Language Engineering for Lesser-Studied Languages
S. Nirenburg (Ed.)
IOS Press, 2009
© 2009 IOS Press. All rights reserved.
doi:10.3233/978-1-58603-954-7-81

Developing Proper Name Recognition, Translation and Matching Capabilities for Low- and Middle-Density Languages

Marjorie MCSHANE[1]

Institute for Language and Information Technologies
University of Maryland Baltimore County

Abstract. This article discusses methods for developing proper name recognition, translation and cross-linguistic matching capabilities for any language or combination of languages in a short amount of time, with relatively minimal work by native speaker informants. Unlike much work on proper name recognition, this work is grounded in knowledge-based rather than stochastic methods, and it extends to multi-lingual and multi-script name processing.

Keywords. low-density languages, proper name recognition, preprocessing

1. Introduction

The recognition of proper names is a basic need of natural language processing (NLP) systems. It is typically handled during the first stage of processing, known as preprocessing. Other phenomena commonly bundled into preprocessing include the determination of word and sentence boundaries, the recognition of dates and numbers, the stripping of metadata and non-textual content (as when processing Web pages), and so on. Recognizing proper names is important for systems that involve syntactic parsing and/or semantic analysis because proper names are not listed in typical lexicons (if they are listed at all, it is in gazetteers or onomastica) and multi-word proper names behave as a single constituent: for example, in the sentence *The Duke of Wellington crossed the Strait of Dover*, *the Duke of Wellington* functions as the subject and *the Strait of Dover* functions as the direct object.

Recognizing proper names is only one of many possible needs of NLP applications. In addition, an application might need to translate proper names, extract information about particular people, places, etc., from texts written in different languages and scripts, or help someone who heard a proper name in speech – but does not know for certain how to write it – look it up in some knowledge base.

This article describes two systems that were designed to treat different aspects of proper name processing. Like all systems, their specific features derive from a combination of their goals, the knowledge and computing resources available, the manpower devoted to their development, and theoretical and practical preferences of

[1] Department of Computer Science and Electrical Engineering, ITE 325, 1000 Hilltop Circle, Baltimore, Maryland, 21250, USA; E-mail: marge@umbc.edu.

developers. However, despite these idiosyncrasies, the conceptual substrate and lessons learned are quite generalizable to work on proper names in any language for any application.

The article is organized as follows. Section 2 describes the so-called "named entity recognition task" and overviews selected approaches and systems developed to carry it out. Section 3 describes the Boas II knowledge elicitation system, which elicits knowledge about proper names from speakers of any language and automatically converts that knowledge into a proper name recognition engine. Section 4 describes the GeoMatch geographical entity recognition and matching system, whose goal is to expand the data and enhance the search capabilities of an existing geographical database in order to make it more useful for multilingual applications. Section 5 concludes the paper.

2. The "Named Entity Recognition" Task

The automatic recognition of so-called "named entities" – which include proper names, numbers and dates – attracted much attention in response to the Named Entity Task of the Message Understanding Conferences, or MUCs (Chinchor 1997) [4]. This series of conferences, held from 1987 to 1997, were evaluation exercises in which participants competed in knowledge extraction tasks as well as a number of specialized subtasks, like reference resolution and named entity extraction. Participants were provided with descriptions of each task, the necessary output format for their systems, and an annotated corpus that could be used for training. Prior to corpus annotation, precise annotation guidelines had to be developed for each subtask. These guidelines are instructive as they elucidate the complexity of what might, at first blush, seem like straightforward phenomena. Taking proper names as an example, the MUC-7 task definition states that family names like *the Kennedys* are not to be tagged, nor are diseases, prizes, etc., named after people: *Alzheimer's, the Nobel prize*. Titles like *Mr.* and *President* are not to be tagged as part of the name, but appositives like *Jr*. and *III* ("the third") are. For place names, compound place names like *Moscow, Russia* are to be tagged as separate entities and adjectival forms of locations are not to be tagged at all: *American companies*. These decisions and many others like them are typically not universally agreed upon but rather reflect necessary compromises influenced by issues such as how difficult it will be to train annotators to follow the conventions and how useful the final corpus will be for a wide variety of NLP practitioners.

The rules of the game for the MUC competitions significantly affected the methods selected by participants: since participants were provided with large annotated corpora, stochastic approaches were favored. However, annotating corpora is expensive: it has been reported that tagging 100,000 words for syntactic features requires at least 33 hours by trained taggers (Bikel, Schwartz, and Weischedel 1999); and 100,000 words isn't even very much for stochastic training. Therefore, the predominance of stochastic systems over pattern-matching (i.e., knowledge-based) ones must be interpreted in context: if a similar competition were launched on low-density languages with no corpora provided, the research efforts in the field might well have taken a different turn.

Both stochastic and pattern-matching methods have produced good results from the best systems: in the 90's as the F-score, which is calculated as a combination of recall and precision.[2] However, both approaches generally require (a) a significant amount of static knowledge and (b) morphological and/or syntactic processors, making them not entirely applicable to low- and middle-density languages. Consider the knowledge needs of some noteworthy systems:

- BBN's IdentiFinder [3] uses a hidden Markov model and a minimum of 100,000 words of training data to learn to carry out named entity recognition for a language. However it performs much better with a million words of training data, which would require months of manual annotation. IdentiFinder uses bigrams rather than trigrams because using trigrams would require "exponentially more training data".
- An experiment in the supervised learning of named entity recognition for Greek involved bootstrapping from English [9]. The Greek resources needed to exploit this bootstrapping methodology include a tokenizer, a sentence splitter, a part-of-speech tagger, a gazetteer, a named-entity parser, a large hand-tagged corpus and, of course, an English system to bootstrap from.
- Another bootstrapping experiment involved Catalan, which is syntactically and lexically close to Spanish. Developers concluded that it is better to use bootstrapping from a similar language than stochastic methods applied to a small tagged corpus for the target language [10]. Of course, this methodology assumes that there exists another "less low density" language for which named entity recognition capabilities already exist.
- The named entity recognition system developed by Mikheev et al. [18] differs from most other systems in its relatively minimal reliance on gazetteers (static lists of named entities). The pitfalls of relying too heavily on gazetteers are well known and include: a) the impossibility of exhaustively listing all named entities; b) the overlap between, e.g., *Washington* as a place and as a person; and c) the fact that a given proper-name string, like *Adam Kluver*, could be a personal name, part of an organization name, part of a place name, etc. This system uses rule-based grammars, statistical models, a tagged corpus and a small inventory of names to learn to recognize named entities.
- The named entity recognition systems configured by NYU for successive MUCs reveal an interesting historical progression. For the first five MUCs, NYU used full syntactic and semantic analysis – based on a large grammar and lexicon of English – to support a pattern-matching approach; but all of this machinery did not produce the particularly good results. So for MUC-6 [7] they cut back on the processing and concentrated on specifics of the named entity recognition task, utilizing only a gazetteer, various specialized dictionaries, scenario-specific terms (each MUC covered a specific domain), a part-of-speech tagger, and task-specific noun phrase rules. Of course, even having cut back their resource requirements to this degree, the resources involved still surpass those likely to be available for low-density languages.
- SPARSER [11] is a high-quality pattern-matching-based named entity recognition system that relies on both internal and external evidence when analyzing named entities (internal evidence is evidence from within the

