted information for a larger, monolingual or multilingual, document collection, serving as a multi-document summary and allowing users to narrow down the search by selecting some extracted information features;
- Advanced search functionality operating on meta-information, allowing, for instance, to search for person names mentioned in documents that mention a given country and a given person in a certain time period;
- Automatic inference: a text mentioning, e.g., the city of *Izmir* can be automatically marked as being about Turkey, even if the country is not mentioned explicitly;
- Visual summary: for document collections, in which locations have been identified and disambiguated, maps can give an overview of the geographical coverage;
- Geographical search: find all documents mentioning a place in a certain country, or within a certain radius of another location;
- Name variants: If spelling variants for the same person have been automatically detected, users can search for mentions of a person independently of the spelling;
- Cross-lingual information access: for instance, display in one language information extracted from texts in other languages;
- Linking of similar documents across languages: e.g. by virtue of the fact that they talk about the same entities;
- Visualisation of information: e.g. to show social network graphs with relations between people, or allow to navigate document collections via a document map.

This non-exhaustive list gives an idea of the power of text analysis applications and of the benefits users can reap by using them.

1.2. Practical arguments for multilinguality

The functionalities listed here are mostly available for the nineteen languages currently covered by the NewsExplorer application, which has been developed according to the basic principles described in this chapter. While many of these functionalities are applicable to information extracted from texts in a single language, the usefulness rises significantly if the information is derived from documents in many different languages. In NewsExplorer, which analyses an average of 35,000 articles gathered from about 1,300 news portals world-wide in 19 languages, about a quarter of the news articles are written in English. However, there is ample evidence that much of the information would not be found if only English texts were analysed. Some examples are:

- Some areas of the world are better covered by different languages. For instance, North Africa is covered better by French, Brazil by Portuguese, and the rest of Latin America by Spanish. By analysing texts in more languages, more information will be found.
- Daily and long-term social network analysis has shown ([1], [2]) that the dominant and most frequently mentioned personalities in English language news are the US President and the British Prime Minister. When looking at French, German, Arabic or Russian news, the respective leaders have a much more central role. Adding analysis results from more languages reduces bias and increases transparency.
- NewsExplorer automatically collects name attributes (titles, professions, age, role, nationality, family relations, etc.) about the approximately 700,000 persons found in media reports in the course of several years from the news in different languages. The various attribute lists show clearly that the information found across languages is often complementary. To give a concrete example: For the murdered Russian spy Alexander Litvinenko, texts in most languages contained the information that Litvinenko was a Russian and that he was an agent. However, we found in French news that he was 43 years of age, in Italian news that he was killed, in German news that he was a *critic* of some sort, etc.³

In a national context with a dominant language, for instance English in the USA, a possible solution to this multilinguality requirement is to use machine translation (MT) to translate the documents written in the most important foreign languages into the national language, and to then apply the text analysis tools to the texts in that language. In the European context, where there is not one, but 23 official languages, using MT is not an option, as there are 253 language pairs ($N * (N - 1) / 2$, with N being the number of languages involved). Multilingual text processing is generally important, but it is considerably more relevant in the European context than in most other settings. Another important issue is the fact that MT often fails when being confronted with proper names or specialist terminology ([3]). Larkey et al.'s *native language hypothesis* ([4]) – the observation that text analysis results are better if performed on the source language text – is also a good argument for applying multilingual text analysis rather than operating on machine-translated texts.

³ See <http://press.jrc.it/NewsExplorer/entities/en/97338.html> for more details regarding this example.

1.3. Design principles making it easier to achieve high multilinguality

There is more than one way to achieve multilinguality and Section 2 shows that there are other systems that cover several languages, but only few developers have explicitly mentioned the principles behind their development. The guidelines underlying the development of the NewsExplorer application are the following:

- Use language-independent rules wherever possible, i.e. use rules that can be applied to any new language that will be added to the system.
- Keep language-specific resources and tools to a minimum: Such tools are, for instance, linguistic software like part-of-speech taggers and parsers. Language-specific resources are morphological or subject domain-specific dictionaries, linguistic grammar rules, etc. As acquiring or developing such language-specific resources is difficult and expensive, it is better not to use them, or – where they are necessary – to use them as little as possible.⁴
- Keep the application modular by storing necessary language-specific resources outside the rules, in language-specific parameter files. That way, new languages can be *plugged in* more easily.
- For the language-specific information that cannot be done without, use bottom-up, data-driven bootstrapping methods to create the monolingual resources.
- Avoid language pair-specific resources and procedures because the almost exponential growth of language pairs would automatically limit the number of languages a system can deal with.

These are very simple and generic guidelines, but sticking to them is not always easy and sometimes comes at a cost. Sections 3 to 9 will describe several applications of different types that each try to follow these guidelines.

2. Related work

The intention of this chapter is to specifically address the issues surrounding the development of multilingual and cross-lingual applications involving larger numbers of languages. When discussing related work, we will distinguish multi-monolingual tools, i.e. monolingual applications applied to more than one language (Section 2.1), from cross-lingual tools, i.e. those that involve crossing the language barrier in one way or another (Section 2.2). As the applications described in Sections 3 to 9 are only there to exemplify the guidelines presented in Section 1.3, we will not give a full state-of-the-art overview for each of these tasks.

2.1. Multi-monolingual systems

Applications that mainly use statistical methods, such as search engines, document categorisation tools, Optical Character Recognition (OCR) software or spell checkers, are usually available for larger numbers of languages. The news aggregators *Google News*, *Yahoo News* and *Europe Media Monitor EMM*, for instance, cover 17, 34 and 43

⁴ Part-of-speech tagging software is freely available and can, in principle, be trained for any language, but we feel that the effort involved for training 20 languages is too big for the expected benefit. Using readily trained taggers is not an option for us, as the overheads of dealing with different tag sets and levels of performance, and with differing input and output formats, etc. are too big.

languages, respectively,⁵ as the crawling of news sites is language-independent and the clustering of related news items can be done with mostly statistical methods. ZyLab's ZyScan® software for OCR claims to cover over 200 languages.⁶ For more linguistic applications such as information extraction tools or grammar checkers, the number of available languages is usually much more restricted. Very few companies have produced and sell text analysis tools for a wider range of languages. InXight's information extraction tool set ThingFinder® is an exception, offering the impressive number of thirty languages.⁷

The usage of Machine Learning methods is rather popular as they are promising methods to overcome the bottleneck faced by introspection-based symbolic (rule-based) methods where linguists need to describe the vocabulary and the grammar separately for each language. In Machine Learning approaches to named entity recognition, for example, the idea is to use texts in which named entities (NEs) have been marked-up (usually manually) and to let algorithms learn the features that help to identify new NEs in new texts. Features are, for example, surrounding words, their distance to the NE, their part-of-speech, case information (uppercase vs. lowercase), and more. In order to train tools for new languages, it is thus necessary to create training sets with examples. For improved accuracy, it is furthermore advised to manually confirm or reject the learned rules. Even for Machine Learning methods, the effort required per language is not negligible. Similarly, the development of statistical part-of-speech taggers relies on rather large manually tagged document sets.

For all approaches, the effort of developing monolingual applications for N languages can be up to N times the effort of developing the application for one language, although the complexity of languages differs, of course. Furthermore, there is usually a learning curve involved and the development environment (e.g. the formalism that allows to write the rules) has to be developed only once, so that the effort per language gets less, the more languages are being covered ([5]). The limited number of languages offered by commercial and academic text analysis tools is due to the large language-specific effort required, but by throwing more manpower at the application, more languages can, in principle, be covered.

Multiple authors have described work on developing resources and tools for a number of different languages, but this was typically done by reusing the resources from a first language and adapting them to new languages (e.g. [5], [6], [7], [8]). Tips from various system developers for achieving multilinguality include modularity ([7], [8], [9]), simplicity of rules and the lexicon ([8]), uniform input and output structures ([8], [10]), and the use of shared token classes that are ideally based on surface-oriented features such as case, hyphenation, and includes-number ([8]). SProUT grammar developers took the interesting approach of splitting the multilingual grammar rule files ([8]): some files contain rules that are applicable to several languages (e.g. to recognise dates of the format *20.10.2008*) while others contain language-specific rules (e.g. to cover *20th of October 2008*). The fact that this latter date format and others can also be captured by using language-independent rules with language-specific parameter files will be shown in Section 8. Many people have used the GATE architecture to write grammars and tools for a large variety of languages and there surely are good-practice rules to save effort by reusing existing resources, but we have not been able to find an

⁵ See <http://news.google.com>, <http://news.yahoo.com> and <http://press.jrc.it>. (last visited 1.02.2008)

⁶ http://www.zylab.com/products_technology/productsheets/ZySCAN.pdf

⁷ See <http://inxight.com/products/sdks/xf/>. (last visited 1.02.2008)

explicit mention of them apart from architectural design issues, the promotion of Unicode and the usage of virtual keyboards to enter foreign language script ([9]).

2.2. Cross-lingual applications

For *cross-lingual* applications, the situation is different. Cross-lingual applications are those that help cross the language barrier and thus involve language *pairs*. Examples are machine translation (MT), cross-lingual question answering, cross-lingual document similarity calculation (CLDS), cross-lingual glossing, cross-lingual topic detection and tracking, name variant mapping across languages and scripts, etc. With state-of-the-art methods, the effort to produce any such software is more or less linear to the number of language *pairs*. As the number of language pairs for N languages is $N * (N - 1) / 2$ (e.g. for 10, 20 or 30 languages, there are 45, 190 and 435 language pairs, respectively), the number of available language pairs for cross-lingual applications is always rather restricted. According to the web sites of some of the biggest MT providers, Language Weaver⁸, Systran⁹ and Google¹⁰, the companies offer 28 language pairs involving 21 languages, 18 language pairs involving 13 languages and 29 language pairs involving 14 languages, respectively. All companies offer mainly language pairs involving English.¹¹ Admittedly, MT is particularly resource-intensive, but similar language pair restrictions apply to other application types. There are various news aggregation systems, but besides our own, NewsTin¹² is the only one that offers any kind of cross-lingual functionality: for 10 languages, NewsTin allows users to search for multilingual articles about a chosen entity (e.g. a person name) or for a given subject category. In comparison, the NewsExplorer application presented in this chapter is able to link news clusters and extracted information in all language pair combinations for 19 languages¹³ and is not restricted by existing categories.

Current approaches to cross-lingual name variant matching are limited to individual language pairs (English/Arabic [11], English/Chinese [12], English/Korean [13] or English to Japanese [14], [15]). The same applies to cross-lingual topic detection and tracking systems, which require CLDS calculation. These systems typically either use MT (e.g. [16]) or bilingual dictionaries (e.g. [17] and [18]) to link related texts across languages, and this again limits the number of language pairs worked on. Statistical methods that establish cross-lingual word associations by feeding the vector space of Machine Learning systems with small pieces of text and their translation (parallel texts), such as bilingual Latent Semantic Indexing (LSI, [19]) and Kernel Canonical Correlation Analysis (KCCA, [20]), have to date only been successful with individual language pairs (involving thus a maximum of two languages). Whatever the application,

⁸ See <http://www.languageweaver.com/page.asp?intNodeID=930&intPageID=960>

⁹ See <http://www.systran.co.uk/translation/translation-products/desktop/systran-premium-translator>

¹⁰ See http://www.google.co.uk/language_tools

¹¹ In the field of MT, effort can be saved by substituting the *direct approach* (i.e. translating words or phrases directly from one language to the other) by a *transfer-based approach*. In the latter, each source language is transformed into an abstract representation, which does not depend on the target language. The only language pair-specific component thus is the transfer module. The *interlingual approach* to MT goes even further as the aim is to transform the source language text into a language-independent representation and to generate the target language equivalence out of this interlingual representation. The trend of the last few years has been to use Machine Learning to produce direct approach systems.

¹² See <http://www.newstin.com/> (last visited 1.2.2008).

¹³ For purely computational reasons, NewsExplorer currently offers 'only' the 93 language pairs involving the six pivot languages Dutch, English, French, German, Italian and Spanish, out of the 171 possible combinations for 19 languages. The remaining language pairs could be covered by adding an extra machine.

existing systems – be they commercial or academic research systems – are rather strictly limited in the number of languages covered.

There is currently an effort to tackle more languages and more language pairs, for instance in the EU-funded projects *EuroMatrix*¹⁴ for MT and SMART¹⁵ (*Statistical Multilingual Analysis for Retrieval and Translation*) for MT and CLDS. The European CLEF¹⁶ project (*Cross-Lingual Evaluation Forum*) aims at pushing multilinguality for information retrieval and other applications. The CONLL'2003 shared task aimed at achieving language-independent named entity recognition.¹⁷ The projects Multext¹⁸, Multext-East¹⁹ and Global-WordNet²⁰ produced – or are producing – multilingual linguistic resources. Furthermore, the highly multilingual parallel text corpora JRC-Acquis ([21]) and the DGT-Translation Memory²¹ have recently become available, showing the increased interest in accessing information hidden in foreign language text. All these resources will certainly help push multilinguality, but additionally to the resources, approaches need to be found that allow producing tools in more languages quickly and to cover more language pairs. The approach presented in this chapter aims at exactly that: offer a possible solution to a large-scale problem. The following seven sections will describe how the approach presented in Section 1.3 has been applied to a number of different text analysis applications. The first six of the described solutions have been implemented as part of the NewsExplorer system and the results can be seen on its public news portal.

3. Recognition and disambiguation of references to geographical locations

The text mining task of *geo-tagging* (also referred to as *geo-coding* or as *grounding of geographical references*) consists of recognising references to geographical locations in free text and to identify unambiguously their (ranges of) latitude and longitude. Geo-tagging goes beyond the more commonly pursued task of *geo-parsing*, which consists of recognising the words or phrases without attempting to put a dot on a map. The latter can theoretically be done simply by looking at linguistic patterns, even if there is evidence that geo-parsing is difficult without a gazetteer ([22]). For instance, when we find the piece of text “... is located near XYZ”, we can infer that XYZ is a place name of some sort. Geo-tagging, on the other hand, is not possible by looking at the text only. Additionally, a gazetteer, i.e. a list of existing places and their latitude-longitude and probably more information, is needed. A number of gazetteers are freely or commercially available, such as *GeoNames*²² and the *Global Discovery*²³ database. These gazetteers usually contain place names in one or more languages, latitude and longitude information, size categories for each place (distinguishing capitals from major or minor cities, villages, etc.), as well as hierarchical information indicating that a town belongs to a county, which is part of a region, which is itself part of a country, etc. A third gaz-

¹⁴ See <http://www.euromatrix.net/> (last visited on 1.2.2008)

¹⁵ See <http://www.smart-project.eu/> (last visited on 1.2.2008)

¹⁶ See <http://clef.isti.cnr.it/> (last visited on 1.2.2008)

¹⁷ See <http://www.cnts.ua.ac.be/conll2003/ner/> (last visited on 5.2.2008).

¹⁸ See <http://aune.lpl.univ-aix.fr/projects/multext/> (last visited on 1.2.2008)

¹⁹ See <http://nl.ijs.si/ME/> (last visited on 1.2.2008)

²⁰ See <http://www.globalwordnet.org/> (last visited on 1.2.2008)

²¹ See <http://langtech.jrc.it/DGT-TM.html> (last visited on 1.2.2008)

²² See <http://www.geonames.org/> (last visited 30.01.2008)

²³ See <http://www.europa-tech.com/> (last visited 1.2.2008)

etteer, the KNAB²⁴ database, is particularly useful as it contains a large number of exonyms (foreign language equivalents like *Venice*, *Venise* and *Venedig* for the Italian city of *Venezia*), as well as historical variants (e.g. *Constantinople* for the Turkish city of *Istanbul*).

Geo-tagging thus consists of finding gazetteer entries (and probably other expressions) in text. This is basically a lookup task, but there are a number of challenges that make this task much harder than it seems at first sight. We will present these briefly in the next section and will then present a language-independent rule set to deal with these issues. For a definition of the tasks and an overview of existing commercial and academic approaches, see the recent Ph.D. thesis by Leidner ([23]). For a more detailed description of the challenges, the proposed knowledge-poor solution and issues concerning the compilation of a multilingual gazetteer, see [24].