[2] The metrics "recall" and "precision" were, by the way, originally invented for the MUC conferences.

sequence of words and characters that comprise the name, whereas external evidence comes from the context adjacent to the name). The required resources include: a lexicalized grammar; a closed-class lexicon (a lexicon of "minor" parts of speech that cannot productively be added to over time); an open-class lexicon (a lexicon of nouns, verbs, adjectives and adverbs that can be added to over time as new phenomena arise in the world); a gazetteer; lists of trigger words; and a "moderately complex control structure that permits a deterministic parse and monotonoic semantic interpretation". SPARSER has been used in two domains: job changing and corporate joint ventures. The development of knowledge resources has focused on these domains.

- Another successful pattern matcher for English is LaSIE [22]. LaSIE is an all-purpose language processor that includes syntactic, semantic and discourse processing, and relies on an ontology to support semantic interpretation. Named entity recognition in LaSIE employs gazetteers, trigger words, a proper name grammar consisting of 177 rules, 100 Sentence Grammar rules from Penn TreeBank-II, and a parse of the whole text that results in a discourse interpretation. The Discourse Interpreter carries out coreference resolution and makes certain inferences about the semantic type of entities.

The common property of all the diverse systems mentioned above is their reliance on a significant amount of pre-prepared data and/or programs – which is natural for systems that target languages that have long been the object of NLP. However, if one needs to develop named-entity recognition capabilities for low-density languages, the cost and feasibility of all prerequisites must be considered.

Here we will explore how to develop systems that are, at base, language independent and can be readily configured to cover any language. Some such approaches already exist but, unlike our approach, they take a stochastic rather than a pattern-matching approach. For example, SRA's RoboTag [1] is a tagging tool and machine learner applicable to any language. Prerequisites for its use are a preprocessor, a morphological analyzer and a lexicon for the given language. Its goal is to allow the end user to build a tagging system for a language by providing examples of what should be tagged rather than requiring the user to learn a pattern language. Another stochastically oriented system that can be applied to any language – and, indeed, was configured to target low-density languages – is the Hopkins named-entity recognition system [6]. Developers sought to "build a maximally language-independent system for both named-entity identification and classification, using minimal information about the source language". Their algorithm begins with seed names for each class, learns contextual patterns that are indicative for those classes, then iteratively learns new class members and word-internal morphological rules. The system can work on both small and large corpora with more or less informant input. One can imagine that this system might productively be combined with the more knowledge-based systems described here such that the stochastic and knowledge-based methods are exploited to their best advantage.

3. Boas II: Proper Name Recognition via Expectation-Driven Knowledge Elicitation

Boas II is an environment that supports the configuration of proper name recognition capabilities for any language. It requires no external resources or processors, and only minimal development time by a speaker of the given language. The methodology, which we call expectation-driven knowledge elicitation, follows that of the original Boas system [12] – [17], [19]. First we will briefly describe Boas, for which Boas II serves as a task-specific supplement.

3.1. The Boas Precedent

Boas is a knowledge-elicitation system that guides linguistically naïve speakers of any alphabetic language (L) through the process of providing machine-tractable information about L. Although Boas was originally intended to support the automatic ramping up of L-to-English machine translation (MT) systems, the elicited knowledge could be used for any application.[3]

The requirements of the system, which oriented its development, were as follows. The knowledge elicited needed to cover ecology/preprocessing (writing system, orthographic conventions, punctuation, etc.), morphology, syntax and lexicon. The system had to be language-independent and applicable to any alphabetic language, with no language-specific adjustments or retrofitting; in other words, all phenomena from all natural languages had to be provided for (to the extent feasible) and the collected information had to be automatically convertible into processing resources. In addition, the system had to be accessible to an untrained user, which meant that the methodological initiative and a large degree of the responsibility for coverage had to rest with the system itself. Since the technological solution to the above requirements had to be practical, the informant's time had to be used efficiently. To enhance the utility of the system in practical applications, the target knowledge elicitation time was set at six months, which could be increased or decreased as resources permitted. The common working language of the interface was English, which not only permitted some degree of English-orientation in the knowledge-elicitation process (e.g., using English seed lexicons to drive lexical acquisition), but also facilitated the preparation of a vast apparatus of training and reference materials, which amount to an on-line introduction to descriptive linguistics. In essence, Boas was designed to act like a field linguist, but whereas a field linguist can describe a language using any expressive means, Boas had to represent the accumulated knowledge in a machine-tractable, structured fashion; and whereas field linguists often focus on idiosyncratic ("linguistically interesting") properties of a language, Boas had to concentrate on the most basic, most widespread phenomena.

The overarching approach to developing Boas was to compile an inventory of cross-linguistically attested parameters, values and realizations that describe languages in general. The parameters represent categories of phenomena that need to be covered in the description of L, the values represent choices that orient what might be included in the description of that phenomenon for L, and the realization options suggest the

[3] Restricting the system to alphabetic languages that have distinct word boundaries was a programmatic decision. This approach to KE could, however, be extended to non-alphabetic languages as well.

kinds of questions that must be asked to gather the relevant information. Users of Boas, therefore, needed to complete what might be described as a "smart" multiple-choice exam: the choices were prepared beforehand but the path through the elicitation process depended on which choices were made. In addition, the option to provide a new value (a new type of realization of any of the phenomena) was always open, since we knew beforehand that it would be practically impossible to achieve complete cross-linguistic adequacy in our recorded realizations. A sample of the parameters, values and realizations used in Boas is shown in Table 1. The first block illustrates inflection, the second closed-class meanings, the third, ecology and the fourth, syntax.