3.1. Challenges for Geo-tagging

When using a gazetteer, the first step in recognising place names in text is a look-up task. In all European languages, it is possible to compare only uppercase words in the text with the entries of the gazetteer. In languages not distinguishing case, such as Arabic or Chinese, every single word must be compared to the gazetteer. However, even for European languages, geo-tagging is more than a simple lookup task, as a number of ambiguities need to be solved and language-specific issues need to be tackled. The challenges are the following:

- (a) Homography between places and persons: For instance, there are both places and persons with the names of *George* (South Africa plus several other places worldwide with this name), *Washington* (capital of the USA and almost 50 other locations), *Paris* (French capital and almost 50 other locations) and *Hilton* (United Kingdom and many more places with this name).
- (b) Homography between different places: An example is *Alexandria*, as there are 24 different cities with this name in ten different countries: Greece, Romania, South Africa, USA, etc.
- (c) Homography between places and common words: For example, the English adjective *Nice* is also a city in France, the English verb *Split* is homographic with a major city in Croatia, etc. We have found a large number of places homographic with common words for all languages we worked with.
- (d) The same place has different names: This is not only true across languages (e.g. the Italian city of *Milano* has the ‘translations’ – or exonyms – *Milan*, *Milán*, *Mailand*, *Μιλάνο*, *Милан*, *Милано*, *ميلانو* etc.). Even within the same country – and sometimes within the same language – places can have several names. Examples are *Bombay/Mumbai* and *Bruxelles/Brussell*.
- (e) Place names are declined or morphologically altered by other types of inflection. This has the consequence that the canonical form found in the gazetteer will frequently not coincide with the declension found in the text. The US-American city of *New York* can for instance be referred to as *New Yorkului* in Romanian or as *New Yorgile* in Estonian.

²⁴ See <http://www.eki.ee/knab/knab.htm> (last visited 1.2.2008)

3.2. Language-independent rules for Geo-tagging

Challenges (d) and (e) necessarily involve the usage of language-specific resources: If name variants like *Mailand*, *Milàn* or *Милан* are not in the gazetteer (challenge (d)), they cannot be looked up. The only solution to this problem is to find ways of populating an existing gazetteer with multilingual place name variants in the least time-consuming way. At the JRC, we have merged various gazetteers and additionally exploit the online encyclopaedia Wikipedia for this purpose. Wikipedia can even be used to find inflected forms of city names (challenge (e)). For instance, for the same city, the following inflection forms were found on the Finnish Wikipedia page: *Milanon*, *Milanossa*, *Milanosta*, *Milanolainen*, *Milanoon*, *Milanolaiset*, *Milanoa*. The temptation is big to simply use generous wild cards such as *Milan**, but due to the hundreds of thousands, or even millions of entries in a gazetteer and the likely similarity with other words of the language, this will lead to many wrongly recognised names, lowering Precision. This specific wild card pattern would, for instance, wrongly recognise the Italian city of *Milano Marittima* and the Polish city *Milanówek*. The issue of inflection and the challenge to recognise other name variants will be discussed in Section 5, together with the variations of person names.

While challenges (d) and (e) cannot be overcome without language-specific resources and procedures, there are generic solutions for challenges (a), (b) and (c) that require no – or extremely little – language-specific effort. The proposed heuristics are the following:

- (a) For known names, prefer person name readings over location name readings: The idea is to ignore all potential locations that are homographic with a name part of a person talked about in the same document. For instance, if *Javier Solana* has been mentioned in the text, we should assume that any further reference to either *Javier* or *Solana* refers to the person and not to the respective locations in Spain and the Philippines. The reason for this rule is that person name recognition is more reliable than geo-tagging.
- (b) Make use of information about a place's importance or size: Many gazetteers like GeoNames or Global Discovery use size classes to indicate whether a place is a capital (size class 1), a major city, a town, etc. or a village (size class 6). If no further information is available from the context, we can assume that the text refers to the larger location. For instance, the Romanian city of *Roma* is of size class 4, while the Italian city with the same name is a capital (size class 1). The Italian capital should thus be chosen over the Romanian town.
- (c) Make use of the country context: the idea is that – if we already know that a text talks about a certain country – then it is likely that a homographic place name should be resolved in favour of the place in that country. For instance, if we know that the text talks about Romania because either the news source is from Romania or because another *non-ambiguous* reference is made to the country or any of its cities, the likelihood that we talk about the Romanian town *Roma* is much bigger.
- (d) Prefer locations that are physically closer to other, non-ambiguous locations mentioned in the text: In the case of ambiguity between two homographic places of the same size class, it is likely that the author meant to refer to the one nearby. For instance, there are two cities called *Brest*, one in France and one in Belarus. If both *Brest* and *Warsaw* are mentioned in a text, the latter reading will be preferred because it is at a distance of 200 km from Warsaw, while the French port is 2,000 km away.

(e) Ignore places that are too difficult to disambiguate: Locations that are homographic with common words of a language frequently lead to wrong hits. Such locations should be put on a language-specific geo-stop word list and ignored if found in a text. If an author really intends to refer to the places called *And* (Iran) or *Had* (India, and others), these references will be missed, but many errors will be avoided. Such geo-stop word lists are language-dependent because the same words are likely not to be ambiguous in another language. For instance, *And* and *Had* are not ambiguous in German as there are no such words. Geo-stop word lists can be produced semi-automatically by comparing the gazetteer entries with a list of the most frequent words of a language and by hand-picking those words that were found. In order to reduce the work load, this list can be narrowed down to those words that appear more often in lowercase than in uppercase. The uppercase word *This* (potentially referring to a location in France), for instance, will be found in such a word frequency list, but the lowercase variant *this* will be much more frequent, meaning that *This* is a good candidate for the geo-stop word list. It is also advisable to compare the lexicon of a language with an international list of frequent first names.

These heuristics were derived from multilingual test data and were found to produce good results in a majority of cases ([24]). The first four are completely independent of the text language as they refer to external parameters such as geographical location and location size. The fifth heuristic is language-dependent, but a good geo-stop word list for any given language can be produced within a few hours.

3.3. Combination of the rules

The rules mentioned in the previous section may contradict each other, so they have to be combined into a single rule that regulates their relative preference. When geo-tagging a new text, the binary rules will first be applied, i.e. geo-stop words will be ignored and potential locations that are homographic with part of a person name found in the text will not be considered. In a second instance, Formula (1) will be applied. Formula (2) explains the calculation of the parameter *kilometric weight*.

$$\begin{aligned} &\text{For computing the score, the current settings are:} && (1) \\ &Score = classScore [80,30,20,10,5] \\ &+ 100 \text{ (if country in context)} \\ &+ 20 \cdot kilometricWeight() \end{aligned}$$

where: *classScore* is a given score depending on the importance of the place (this is coded by the *class* attribute: 80 for country name, capital or big city, 30 for province level city, 20 for small cities, 10 for villages, 5 for settlements); *kilometricWeight*, which has a value between 0 and 1, is the minimum distance between the place and all non-ambiguous places. This distance d is weighted using the arc-cotangent formula, as defined by Bronstein ([25]), with an inflexion point set to 300 kilometres²⁵, as shown in Equation 2.

²⁵ Empirical experiments showed that distances of less than 200 km are very significant, and distances more than 500 km do not make a big difference. Therefore, we have chosen the inflexion point between those two values: 300.

$$\text{kilometric Weight}(d) = \frac{1}{\text{arcCot}\left(-\frac{300}{100}\right)} \text{arcCot}\left(\frac{d-300}{100}\right) \quad (2)$$

The formulae do not make reference to any language-specific features so that they can be applied to any new language without further consideration.

4. Recognition of person and organisation names

The names of *known* persons and organisations can be identified in new documents through a lookup procedure, just like the geographical place names from a gazetteer. For morphological and other variants, the same measures can be taken as for geographical names. These measures will be discussed in Section 5. However, as exhaustive lists of person and organisation names do not exist, *new names* need to be identified in one of two different ways: (1) When using dictionaries of a language, one could assume that any unknown word found in text is a proper name, but using dictionaries alone would be dangerous because any typo would then be identified as a name, as well. For languages that distinguish case and that begin proper names with an uppercase letter, the number of name candidates can of course be limited. (2) Additionally, or instead of using dictionaries, local patterns can be used, which can be either hand-written or acquired automatically with Machine Learning methods. Such local patterns are words, multi-word expressions or regular expressions that occur near to the names and that indicate that some (uppercase) words are likely to be a name. Such patterns can be titles (e.g. *Minister*), words indicating nationality (e.g. *German*), age (e.g. *32-year old*), occupation (e.g. *playboy*), a significant verbal phrase (e.g. *has declared*), and more. The words and expressions of different types can be referred to generically as *trigger words*, as their presence triggers the system to identify names. It goes without saying that these pattern recognition resources are necessarily language-specific. The challenge is thus to (a) keep the effort to produce these patterns to a minimum and (b) to formulate them in a generic way, which makes it easy to produce the patterns for a new language.

4.1. Patterns for name guessing

In JRC's named entity recognition tools, the generic patterns are basically language-independent, but they make use of language-specific resource files that contain the language-specific information such as lists of titles, nationality adjectives, etc. For full details on the used methods, see [3] and [26]. Here, we will focus on the aspect of language independence and on how to create language-specific resources quickly and with little effort. One intrinsic feature of JRC's tools is that they will only recognise names with at least two name parts. The reason for this is that the aim in NewsExplorer and other JRC tools is not only to recognise that *Angela* or *Bush* are names, but to identify the exact referent and its name variants so that users can search for all news items mentioning this person.

The following are the basic rules to identify person names in languages that write names with uppercase. For other languages, see Section 10:

- (a) Any group of at least two uppercase words that is found to the left or to the right of one or more trigger words will be identified as a name candidate. Trigger words can

also be regular expressions such as $[0-9]+-?year-old$ to capture *43-year-old Alexander Litvinenko*.

- (b) The pattern allows the presence of a number of name infixes which can also be written in lower case, such as *von, van der, bin, al, de la*, etc. to also capture names such as *Mark van der Horst, Oscar de la Renta, Khaled bin Ahmed al-Khalifa, etc.*
- (c) Furthermore, a number of other frequent words are allowed between trigger words and the name. These can be determiners (e.g. *the, a*), adjectives and other modifiers (*former, wandering*), or compound expressions (*most gifted*), allowing name recognition in sentences like “... *Juliette Binoche, the most gifted French actress*”.
- (d) Patterns should also allow for the inclusion of a slot for other names (e.g. *United Nations*), in order to capture expressions such as *Envoy to the United Nations*.
- (e) Names are also recognised if one of the potential name parts is a known first name. Large lists of first names from different languages and countries are thus a useful resource, that can be used for the name recognition in all languages. In the example *Angela UnknownWord*, the second element would thus be identified as the second name part. First names are thus different from the other trigger words mentioned earlier because they are part of the name, while titles and country adjectives are not.
- (f) Organisation names are different in that they are often longer and they can contain several lowercase words that are normal words of the language, as in *Federal Ministry of the Interior*, etc. In order to capture such names, the patterns must allow various sequences of typical organisation trigger words (e.g. *Bank, Organi[sz]ation, Ministry, Committee, etc.*), lowercase filler words (e.g. *of the*) and other content words (e.g. *Interior, Finance, Olympic, etc.*).

It is useful to also allow long sequences of trigger words to capture expressions like *former Lebanese Minister of Agriculture*. While the combination *Minister of Agriculture* may be enough to recognise the proper name, storing trigger words and their combinations has the advantage that they provide useful information on a person. In NewsExplorer, the historically collected trigger words are displayed together with each person.

4.2. Bootstrapping the acquisition of language-specific pattern ingredients

Besides the list of common first names, which can be used for the recognition of new names in any language, the various trigger words mentioned in the previous section are clearly different from one language to the other. These lists can be rather long. Our English list, for instance, consists of about 3400 trigger words. In order to compile such a list for a new language (e.g. Romanian), it is convenient to search a large corpus of that language for known names and to produce a frequency list of left and right-hand-side contexts of various sizes (e.g. between one and five words). The most frequent trigger words can then be manually selected. Bootstrapping will make the process more efficient: instead of going manually through the typically very long name context lists, it is better to use the most frequent trigger words found and to search the collection again for new names in order to collect more name contexts, and so on. Wikipedia or similar sources often contain lists of common first names and titles, so that such lists can also be used as a starting point.²⁶ The effort to produce working lists of recognition

²⁶ See, for instance, the web site www.behindthename.com for first names, and the following page for lists of professions: <http://en.wikipedia.org/wiki/Category:Occupations> (available in various languages).

patterns for a new language is between half a day and 5 person days. Note that the manual selection of trigger expressions is indispensable and that an intelligent reformulation of expressions can improve the rules massively. For instance, for Romanian occupations like *ministrul de interne*, experts can expand the rule immediately to other occupations: *ministrul de externe*, *ministrul de finanțe*, *ministrul justiției*, *ministrul transporturilor* and more (Minister for external affairs, finance, justice and transport, respectively). They can even write a more complex pattern to allow the recognition of combinations like *ministrul transporturilor, construcțiilor și turismului* (Minister of transport, construction and tourism) or *Ministrul delegat pentru comerț* (Vice-minister for commerce). In the case of Romanian, the first list has been validated to a certain extent and the expert produced 231 trigger words in about 3 hours of time. After this new list has been compiled, we launched the candidate extractor again and another validation was done by the expert. We now have 467 trigger words and Romanian is used fully in NewsExplorer, where it recognises an average of one hundred new names every day.

Another bootstrapping method would be to use MT or bilingual dictionaries in a triangulation approach, i.e. translating from two or more different languages into the new target language and to use only the overlapping results.

5. A light-weight way of dealing with inflection and other regular variations

We have seen that – for the tasks of geo-tagging, recognition of known person names and even for the guessing of new names in text – it is necessary to compare word forms in the text with lists of known words in lookup lists. Even though the task looks simple, looking up known entities in a new language is not always so straightforward because the word forms found in the text often differ from the word forms in the lookup tables.

5.1. Reasons for the existence of name variants

The main reasons for these differences between the dictionary form in the lookup tables and the word forms found in real text – together with the adopted solutions – are the following:

- (a) Hyphen/space alternations: For hyphenated names such as *Jean-Marie*, *Nawaf al-Ahmad al-Jaber al-Sabah* or the place name *Saint-Jean*, we automatically generate patterns to capture both the hyphenated and the non-hyphenated forms (e.g. `Jean[\- \]Marie`).
- (b) Diacritic variations: Words that carry diacritics are often found without the diacritic. For example, *François Chérèque* is often written *Francois Chereque*, Schröder as Schroder, Lech Wałęsa as Lech Walesa, Raphaël Ibañez as Raphael Ibanez, etc.. For each name with diacritics, we therefore generate a pattern that allows both alternatives (e.g. `Fran(ç|c)ois Ch(é|e)r(è|e)que`).
- (c) Further common variations: Especially for place names, there are a number of very common variations, including the abbreviation of name parts such as *Saint* to *St* (with or without the final dot) or the use of a slash instead of common name parts. For instance, *Nogent-sur-Seine* and *Frankfurt am Main* can be found as *Nogent/Seine* and *Frankfurt/Main* (also: *Frankfurt a. Main*), etc. For all such names, we thus pre-generate the various common variants.

- (d) Name inversion: While news texts in most languages mention the given name before the surname, the opposite order can also be found in some languages. In Hungarian, for example, local names are usually written with the last name first, while foreign names have the opposite order. The lookup procedure must consider this variation, as well.
- (e) Morphological declensions: In some languages (especially the Balto-Slavonic and Finno-Ugric languages), person names can be declined. In Polish, for example we can find the inflected form *Nicolasowi Sarkozy'emu* (or – less frequent – *Nicolasowi Sarkoziemu*) referring to the French president *Nicolas Sarkozy*. Similarly *Tony'ego Blaira* or *Toniego Blaira* are found for the former British prime minister. For these languages, we pre-generate morphological variants for all known names according to rules that will be discussed below. It must be high-lighted that – in some languages – variations can also affect the beginning of the name. For instance, for the Irish place name *Gaillimh* (Irish version of *Galway*), we can find *nGaillimh* (in *Galway*). For some languages, the number of different inflections can be rather high.
- (f) Typos: Even in the printed press, typos are relatively frequent. This is especially the case for difficult names such as *Condoleezza Rice*. For this person's given name, we found the typos *Condoleza*, *Condaleezza*, *Condollezza* and *Condeleeza*, each more than once.
- (g) Simplification: In order to avoid repetition, names such as *Condoleezza Rice* and *George W. Bush* are frequently simplified to *Ms. Rice* and *President Bush*.
- (h) Transliteration: Names are normally transliterated into the target language writing system. In the case of NewsExplorer, we are mainly interested in *Romanisation*, i.e. in using the Latin script as a target language. Depending on the target language, transliteration rules often differ so that two different Romanised versions of the same name can co-exist. For example, the Russian name Владимир Устинов is typically transliterated to *Wladimir Ustinow* in German, to *Vladimir Ustinov* in English, to *Vladimir Ustinov* in Spanish, to *Vladimir Oestinov* in Dutch and to *Vladimir Oustinov* in French. Conversely the French city *Strasbourg* is sometimes written in Russian Страсбург (/strasburg/) sometimes Стразбург (/strazburg/), in Ukrainian Страсбур (/strasbur/), in Serbian Стразбур (/strazbur/), without the final 'g' mimicking the original French pronunciation.
- (i) Vowel variations, especially from and into Arabic: In Arabic and some other languages, short vowels are not always written. The string محمد (Mohammed) contains only the four consonants Mhmd, which is the reason why so many variants exist for this name: *Mohammed*, *Mohamed*, *Mahmoud*, *Muhamad*, and more.