Table 1. Sample parameters, values and means of their realization in Boas.

Class	Parameter	Values	Means of Realization
Inflection	Case Relations	nominative, accusative, dative, instrumental, abessive, etc.	flective morphology, agglutinating morphology, isolating morphology, prepositions, postpositions, etc.
	Number	singular, plural, dual, trial, paucal	flective morphology, agglutinating morphology, isolating morphology, particles, etc.
	Tense	present, past, future, timeless	flective morphology, agglutinating morphology, isolating morphology, etc.
Closed-class meanings	Possession	+/-	case-marking, closed-class affix, word or phrase, word order, etc.
	Spatial Relations	above, below, through, etc.	word, phrase, preposition or postposition, case-marking
Ecology	Expression of Numbers	integers, decimals, percentages, fractions, etc.	numerals in L, digits, punctuation marks (commas, periods, percent signs, etc.) or a lack thereof in various places
	Sentence Boundary	declarative, interrogative, imperative, etc.	period, question mark(s), exclamation point(s), ellipsis, etc.
Syntax	Grammatical Role	subjectness, direct-objecness, indirect-objectness, etc.	case-marking, word order, particles, etc.
	Agreement (for pairs of elements)	+/- person, +/- number, +/- case, etc.	flective, agglutinating or isolating inflectional markers

Boas was implemented as a prototype system and used by developers in a number of applications. It is not a commercial product and is not distributable. However, sufficient details of the system are available in the papers referred to above to permit reimplementation.

3.2. Boas II: Overview and Goals

Like Boas, Boas II is a language-independent system whose resident knowledge includes cross-linguistically attested expectations about linguistic phenomena. In the case of Boas II, the purview is proper names. The system guides users through the process of providing language-specific information about proper names that is then used to automatically configure a proper name recognizer. For reasons associated with the funder's preferences, Boas II places primary emphasis on names of people, but also covers names of companies, institutions, buildings, locations, geographical names and events (e.g., World War II). Like Boas, Boas II was developed as a prototype system and is not distributable. For that reason – and because this article seeks to describe not specific systems but approaches to treating low-density languages – we will not focus on details of implementation but, rather, on the content of and rationale behind the system. Interested readers can find details about our implementation in McShane et al. 2005.

Boas II takes a pattern-matching approach to proper name recognition. An important aspect of the work is compiling inventories of named entity components (e.g., personal names, family names) by means of iterative corpus-based methods. These inventories both support higher-level corpus work and improve the overall functioning of the named entity recognizer. As explained above, compiling inventories does not solve all problems of proper name recognition; however, it does help a great deal. Although Boas II does not employ machine learning, the knowledge acquired during use of this system could support stochastic methods. For example, the Hopkins team (Cucerzan and Yarowsky 1999; cf. above) reports that "F-measure increases roughly logarithmically with the total length of the seed wordlists in the range 40-300", meaning that the larger the available inventories of elements, the better the results. Since Boas II is strong on inventory building, its output could be input to something like the Hopkins' system.

Like the original Boas, Boas II elicits only that information that can be immediately exploited by the system. It does not elicit interesting factoids about name use in different languages that have more sociolinguistic interest than computational linguistic import – at least given the current state of the art. For example, some of the cross-linguistic aspects of proper name usage that we learned from responses to a survey conducted through the Linguist List are:[4]

- Afghans generally do not have a surname, but they do have two personal names, the latter of which is often mistakenly taken to be a surname by Westerners (though a reanalysis of the status of the second name has come about, at least for many Afghans who have contacts with the west);
- Brazilian children have a compound surname consisting of their mother's surname followed by their father's surname; so Elisa Wamierbon Pinchemel is the child of Augusto Pinchemel and Elisa Wamierbon;
- Brazilian children tend to have 2 or 3 personal names and tend to be called by the second or third of them;
- In Serbo-Croatian, following the personal name is a patronymic, but it can be either a special form of the patronymic or the base form of the father's name;

[4] Thanks to the many respondents to this survey, whose observations are cited here and below.

- In Swahili, parents' names change after a child is born: the mother is called Mama-wa <Mama-ya, Mama> + son's personal name, and the father is called Baba-wa <Baba-ya, Baba> + son's Personal name for father.

Some such information could be important if an NLP system attempted to carry out reasoning based on cross-referencing family members: e.g., a system might use a patronymic to link a particular real-world son to his father. However, if a system that advanced were developed for a language, the elicitation of such coreference-based information – which is difficult to render using pre-defined parameters and values – could be carried out independently.

3.3. Creating Pattern Inventory for People's Names

In this section, we describe the subtasks of the Boas II system. We omit certain pedagogical materials used to initiate readers into the goals of the system, since that information has already been provided above. Most of the subtasks can be completed in any order, with the exception of certain self-evident prerequisites: e.g., you cannot create patterns of components until you have selected an inventory of components to participate in those patterns.

3.3.1. Basic inventory of components

In this subtask, the user is presented with a list of category names from which he chooses the ones relevant for L. The inventory and examples, minus the checkboxes used in the interface, are shown in Table 2.

Table 2. Components of people's names.

Category Name	**Example**
Personal	John
Family	Smith
Tribal	Abnaki
Patronymic	Ivanovich
Matronymic	Espinosa
Middle	Ann
Title	Mr., Mrs.
SocialRole	Professor, Dr.
Descriptor	III, Jr.
Particle	von, de
Initial	A.
Comma	(John Smith, DDS)
Article	the, 'la'/'en' (Catalan) (the Duke of Marlborough; the Greens; also used before names in Modern Greek)
Preposition	of (the Duke of Marlborough)
TerritorialDesignation	Marlborough (John, Duke of Marlborough)
TermOfRespect	e.g., 'Mother' in Bahasa Indonesian can be used as a sign of respect, with no kinship implied or the necessity that the addressee be older than the speaker: e.g., Mother Susan, for a woman named Susan
TribalParticle	Al (Al Ghamdi in Arabic)
Caste	Pico Iyer ('Iyer' is the caste name for Saivite Brahmins)
SocialRelation	this category includes, but is not limited to, kinship terms (cf. TermOfRespect above); e.g., Arabic 'abu' "father of"; 'ibn' "son of"; also servant of, etc.