5.2. Generating variants for known names

The variation types (a) to (d) are rather generic so that it is not difficult to pre-generate the most common name variants, as shown in the previous section. Morphological variations such as those shown in (e) are much harder to predict and they differ from one language to the next. This section describes how this variation type can be dealt with. For the variation types (f) to (i), see Section 6 below.

Profound linguistic skills and native speaker competence will help to produce good suffix addition and suffix replacement rules. [27], for instance, have produced extensive inflection rules for Serbian. However, in order to achieve high multilinguality, it is important to find an efficient and quick method to generate at least the most common variants. We found that, even without native speaker competence, it is possible to iden-

tify a number of frequent inflection rules purely by observing common variations for known names. These can be found by searching text collections using several generous regular expressions such as `Nicol.*Sarko[[:alpha:]]+` (for the French president) allowing to capture name variants and by then looking for regularities. In Romanian, for example, one will discover that known names are frequently accompanied by the suffixes `-ul` and `-ului` (suffix addition), and that the endings are `-l` and `-lui` if the name already ends in `-u`. If the name ends with `-a` we frequently find the `-ie` ending (suffix replacement). By collecting a number of such observations, we can produce suffix addition and replacement rules to pre-generate – for all names in the lookup tables – the most frequent variants. For the names *Paris*, *Bacău* and *Venezia*, for example, we can then generate the variants *Parisul*, *Parisului*, *Bacăul*, *Bacăului*, *Venezia* and *Veneziei*. Slavic languages are more complex, but the same principle holds. For the Slavic language Slovene, the regular expression substitution rule for person names is:

$$s/[aeo]?/(e|a|o|u|om|em|m|ju|jem|ja)?/$$

meaning that – for every name ending in `-a`, `-e` or `-o` – we pre-generate ten different variants, ending in `-e`, `-a`, `-o`, `-u`, `-om`, `-em`, `-m`, `-ju`, `-jem` and `-ja`, respectively. For every frequent known name in our name database such as the previous Lebanese political leader *Pierre Gemayel*, for instance, we will thus generate the pattern:

$$Pierr(e|a|o|u|om|em|m|ju|jem|ja)?\ Gemayel(e|a|o|u|om|em|m|ju|jem|ja)?$$

That way, *Pierrom Gemayelom* and any other of the other possible combinations will be recognised in text. Note that over-generation, i.e. producing name variants that do not exist in the language, is not normally a problem because they will simply not be found. However, especially short names can lead to over-generous patterns and new patterns should always be tested on large document collections before being applied in a real-world scenario.

To give an indication of the effort required: it takes us usually between 1 hour and 2 days to produce basic working lists of inflection patterns for a new language.

An alternative to *generating* morphological variants for highly inflective languages would be to map different name variants found in text using a mixture of string distance metrics and automatically acquired suffix-based lemmatisation patterns, as it was proposed by [28] for Polish.

5.3. Transcription of names written in different writing systems

Names from a language using a different writing system are usually transliterated. Transliteration rules are simple, normally hand-written rules mapping characters or sequences of characters from one writing system to their counterparts in another writing system ([29]). Some Greek examples are:

- $\psi \Rightarrow ps$
- $\lambda \Rightarrow l$
- $\mu\pi \Rightarrow b$

Frequently, more than one transliteration system exists for the same language pair (the Arabic first name *سعيد* is mainly transliterated in French as *Saïd* but sometimes also as *Said*, *Sayyed* and *Saeed*), which explains why different target language versions may exist for the same name and the same source-target language pair. As pointed out in bullet (h) in Section 5.1, transliteration rules usually differ depending on the target lan-

guage. It is less common knowledge that transliteration also exists for languages using the same writing system. For example, the name *Bush* will be found in Latvian language as *Bušs*. We deal with intra-script transliteration in the normalisation step described in Section 6. At the JRC, we use transliteration rules for the following scripts:

- Cyrillic (used for Russian, Bulgarian and Ukrainian):
Симеон Маринов → Simeon Marinov;
- Greek: Κώστας Καραμανλής → Kostas Karamanlis;
- Arabic (used for Arabic, Farsi and Urdu; some additional transliteration rules were added for Farsi and Urdu):
جلال طلباني → jlal tlbani (“Jalal Talabani”);
- Devanagari (used for Hindi and Nepalese):
सोनिया गांधी → soniya gandhi.

Transliteration sometimes produces name forms that are rather different from the usual spelling. For frequent names, it is thus appropriate to hard-code the transliteration of the full name. Here are some examples of source language strings, their letter-by-letter transliteration and the aimed for target language form:

- Russian Джордж → [Djordj] → *George*;
- Russian Джеймс → [Djaims] → *James*;
- Hindi डब्ल्यु → [dableyu] → W (as in *George W. Bush*);

All other spelling differences of transliterated versus non-transliterated names are dealt with in the normalisation step, described in the next section. Adding transliteration tables for new languages using letters (alphabetical scripts) or syllables is not difficult.²⁷ Adding rules for Hindi took us two hours. Dealing with ideographic languages such as Chinese is harder and needs different procedures.

6. Rules to identify name variants

In Section 5.1, we have seen a number of reasons why variants exist for the same name. After having applied named entity recognition in up to nineteen languages over a period of about five years, we have found up to 170 variants for the same name.²⁸ Identifying these variants as referring to the same person has many advantages, including improved search and retrieval, as well as more accurate results for tasks where person co-references are required such as social network generation based on quotations ([30]) or on co-occurrence ([1]; see also Feldman’s chapter on *Link Analysis in Networks of Entities*, in this volume). There is thus a clear need for methods to identify whether similar, but different names found in text are variants belonging to the same person or not. There are a number of known approaches to identify name equivalences for specific language *pairs*. In this section, we present an alternative approach, which is language and language pair-independent. It consists of normalising names into an abstract *consonant signature*. This consonant signature can then be used as the basis for comparing all (normalised) names found. All name pairs that have a similarity above a certain threshold will be marked as referring to the same person. The threshold was set so that only good name pairs would be merged for a given test set. The idea behind this is

²⁷ Various transliteration tables can be found at the *Institute of the Estonian Language* and the *American National Geospatial Agency* (<http://transliteration.eki.ee/>; <http://earth-info.nga.mil/gns/html/romanization.html>).

²⁸ The terrorist Abu Musab al-Zarqawi: see <http://press.jrc.it/NewsExplorer/entities/en/285.html>.

that having two entries for the same person is less harmful than that of merging two different persons. For details on the name normalisation and name merging processes, see [26].

For every name found in the analysis of news articles in 19 languages carried out daily by the NewsExplorer application, we first check whether the name already has an entry in the NewsExplorer name database. Both main names (aliases) and known name variants are considered. All unknown names will be normalised (see Section 6.1) and compared to the consonant signature of any of the known names. If any of the consonant signatures coincide, name similarity measures will be applied (see Section 6.2). If successful, the new name variant will be added to the existing name. Otherwise, the name will be added as a new name into the database. Names found only ever once are otherwise ignored in order to avoid typos entering the database. If a name is found at least five times, this name gets the status of a frequent name so that it will be found by lookup in any future news articles (see Section 5).

6.1. Language-independent name normalisation rules

The idea behind name normalisation is to create a language-independent representation of the name. In principle, a representation of the (approximate) pronunciation of the name would be a good normalised form, as suggested by the *Soundex* algorithm for English ([31]). However, in order to infer the pronunciation of a name, it is necessary to know the original language of the name. In news articles, this information is not normally known as known persons from around the world are being talked about in the newspapers of any other country. To give an example: the name of the previous French president *Chirac* (pronounced in French as /ʃiʁak/) would be pronounced as /kiʁak/ if it were an Italian name, as /çirak/ in German, etc., while the name *Chiamparino* (Mayor of the city of Turin) should be pronounced as /kiamparino/. Automatic identification of the origin of a name ([32]) produces moderate results.