FormerNameIndicator	e.g., Alice Smith nee Johnson
Called	German 'gen.', as in Theo Vennemann gen. Nierfeld
WordForFamily	familia (e.g., la familia [Husband'sFirst Surname] in Castillian Spanish)
HistoricalFamily	In Westphalia, the typical means of calling Theo Schulte might be Winter's Theo, since the old family name or name of the estate was Winter
Pangilan	A shortened name but, unlike nicknames as English speakers understand them, these are (e.g., in Javanese)
Conjunction	e.g., i 'and' in Catalan can be used between surnames (Antoni Badia i Margarit, where Badia and Margarit are Family names)

The interface explains to users that we have tried to use maximally uncontroversial labels for categories of names, but no labeling system is perfect. Users can accept our labels or they can use their own labels. However, if they do the latter, they will not be able to take advantage of the previously prepared syntactic patterns that use our labeling conventions (cf. Section 3.3.2). In other words, there is a practical advantage to accepting our label "personal", but if a user strongly prefers the label "given" or "first", he can use it; he will simply have to create by hand all of the syntactic patterns in which it participates.

3.3.2. *Basic syntactic patterns*

The user is presented with the subset of patterns from our inventory of common syntactic patterns that contain the components selected for L. For example, if Personal, Initial and Family are all selected by the user, then the patterns he will see will include Personal Family (*Robert Jones*), Personal Initial Family (*Robert T. Jones*), Personal (*Robert*), Family (*Jones*). The full inventory of patterns is shown in Table 3. The basic syntactic patterns do not include iteration of elements, information about which is elicited later. Examples are missing in cases for which "native" illustrations were not readily available, although the patterns were attested by speakers of some language.

Table 3. Inventory of syntactic patterns for people's names.

Pattern	Example
Personal Family	Howard Jones
Personal Tribal	~
Personal Caste	Pico Iyer (Tamil)
Family Personal	Li Bai (Chinese)
Initial Family	H. Jones
Initial Tribal	~
Family Initial	Li B.
Personal Initial Family	Howard P. Jones
Personal Initial Tribal	~
Family Personal Initial	~
Personal Middle Family	Howard Paul Jones
Personal Middle Tribal	~
Family Personal Middle	~
Personal Patronymic Family	Ivan Pavlovich Belyj (Russian)
Personal Patronymic Matronymic	found in Spanish[5]

[5] One can also interpret such names as Personal Family Family. In this system, doubled family names are elicited later. The only reason to split patronymics from matronymics is in case they belong to different stored inventories of names.

Initial Initial Family	H. P. Jones
Initial Initial Tribal	~
Family Initial Initial	~
Initial Middle Family	H. Paul Jones
Initial Middle Tribal	~
Family Initial Middle	~
Title Family	Mr. Jones
Title Personal Family	Mr. Howard Jones
Title Initial Family	Mr. H. Jones
Title Personal Initial Family	Mr. Howard H. Jones
Title Personal Middle Family	Mr. Howard Paul Jones
Title Initial Initial Family	Mr. H. P. Jones
Title Initial Middle Family	Mr. H. Paul Jones
SocialRole Family	Mr. Jones
SocialRole Personal Family	Mr. Howard Jones
SocialRole Initial Family	Mr. H. Jones
SocialRole Personal Initial Family	Mr. Howard H. Jones
SocialRole Personal Middle Family	Mr. Howard Paul Jones
SocialRole Initial Initial Family	Mr. H. P. Jones
SocialRole Initial Middle Family	Mr. H. Paul Jones
Personal Family Comma Descriptor	Howard Jones, Jr.
Initial Family Comma Descriptor	H. Jones, Jr.
Personal Initial Family Comma Descriptor	Howard P. Jones, Jr.
Personal Middle Family Comma Descriptor	Howard Paul Jones, Jr.
Initial Initial Family Comma Descriptor	H. P. Jones, Jr.
Initial Middle Family Comma Descriptor	H. Paul Jones, Jr.
Title Personal Family Comma Descriptor	Mr. Howard Jones, Jr.
Title Initial Family Comma Descriptor	Mr. H. Jones, Jr.
Title Personal Initial Family Comma Descriptor	Mr. Howard H. Jones, Jr.
Title Personal Middle Family Comma Descriptor	Mr. Howard Paul Jones, Jr.
Title Initial Initial Family Comma Descriptor	Mr. H. P. Jones, Jr.
Title Initial Middle Family Comma Descriptor	Mr. H. Paul Jones, Jr.
SocialRole Personal Family Comma Descriptor	Mr. Howard Jones, Jr.
SocialRole Initial Family Comma Descriptor	Dr. H. Jones, Jr.
SocialRole Personal Initial Family Comma Descriptor	Dr. Howard H. Jones, Jr.
SocialRole Personal Middle Family Comma Descriptor	Dr. Howard Paul Jones, Jr.
SocialRole Initial Initial Family Comma Descriptor	Dr. H. P. Jones, Jr.
SocialRole Initial Middle Family Comma Descriptor	Dr. H. Paul Jones, Jr.
Personal Family Descriptor	Howard Jones Jr.
Initial Family Descriptor	H. Jones Jr.
Personal Initial Family Descriptor	Howard P. Jones Jr.
Personal Middle Family Descriptor	Howard Paul Jones Jr.
Initial Initial Family Descriptor	H. P. Jones Jr.
Initial Middle Family Descriptor	H. Paul Jones Jr.
Title Family Descriptor	Mr. Jones Jr.
Title Personal Family Descriptor	Mr. Howard Jones Jr.
Title Initial Family Descriptor	Mr. H. Jones Jr.
Title Personal Initial Family Descriptor	Mr. Howard H. Jones Jr.
Title Personal Middle Family Descriptor	Mr. Howard Paul Jones Jr.
Title Initial Initial Family Descriptor	Mr. H. P. Jones Jr.
Title Initial Middle Family Descriptor	Mr. H. Paul Jones Jr.
SocialRole Personal Family Descriptor	Dr. Howard Jones Jr.
SocialRole Initial Family Descriptor	Dr. H. Jones Jr.
SocialRole Personal Initial Family Descriptor	Dr. Howard H. Jones Jr.
SocialRole Personal Middle Family Descriptor	Dr. Howard Paul Jones Jr.
SocialRole Initial Initial Family Descriptor	Dr. H. P. Jones Jr.
SocialRole Initial Middle Family Descriptor	Dr. H. Paul Jones Jr.
Title Personal Patronymic Family	Gospodin Ivan Pavlovich Belyj (Russian)

SocialRole Personal Patronymic Family	Profesor Ivan Pavlovich Belyj (Russian)
Article Family	The Greens; la Badia (Catalan – fem. sg.)
Article Personal Family	la Antoni Badia (Catalan)
Personal Comma SocialRole Preposition Family	John, Duke of Marlborough
TribalParticle Tribal	Al Ghamdi (Arabic)
SocialRole Personal Family Preposition TerritorialDesignation	Lord Stewart Sutherland of Houndwood
SocialRole Family Preposition TerritorialDesignation	Lord Sutherland of Houndwood
Personal Comma SocialRole Preposition TerritorialDesignation	John, Duke of Marlborough
Personal Comma SocialRole Family Preposition TerritorialDesignation	Stewart, Lord Sutherland of Houndwood
TerritorialDesignation Initial Personal	Vilayanur S. Ramachandran (Tamil)
TerritorialDesignation Personal Personal	Attipat Krishnaswami Ramanujan (Tamil)
Personal SocialRelation Personal	Ali b. Abu-Talib (Ali son of Abu-Talib); A'ishah b. Abu-Bakr (A'ishah daughter of Abu-Bakr) (Arabic)
SocialRelation Personal	Umm Habibah (mother of Habibah)
Personal Family FormerNameIndicator Family	Jane Smith nee Johnson
Personal Initial Family FormerNameIndicator Family	Jane R. Smith nee Johnson
Personal Middle Family FormerNameIndicator Family	Jane Ruth Smith nee Johnson
Initial Middle Family FormerNameIndicator Family	J. Ruth Smith nee Johnson
Personal Family Called Family	Theo Vennemann gen. Nierfeld (German)
Title Personal	Dr. Abdullah (Arabic)
Personal SocialRole Particle Family Called Family	Bruno Baron von Freytag gen. Löringhoff (German)
Article Title Preposition Surname	la Señora de [Husband's First Surname] (Spanish)
Article Family WordForFamily	The Cook Family
Article WordForFamily Family	la familia [Husband'sFirst Surname] (Spanish)
HistoricalFamily Personal	Winter's Theo (German in Westphalia)
Title Pangilan	(Javanese)
SocialRelation Pangilan	(Javanese)
Personal Family Conjunction Family	Antoni Badia i Margarit (Catalan)
Article Personal	la Antoni (Catalan)
Article Personal Family Conjunction Family	la Antoni Badia i Margarit (Catalan; less common)
Article Family Conjunction Family	la Badia i Margarit (Catalan)
Personal Patronymic	Ivan Pavlovich (Russian)
Patronymic	Pavlovich (Russian; colloq.)
TermOfRespect Personal	Mother Susan (Bahasa Indonesia)
Personal	Mary
Personal Middle	Mary Elizabeth

3.3.3. *Additional Components and Syntactic Patterns*

The user is then presented with his or her current inventory of personal name components and permitted to supplement it, if necessary, either with completely new elements or by preferred names for elements that were included in the initial inventory. If any new elements are provided, the current inventory of syntactic patterns is displayed to the user and he is asked to supplement it to account for the new elements.

3.3.4. *Iteration of components*

Iteration of name components is a common phenomenon: e.g., German permits multiple titles, as in *Dr. Dr. Mueller*; personal names in French (not to mention English) often contain two elements: *Jean Claude, Mary Beth*; multiple descriptors are used in many languages, as in *John Smith, MD, PhD*. In this task, the user is asked to indicate which name components can iterate, how many times (2-4 are elicited directly; if more, the user must manually enter the relevant patterns), and what punctuation can intervene between iterated components (e.g., dash, space).

3.3.5. *Inventories of titles, professions, etc.*

The user is asked to translate whichever elements from our resident list of titles and professions might be useful for proper noun recognition in L. For convenience, we loosely group the list according to the categories Titles, General Professional, Military, Royalty, Political, Business, Medical, Academic, Entertainment/Communication, Family Role, Legal Role and Other. We do not provide these lists in full for reasons of space.

3.3.6. *Compiling Inventories of components*

The user is asked to provide a seed inventory of examples for each type of name component: e.g., if he were configuring a system for English, he might add the following components to the categories "personal" and "family":

Personal: Ann, Mary, Susan, Keith, Albert, …
Family: Jones, Smith, Harris, McDuff, …

These seed lists, along with the seed list of titles described in Section 3.3.5, will be used as heuristics during later corpus work. If inventories of such elements are available externally, they can be imported using Boas II's import function.

In the current implementation, a word that explicitly belongs to one category will be blocked from matching another category: e.g., if *Mr.* is listed as a Title, it will not match the Family name slot in a pattern. If a given entity can belong to more than one category – e.g., *Washington* can be a TerritorialDesignation, a Personal name or a Family name – it must be listed explicitly in each category. Although the decision to permit strings to belong to only one category unless otherwise indicated can lead to some missed matches (e.g., if *Washington* were listed as a Family name, the string *Washington Erving* would not be properly analyzed), we have found it a more practical solution than permitting the extensive false positives encountered when not using such a filter.

The processing of eliciting inventories of category members also includes the option of compiling a stop list: that is, entities that should not be matched in any corpus searches. This stop list will be very helpful for people's names, since one would not expect words like *house* or *dog*, even if capitalized, to be part of a person's name. However, if a user will be covering, for example, company names, he or she must be careful not to overpopulate the stop list: e.g., if *dog* is in the stop list, the system will not find *Happy Dog Dog Food Company*. Eliciting different stop lists for different types of named entities is not part of this version of the system, though it would be a useful future enhancement.

3.3.7. Nicknames

The user is given the option of providing nickname equivalents for the current inventory of Personal names, since such correspondences can be important for systems that carry out coreference resolution for named entities. When searching the corpus for named entities, the system interprets nicknames the same as full personal names, and includes nicknames as entries in the list of personal names. The correspondences between full names and nicknames is, however, stored should that information be useful for an application.