In order to overcome this problem, empirical observations on name spelling differences for the same name in many different languages have been used to produce normalisation rules that will be applied to all names, independently of their origin. The following are some examples:

- Name-initial ‘Wl-’ and the name-final ‘-ow’ for Russian names will get replaced by ‘Vl-’ and ‘-ov’. This is to accommodate the typical German transliteration for names like *Vladimir Ustinov* as *Wladimir Ustinow*.
- The Slovene strings ‘š’, the Turkish ‘ş’, the German ‘sch’, the French ‘ch’ will get replaced by ‘sh’ in order to neutralize frequent spelling variants such as *Başar al Asad*, *Baschar al Assad*, *Bachar al Assad* and *Başar Esad*.
- The French letter combination ‘ou’ will get replaced by ‘u’ to capture the regular transliteration differences for the sound /u/ in names like *Ustinov*, which is spelled in French as *Oustinov*.
- The letter ‘x’ will get replaced by ‘ks’, etc.

These normalisation rules are exclusively driven by practical requirements and have no claim to represent any underlying linguistic concept. They are written as a set of regular expression substitutions. For example, the normalisation rule for the Slavic ending –*ski* is written by the substitution rule below and will normalise the following names to a common suffix: *Stravinski*, *Stravinsky* (Finnish), *Stravinskij* (Slovak), *Sztravinszkij* (Hungarian), *Stravinskij* (Icelandic):

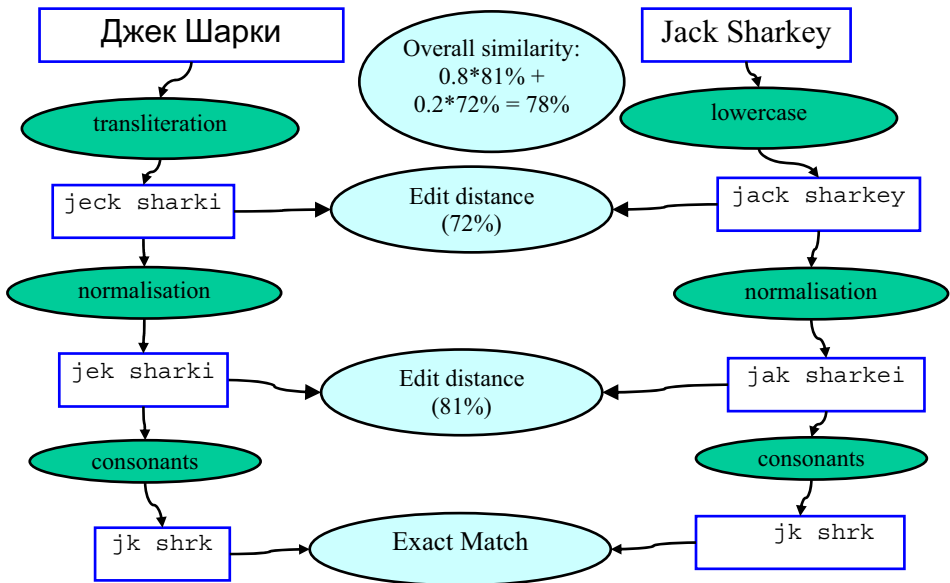


Figure 1. The Figure shows the transliteration and normalisation steps, as well as the similarity measurement applied to a name pair. The two names will not be associated to the same person because the overall similarity is 0.78 and thus lower than the requested threshold of 0.94.

$s/sz?k [y\acute{i}\acute{i}] j?/ski/$ (At the end of a word)

The final normalisation step consists of removing all vowels. Vowels frequently get distorted during transliteration and their removal is compulsory for languages using the Arabic script as short vowels are not normally written in Arabic. An original Russian name such as Джек Шарки will thus go through the stages transliteration (*jeck sharki*), normalisation (*jek sharki*) and vowel removal (*jk shrk*).

6.2. Similarity measure used to compare names with the same normalised form

All names having the same normalised form are considered name variant *candidates*. For each candidate pair, the edit distance similarity measure is applied twice, one time each on two different representations of the name: once between the normalised forms with vowels and once between the lowercased transliterated forms (see Figure 1). The two similarities have relative weights of 0.8 and 0.2. By applying the similarity measure only to name pairs with an exact signature match, we miss some good candidates, but the pre-filtering saves a lot of precious computing time. For further details, see [26].

Both the name normalisation and the similarity calculation steps are applied across the board to all names found in the currently 19 NewsExplorer languages and do not depend on the input language. There is thus no need for language pair-specific training or mapping rules. However, when adding a language with a new script, a new set of transliteration rules needs to be added and – depending on the language – it may be useful to add additional normalisation patterns (which will then be applicable to all languages).

7. Quotation recognition

Direct speech quotations can typically be identified in text due to the presence of quotation markers (e.g. single or double, lower or upper quotes (“, ’), single or double angled brackets (<<, >>, “, ”, etc.)), a reporting verb (*says, quotes, criticises, objects*, etc.) and a reference to the person who makes the quote. In [30], we present a number of language-independent patterns that allow to recognise quotations in running text. The most frequent quotation pattern is covered by the following rule:

name [, up to 60 chars .] *reporting-verb* [:[that] *quote-mark* QUOTE *quote-mark*
e.g. *John Smith, supporting AFG, said: "Here we are!"*.

By keeping the language-independent quotation recognition patterns separate from the language-specific lists of reporting verbs, the quotation recognition software can very easily be extended to new languages. For language-specific issues, see Section 10.

8. Date recognition

Date recognition is a known named entity recognition task (e.g. MUC-7)²⁹, but recognition patterns for date formats other than numerical dates (such as *26.05.2009*) are usually developed for individual languages (e.g. [10]). Our approach, described in [33], completely separates the language-independent rules from language-specific parameter files, for numerical and alpha-numeric date formats. The tool detects dates in full form format (such as *26th of May of the year two thousand and nine*), but also partial dates such as *26 May* or *May 2009*. Rules make reference to slot fillers, for which the language-specific expressions can be found in parameter files. The following rule captures complete dates with the day mentioned before the month (simplified pseudo-code).

(0..31|CARDINAL|ORDINAL)(SEPARATOR|MONTH-INTRO)
(0..12|MONTH)(SEPARATOR|YEAR-INTRO)
(1800..2099|00..99|YEAR-IN-LETTERS)

CARDINAL stands for a cardinal number in letters (e.g. *twenty-one, vingt-et-un*), *ORDINAL* for an ordinal number in letters or numbers (*1st, third*, French: *1^{er}, troisième*, etc.), *SEPARATOR* for a common character used to separate numbers (e.g. */*), *MONTH-INTRO* for a common expression to introduce the month (*twenty-sixth of May*; Spanish: *catorce de febrero*), *MONTH* for a month name or an equivalent abbreviation (e.g. *July, Jan.*), *YEAR-INTRO* for a common expression to introduce the year (*26th of May in the year 2009*; Romanian: *întîi martie al anului 2009*), and *YEAR-IN-LETTERS* for the full or partial year in letters (*sixty-four, nineteen eighty*; French: *deux mille un*). Variants of these words (e.g. words with and without diacritics, with or without case endings) should also be listed. Language-specific parameter files for date recognition are simple and can be created within half a day's work per language (including variants).

Procedures to resolve relative date expressions (e.g. *last May*), to disambiguate ambiguous date formats (e.g. the European vs. the US-American reading of a date like *10-05-2009*: 10 May vs. 5 October) and to convert dates from other calendars (e.g. Arabic *01/06/1430*, which is equivalent to *26/5/2009*) are obviously language-

²⁹ See http://www.itl.nist.gov/iad/894.02/related_projects/muc/proceedings/ne_task.html for the MUC-7 task specification (last visited 6 May 2008).

independent. Date expressions found can be normalised for storage and improved retrieval.