3.3.8. Punctuation

An inventory of punctuation marks that occur outside of personal names is elicited (those that occur inside of names were elicited earlier and incorporated into the syntactic patterns). This inventory is used for parsing corpora: for example, if a period is a name-external punctuation mark, then the text string *Ann. Bill* will be understood as containing two different names, not a single proper name. Recall that Boas II does not require any prerequisite technologies – it answers for all its own prerequisites; therefore, the understanding of punctuation, which would be an aspect of preprocessing in an end application, must be handled explicitly.

3.3.9. Morphological forms

The user is asked if components of names can occur in non-base morphological forms, like the plural or in a case that is different from that of the citation form. The answer to this question will alert developers to the need to incorporate external morphological analysis, as could be carried out, for example, by the type of analyzer automatically generated by the original Boas system. Morphological analysis is particularly important for flective languages, like Russian, for which a given proper name can have a dozen inflectional forms.

3.3.10. Capitalization

The user is asked if capitalization can aid in the detection of proper names. Note, however, that even if capitalization is generally a strong heuristic for a proper names, like it is in English, capitalization conventions are often not followed in informal genres, like email and blogs. Therefore, even if capitalization is a heuristic in a language, the user is asked before each run of the proper name recognizer whether or not he wants capitalization to be considered as a heuristic.

3.3.11. Heuristics for components

The user is asked to provide prefixes and/or suffixes that suggest that a given string represents a certain type of personal name component. For example, the suffix *-ovich* in

Russian strongly suggests a patronymic. As with capitalization, this heuristic can either be used or ignored during any given run of the proper name recognizer.

3.4. Treating Proper Names Not Referring to People

In this section, the user provides pattern-matching knowledge for proper names that refer to entities other than people: bodies of water, buildings, geological entities, publications, company names and proper-noun events (e.g., *World War II; the Sydney Olympics*). The knowledge elicitation process is the same for all these subtypes of entities. We will use bodies of water for illustration.

The user is presented with a list of types of bodies of water in English and asked (a) to provide translations, as applicable, into L; (b) to add L variants of any other keywords (like *lake* and *river*) that indicate bodies of water; (c) to select from among three syntactic patterns in which "body of water keywords" can participate:

1. word* + body of water keyword (e.g., *Amazon River*)
2. body of water keyword + word* (e.g., *Lake Michigan*)
3. body of water keyword + preposition/postposition/particle + word* (*Gulf of Mexico*)

Word* indicates one or more proper-noun words. If pattern 3 is selected, then the applicable prepositions, postpositions or particles are elicited. The elicitation of these patterns is not as fine-grained as for people's names, which, as mentioned earlier, reflects the stated goals of the funder. Enhancements to this thread of knowledge elicitation would include providing more patterns to select from, especially for company names since, as [6] shows, the variety of patterns for company names is rich indeed. In Boas II, knowledge about proper names that do not refer to people is used primarily to block false positives when searching for people's names.

3.5. Corpus Work

Using just the knowledge requested above, which might take a user between 30 minutes and two hours to provide, Boas II can automatically configure a proper name recognition system. However, the quality will probably not be very good from the outset because listing is not easy: if one is asked to list 50 kinds of dogs, he might get stuck at 12, whereas if he sees a list of words, he can easily pick out which ones are dogs. The same is true of patterns of proper names: it is likely that the user will have forgotten some the first time round, and it is likely that the inclusion of certain stop words into the stop words list will significantly improve results. For this reason, we use iterative corpus-based methods to help users to improve the system.

3.5.1. Upload/select corpus

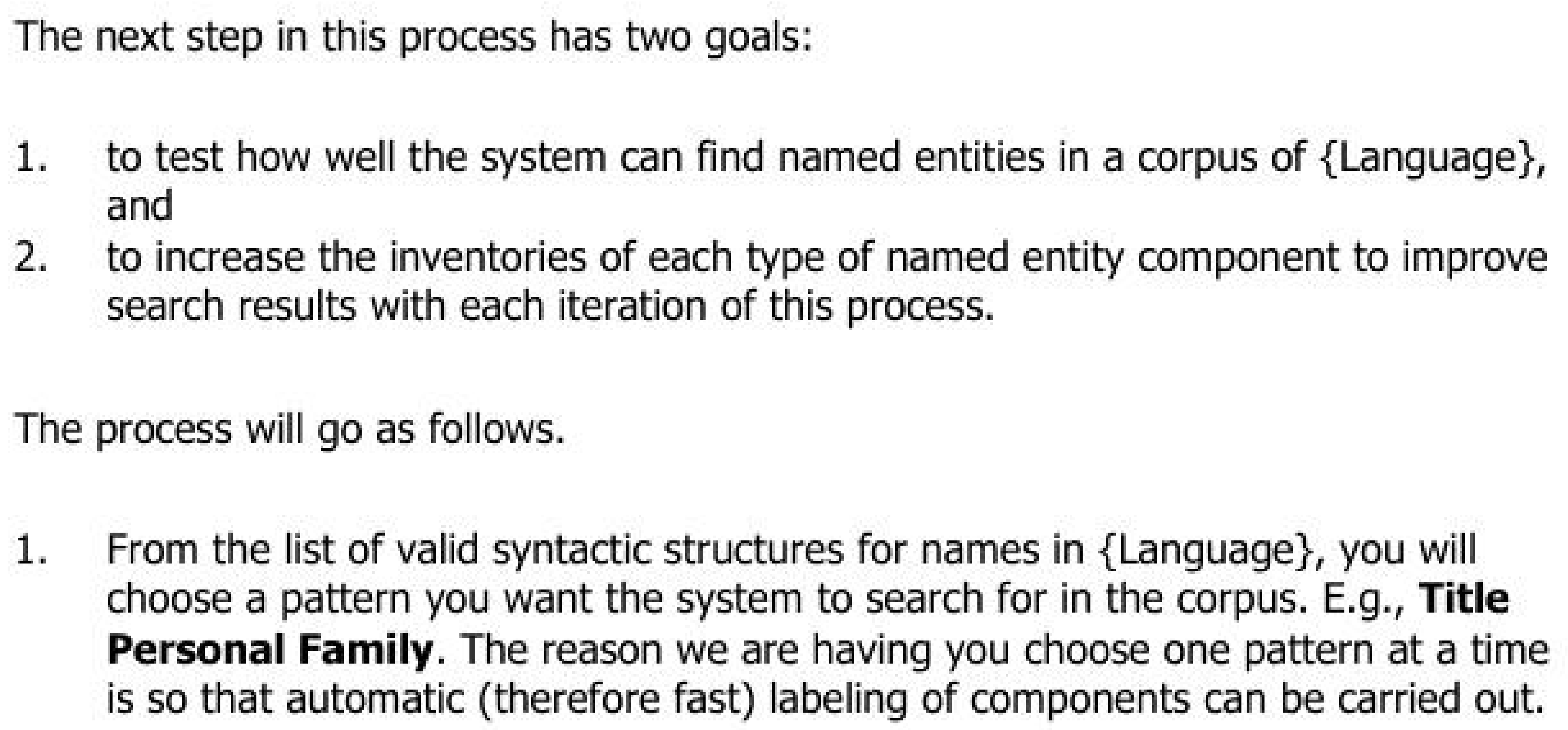
The user uploads one or more corpora, following the instructions to convert them into UTF-8 encoding, then selects one to work with. This assumes, of course, that some electronic text is available, thus justifying the configuration of a proper name recognition tool to begin with.

3.5.2. Preparing to search

The user is asked to select one syntactic pattern at a time from the inventory he created earlier. This pattern is searched for in the selected corpus, with matches being returned for the user's approval or rejection. When the user accepts a candidate, all components of it that are not already part of the respective inventory are recorded. For example, if the user agrees that *Polly Jones* represents the pattern Personal Family, and if *Polly* is not yet in the list of Personal names for English (but *Jones* is in the list of Family names), *Polly* will automatically be added to the inventory of personal names, and Polly Jones will be added to the inventory of complex known entities – another part of the growing knowledge base of Boas II.

Patterns are searched for individually precisely in order to permit components of approved candidates to be automatically added to the respective inventories. This would not be possible if one searched simultaneously for different patterns, like Title Surname and Personal Surname, since the system would not know if the first string of an approved entity were a Title or a Personal name. (Another implementation option would have been to permit searching for multiple patterns at one time; however, the necessity of individually labeling each element of approved candidates would have been, we hypothesized, too time-consuming. Yet another, more expensive, implementation option would have been to permit both search options, with each option being employed by users as they chose.) The inventory of patterns presented includes any necessary expansions based on component iteration. So, if a user indicated that Personal Patronymic Family was a valid pattern, and also indicated that Family names could be iterated twice with either a hyphen or a white space between them, the inventory of patterns would be expanded to include Personal Patronymic Family-Family and Personal Patronymic Family Family.

Since selecting good search strategies is important for making Boas II robust with a minimum of user effort, we carefully explain the ramifications of various search strategies to users. As an example of the pedagogical aspect of Boas II, we provide this explanation in full.



The next step in this process has two goals:

1. to test how well the system can find named entities in a corpus of {Language}, and
2. to increase the inventories of each type of named entity component to improve search results with each iteration of this process.

The process will go as follows.

1. From the list of valid syntactic structures for names in {Language}, you will choose a pattern you want the system to search for in the corpus. E.g., **Title Personal Family**. The reason we are having you choose one pattern at a time is so that automatic (therefore fast) labeling of components can be carried out.

2. The system will carry out the search and present you with candidate names, e.g.,

Mr.	Tom	Smith
Ms.	Judy	Garland

3. You will accept or reject each as a valid representative of **this pattern**. Any candidate that you accept will automatically have its component parts labeled according to the original search pattern, and those components will be automatically added to the relevant name component list. So, if you are searching for "Title + Family" and the system returns "Mrs. Mary" (which is actually Title + Personal), you should reject that candidate; otherwise, the name "Mary" would be incorrectly added to the list of valid Family names for English.
4. Then you will launch another search, choosing a different pattern to search for: e.g., **Personal Patronymic Family**. You will keep repeating this process until the system is finding most of the relevant syntactic patterns for names and the system's inventory of elements belonging to each name components is quite large, or until you run out of time.

At any time you can return to the pages that elicit components of named entities or patterns using them, should you find that some components or patterns are missing. You can also manually add to the inventories of components (like Family names) or to the stop list at any time.

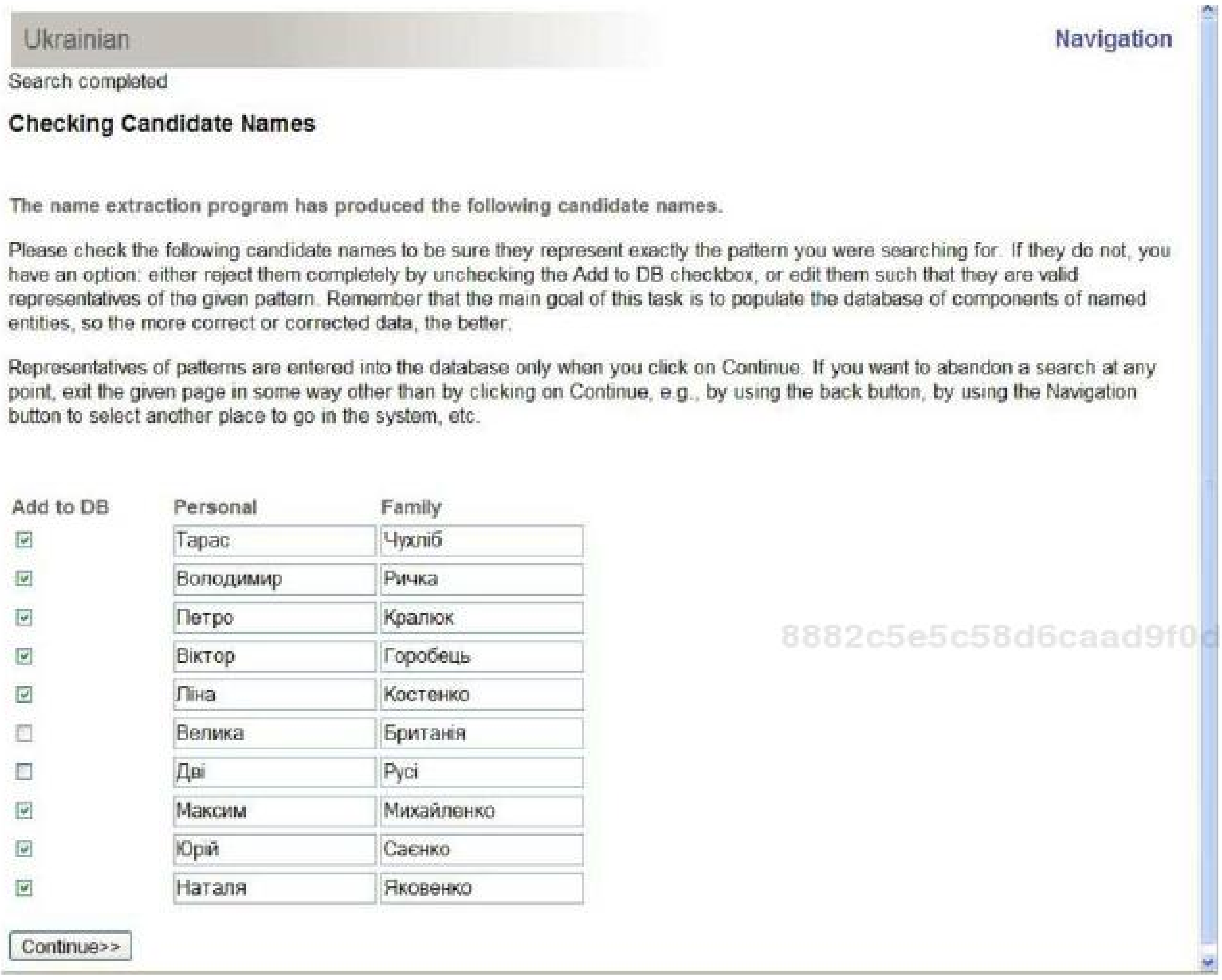
The most important aspect of this process is **figuring out a good strategy** for searching – we'll try to help. The worst case would be to start out searching for, say, a Family name used alone because if {Language} uses capitalization like English does, every single capitalized word – including the first word of every sentence – will be selected as a candidate; and if {Language} doesn't use capitalization, every single word in general will be selected (after all, how can the system know something is *not* a Family name?). A better strategy is to start from the most restricted types of patterns, like those that use titles or contain many components.