9. Cross-lingual document similarity calculation

Document similarity calculation across languages can be useful for news navigation (e.g. in NewsExplorer), for multilingual topic detection and tracking, for cross-lingual plagiarism detection, in query-by-example scenarios, and possibly more. We have shown in Section 2 that existing methods for cross-lingual document similarity (CLDS) calculation are bilingual in nature and are thus limited with respect to the number of language pairs. NewsExplorer can calculate CLDS for currently 19 languages (171 language pairs), although one of the ingredients of the formula (Eurovoc indexing) is not available for the six non-EU languages covered, so that the software is fully functional for language pairs involving *only* 13 languages.

The NewsExplorer approach (described in detail in [33]) is, in principle, almost language-independent. The main idea behind it is to find a language-independent representation for each of the texts written in different languages, i.e. to find a collection of anchors that serve to establish the link between texts in different languages. The CLDS in NewsExplorer is based on four ingredients (represented each as a vector). The first three of these are represented in a language-neutral manner:

1. A weighted list of subject domain classes using the Eurovoc thesaurus ([35]). The Eurovoc classes are represented by their numerical identifier.
2. A frequency list of countries to which each text makes a reference, either direct (country name) or indirect (city name, country adjective, citizen name, etc.; see Section 3). Countries are represented by their country ISO 3166 code.
3. A frequency list of persons made reference to in each of the texts. The persons are represented by their numerical identifier in the multilingual name database, which subsumes various name variants under the same identifier (see Section 4).
4. A weighted list of monolingual keywords. While most words of texts in different languages will not match, we found that there is frequently a number of cognates, numbers, acronyms, names or name parts that match between the languages.

Each document or group of documents (NewsExplorer works with clusters of related documents) is thus represented by four language-neutral vectors. For each pair of documents in different languages, the CLDS formula calculates the cosine similarities between each of the four vectors and combines them with the relative weight of 0.4, 0.3, 0.2 and 0.1 in order to come up with an overall CLDS. The higher the overall similarity is, the more similar the two (clusters of) documents are. In NewsExplorer, document pairs below the similarity threshold of 0.3 are ignored, i.e. the documents are treated as if they were not related. The similarity formula itself is language-independent, but language-specific tools obviously need to be developed to produce the language-neutral representation. A big advantage is, however, that no *language pair-specific* procedures or resources are needed. New languages can be plugged in without any additional effort.

10. Language-specific issues – the limitations of language-independence

In previous sections, we have already seen that some language-specific resources are needed in addition to the language-independent rules, including:

- Gazetteers of place names (for geo-tagging);
- Geo-stop word lists (for geo-tagging);
- Lists of trigger pattern components (for person and organisation name recognition);
- Lists of inflection patterns for all lookup procedures (regular expressions to add or substitute suffixes);
- Lists of days of the week, months, numbers in words, etc. (for date recognition).

Apart from the gazetteers, these resources can be produced relatively quickly by exploiting bootstrapping methods, as described earlier. Additionally, there are a number of issues where language-specific rules or procedures are required or will at least help improve the results. These are:

- (a) Diacritic normalisation: In some Italian sources, diacritics are replaced by an apostrophe following the word (e.g. *Libertà* – freedom – can be found as *Liberta*).
- (b) Case information: Some languages (e.g. Arabic, Farsi, Hebrew, Chinese, Japanese, Hindi, etc.) do not distinguish upper and lower case so that every word in the text needs to be considered in a lookup procedure instead of considering only uppercase words. This also has the consequence that it is harder to guess where a person name ends. The opposite also holds for languages like German, which write nouns or other non-names in uppercase: name boundaries are less easily detected because uppercased nouns may follow a person name.
- (c) Missing vowels: In Arabic, Hebrew and – to some extent – Hindi, short vowels are not normally written (the Arabic spelling of the name *Mohammed* consists of the four consonants *mhmd* only). The result of transliterating names written with the Arabic script is thus often different from their Latin counterpart.
- (d) Tokenisation (no word separators): In Chinese, Thai and some other languages, words are not separated by spaces. Words could thus start at any character in a string of characters. This needs to be considered in the lookup procedures.
- (e) Tokenisation (apostrophe): Different, language-specific tokenisation rules are required to deal with the following strings (which all contain proper names), because the apostrophe sometimes marks the word border and sometimes it is part of the name: *Le Floc'h*, *Tony Blair's*, *Jan Figel'*, *Stéphane M'Bia*, etc.
- (f) Tokenisation (agglutinative languages): In languages such as Turkish, Hungarian and Finnish, various particles (e.g. prepositions or determiners) may be attached to the end of any word, including names. This increases the number of possible suffixes to be considered in lookup procedures enormously. It is possible that morphological analysis software will eventually be needed for such languages to achieve good results. At the same time, our analysis has shown that the number of proper noun endings – even for Finnish – is limited.
- (g) Compounding: In languages like German, nouns can be combined with (mostly) other nouns to produce compound nouns. For instance, the words *Bundesverkehrsminister* consists of *Bund* (Federal), *Verkehr* (transport) and *Minister*. As this combination is productive, it is not possible to list all combinations. This is obviously a problem for producing trigger word lists for name recognition. Regular expressions with generous wild cards must be used word-initially (*.*minister*), which could lead to problems regarding the computational performance of the sys-

tem. The Finnish word *Lontoolaishotelliin* (to a hotel in London) shows that compounding and agglutination can also occur combined.

- (h) Affix types: while most Western-European languages almost exclusively use suffixes for name inflection, Irish also uses prefixes (*le hAngela Carter* "with Angela Carter"). We are not aware of any language using infixes for names.
- (i) Order of first name and family name: As discussed in Section 5.1, given and last names of local names are inverted in Hungarian, but not in foreign names. This must be considered in the lookup procedure.
- (j) Quotation markers (see Section 7): In most NewsExplorer languages, the information on the speaker and the reporting verb are found *outside* the quotation, but Italian and Swedish allow to move them to a place inside the quotation markers (see [30]) (e.g. "Sacco e Vanzetti – ha ricordato Nichi Vendola – erano due emigranti" / "Sacco and Vanzetti were two emigrants" reminded Nichi Vendola).

11. Conclusion

In this chapter, we proposed a number of guidelines that are useful when aiming at building multi-monolingual and cross-lingual applications for larger numbers of languages (Section 1.3). These guidelines are: (a) To the largest extent possible, use language-independent rules instead of language-specific ones. (b) Reduce the usage of language-specific and especially *language pair*-specific resources to a minimum. (c) Where they are necessary, keep them in language-specific parameter files to keep the applications modular so that new languages can be *plugged in*. (d) For those resources that cannot be avoided, use bottom-up bootstrapping methods to create the resources. (e) For *cross-lingual* applications, attempt to use a language-neutral document representation to reduce the effect of the near-exponential complexity increase when dealing with many languages.

To our knowledge, other system developers do not work according to these seemingly obvious and simple principles. Instead, the most common approach is to develop tools for one language and to then adapt these tools to further languages (See Section 2 on related work). We believe that tools for new languages will be developed quicker and more easily if the proposed guidelines are followed.

In order to show that putting these principles into practice is feasible for a range of different applications, we described such rule sets – and, where necessary, the simple acquisition procedures for the related resources – for seven applications: geo-tagging (Section 3); person and organisation name recognition (4); processing of name inflection (morphological variation) and other surface form variations (5); name variant mapping within the same language or across different languages and scripts (6); the recognition of quotations (7), date recognition (8), and cross-lingual document similarity calculation (9). These application examples are followed by a list of specific issues where the applicability of language-neutral rules ends and where language-specific adaptations are needed.

All the applications described in this chapter have been implemented to an operational level. The first six of them are used daily in the public large-scale news aggregation, analysis and exploration system NewsExplorer, which analyses 35,000 news articles per day in the 19 languages Arabic, Bulgarian, Danish, Dutch, English, Estonian, Farsi, French, German, Italian, Norwegian, Polish, Portuguese, Romanian, Russian, Slovene, Spanish, Swedish and Turkish.

12. Acknowledgment

We would like to thank the many people who – in the course of several years – have contributed with the knowledge of their languages to produce language-specific resources and to test the results. We would like to thank the whole team of the *Web Mining and Intelligence* group at the JRC – and especially the team leaders Clive Best and Erik van der Goot – for having provided the reliable and high-quality news data in a standardised and clean format. We are grateful to our colleagues Ken Blackler, Flavio Fuat and Martin Atkinson for making the results of our work accessible to the wider public using robust web portals, and to Jenya Belyaeva for her quality control.

13. References

- [1] Pouliquen Bruno, Ralf Steinberger & Jenya Belyaeva (2007). Multilingual multi-document continuously updated social networks. Proceedings of the RANLP Workshop Multi-source Multilingual Information Extraction and Summarization (MMIES'2007). Borovets, Bulgaria.
- [2] Tanev Hristo (2007). Unsupervised Learning of Social Networks from a Multiple-Source News Corpus. Proceedings of the RANLP Workshop *Multi-source Multilingual Information Extraction and Summarization* (MMIES'2007). Borovets, Bulgaria.
- [3] Pouliquen Bruno & Ralf Steinberger (in print). Automatic Construction of a Multilingual Name Dictionary. In: Cyril Goutte, Nicola Cancedda, Marc Dymetman & George Foster (eds.): *Learning Machine Translation*. MIT Press.
- [4] Larkey Leah, Fangfang Feng, Margaret Connell & Victor Lavrenko (2004). Language-specific Models in Multilingual Topic Tracking. Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 402-409.
- [5] Gamon Michael, Carmen Lozano, Jessie Pinkham & Tom Reutter (1997). Practical Experience with Grammar Sharing in Multilingual NLP. In Proceedings of ACL/EACL, Madrid, Spain.
- [6] Rayner Manny & Pierrette Bouillon (1996). Adapting the Core Language Engine to French and Spanish. Proceedings of the International Conference NLP+IA, pp. 224-232, Mouncton, Canada.
- [7] Pastra Katerina, Diana Maynard, Oana Hamza, Hamish Cunningham & Yorick Wilks (2002). How feasible is the reuse of grammars for Named Entity Recognition? Proceedings of LREC, Las Palmas, Spain.
- [8] Carenini Michele, Angus Whyte, Lorenzo Bertorello & Massimo Vanocchi (2007). Improving Communication in E-democracy Using Natural Language Processing. In *IEEE Intelligent Systems* 22:1, pp 20-27.
- [9] Maynard Diana, Valentin Tablan, Hamish Cunningham, Christian Ursu, Horacio Saggion, Kalina Bontcheva & Yorick Wilks (2002). Architectural Elements of Language Engineering Robustness. *Journal of Natural Language Engineering* 8:3, pp 257-274. Special Issue on Robust Methods in Analysis of Natural Language Data.
- [10] Bering Christian, Witold Drożdżyński, Gregor Erbach, Lara Guasch, Peter Homola, Sabine Lehmann, Hong Li, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, Atsuko Shimada, Melanie Siegel Feiyu Xu & Dorothee Ziegler-Eisele (2003). Corpora and evaluation tools for multilingual named entity grammar development. Proceedings of the Multilingual Corpora Workshop at Corpus Linguistics, pp. 42-52, Lancaster, UK.
- [11] Glover Stalls Bonnie & Kevin Knight (1998). Translating Names and Technical Terms in Arabic Text. In Proceedings of the ACL-CoLing Workshop 'Computational Approaches to Semitic Languages'.
- [12] Lee C.-J., J. S. Chand, and J.-S.R. Jang (2006). Extraction of transliteration pairs from parallel corpora using a statistical transliteration model. *Information Science* 17, 6, pp. 67-90.
- [13] Oh Jong-Hoon and Key-Sun Choi (2002). An English-Korean transliteration model using pronunciation and contextual rules. Proceedings of ACL.
- [14] Knight Kevin & Jonathan Graehl (1998). Machine Transliteration. *Computational Linguistics* 24:4, pp. 599-612.
- [15] Qu Yan & Gregory Grefenstette (2004). Finding ideographic representations of Japanese names written in Latin script via language identification and corpus validation. Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, Barcelona, Spain.

- [16] Leek Tim, Hubert Jin, Sreenivasa Sista & Richard Schwartz (1999). *The BBN Crosslingual Topic Detection and Tracking System*. In 1999 TDT Evaluation System Summary Papers.
- [17] Wactlar H.D. (1999). New Directions in Video Information Extraction and Summarization. In Proceedings of the 10th DELOS Workshop, Sanorini, Greece.
- [18] Saralegi Urizar Xabier & Iñaki Alegria Loinaz (2007). Similitud entre documentos multilingües de carácter científico-técnico en un entorno Web. Proceedings of SEPLN, Seville, Spain.
- [19] Landauer Thomas & Michael Littman (1991). A Statistical Method for Language-Independent Representation of the Topical Content of Text Segments. Proceedings of the 11th International Conference 'Expert Systems and Their Applications', vol. 8: 77-85. Avignon, France.
- [20] Vinokourov Alexei, John Shawe-Taylor & Nello Cristianini (2002). Inferring a semantic representation of text via cross-language correlation analysis. Proceedings of Advances of Neural Information Processing Systems 15.
- [21] Steinberger Ralf, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, Dan Tufiş & Dániel Varga (2006). The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. Proceedings of the 5th International Conference on Language Resources and Evaluation LREC, pp. 2142-2147. Genoa, Italy.
- [22] Mikheev A., M. Moens & C. Gover (1999) Named Entity Recognition without Gazetteers. In Proceedings of EACL, Bergen, Norway.
- [23] Leidner Jochen (2007). Toponym Resolution in Text: Annotation, Evaluation and Applications of Spatial Grounding of Place Names. Ph.D. thesis, School of Informatics, University of Edinburgh, Edinburgh, Scotland, UK.
- [24] Pouliquen Bruno, Marco Kimler, Ralf Steinberger, Camelia Ignat, Tamara Oellinger, Ken Blackler, Flavio Fuat, Wajdi Zaghouni, Anna Widiger, Ann-Charlotte Forslund, Clive Best (2006). Geocoding multilingual texts: Recognition, Disambiguation and Visualisation. Proceedings of the 5th International Conference on Language Resources and Evaluation LREC, pp. 53-58. Genoa, Italy.
- [25] Bronstein I. N., K. A. Semendjajew, G. Musiol & H Muhlig (1999). Taschenbuch der Mathematik (4. ed.). Frankfurt am Main, Thun: Verlag Harri Deutsch.
- [26] Steinberger Ralf & Bruno Pouliquen (2007). Cross-lingual Named Entity Recognition. In: Satoshi Sekine & Elisabete Ranchhod (eds.). *Lingvisticæ Investigations LI 30:1*, pp. 135-162. Special Issue Named Entities: Recognition, Classification and Use.
- [27] Vitas Duško, Cvetana Krstev & Denis Maurel (2007). A note on the Semantic and Morphological Properties of Proper Names in the Prolex Project. In: Satoshi Sekine & Elisabete Ranchhod (eds.). *Lingvisticæ Investigationes LI 30:1*, pp. 115-134. Special Issue Named Entities: Recognition, Classification and Use.
- [28] Piskorski Jakub, Karol Wieloch, Mariusz Pikula & Marcin Sydow (2008). Towards Person Name Matching for Inflective Languages. In: Proceedings of the WWW'2008 workshop 'Natural Language Processing Challenges in the Information Explosion Era'. Beijing, China.
- [29] Daniels Peter T. & William Bright (eds.) (1996). *The World's Writing Systems*. Oxford University Press, Oxford, UK.
- [30] Pouliquen Bruno, Ralf Steinberger & Clive Best (2007). Automatic Detection of Quotations in Multilingual News. In: Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP. Borovets, Bulgaria.
- [31] Hall P. and G. Dowling (1980). Approximate string matching. *Computing Surveys*, 12:4, pp. 381-402.
- [32] Konstantopoulos Stasinou (2007). What's in a name? quite a lot. In Proceedings of the RANLP workshop 'Workshop on Computational Phonology'. Borovets, Bulgaria.
- [33] Ignat Camelia, Bruno Pouliquen, António Ribeiro & Ralf Steinberger (2003). Extending an Information Extraction Tool Set to Central and Eastern European Languages. Proceedings of the RANLP Workshop Information Extraction for Slavonic and other Central and Eastern European Languages (IESL'2003). Borovets, Bulgaria.
- [34] Steinberger Ralf, Bruno Pouliquen & Camelia Ignat (2005). Navigating multilingual news collections using automatically extracted information. *Journal of Computing and Information Technology - CIT* 13, 2005, 4, 257-264.
- [35] Pouliquen Bruno, Ralf Steinberger & Camelia Ignat (2003). Automatic Annotation of Multilingual Text Collections with a Conceptual Thesaurus. In: Proceedings of the EUROLAN Workshop Ontologies and Information Extraction at the Summer School The Semantic Web and Language Technology - Its Potential and Practicalities. Bucharest, Romania.