3.5.3. Launch search

To launch a search, the user selects one of the syntactic patterns he has already established for L and indicates whether he wants any heuristics to be attached to any of the components. For example, if the language is Ukrainian, and if the user correctly indicated that the pattern "Personal Family" was permitted. He can then say:

1. whether each component must be capitalized (yes)
2. whether only known instances of components should be sought (this depends upon his goal; if, for example, he wants to extend the inventory of known Family names without finding too many false positives, he might want to accept only 'known' Personal names)
3. whether affixation heuristics should be used (probably not, but possibly yes, depending on how large the corpus is and the user's particular goals).

Figure 1 shows the results of one search in Ukrainian; the actual text is less important than the look and feel of the process.



Search completed

Checking Candidate Names

The name extraction program has produced the following candidate names.

Please check the following candidate names to be sure they represent exactly the pattern you were searching for. If they do not, you have an option: either reject them completely by unchecking the Add to DB checkbox, or edit them such that they are valid representatives of the given pattern. Remember that the main goal of this task is to populate the database of components of named entities, so the more correct or corrected data, the better.

Representatives of patterns are entered into the database only when you click on Continue. If you want to abandon a search at any point, exit the given page in some way other than by clicking on Continue, e.g., by using the back button, by using the Navigation button to select another place to go in the system, etc.

Add to DB	Personal	Family
☑	Тарас	Чухліб
☑	Володимир	Ричка
☑	Петро	Кралюк
☑	Віктор	Горобець
☑	Ліна	Костенко
□	Велика	Британія
□	Дві	Русі
☑	Максим	Михайленко
☑	Юрій	Саєнко
☑	Наталя	Яковенко

Figure 1. First page of results of a search for the pattern Personal Family in Ukrainian.

The user can review the results, unchecking any that do not represent the pattern in question. For orientation, the first of the rejected entities refers to *Great Britain*.

3.6. Informant Time Needed

Developing proper name recognition capabilities for a language using Boas II is intended to take from several hours to several days, depending on the desired quality and coverage of the system as well as the named-entity identification heuristics of the given language We expect the lower boundary of useful elicitation time to be about two hours, during which time the informant would indicate basic components of proper names, the syntactic patterns in which they participate, and some detection heuristics (e.g., capitalization, morphological triggers); he would also build seed lists of components. In this very fast ramp-up scenario, only very small inventories of components would be compiled.

The amount of informant time necessary to ramp up a system of a given quality will depend, among other things, upon the following.

- **Language typology.** Languages with few proper name heuristics – like those that lack capitalization or that use capitalization for every noun (like German) – will require large inventories of name components before good results are achieved. We cannot suggest any magic bullets for proper name recognition in such languages, as we do not believe any such exist. By contrast, languages in which titles commonly introduce names, or in which morphological forms strongly suggest certain types of components (e.g., a given suffix is only used in patronymics), will permit better results with smaller inventories of components.

- **The size and coverage of inventories built by the informant.** The inventories of named entity components act as both positive and negative heuristics for entity recognition. As such, the bigger the inventory, the better the results.
- **What resources are already available.** As part of the Boas II project, we compiled inventories of personal and family names for many languages, which can be exploited by informants for those languages. More such lists might be available on the Web or in other machine-readable resources. In fact, even print resources – like phone books – could be helpful for creating lists, despite the time needed to scan or even type in the entities. In addition, stop lists – like the words in a basic dictionary – are very useful. Any on-line lexicon that can be formatted into a list of head words can be imported into Boas II and used as a blocking heuristic (e.g., capitalized *And* at the beginning of a sentence is certainly not part of a proper name in English).
- **The breadth of the system.** Any subset of named entities can be the focus of a system built with Boas II: e.g., one could build only a person identifier, in which all elicitation tasks not related to people could be skipped.
- **How easily one can find or build a corpus.** Corpus-based elicitation methods are used to drive the compilation of inventories of named entity components as well as to help informants to recall patterns that might not have come to mind in the initial elicitation of patterns; however, the corpus must be built outside of the system and uploaded.

3.7. Recap of Boas II

Let us reconsider the status and potential contribution of a system like Boas II. Such a system can be put in front of a speaker of any alphabetic language, who will be lead through a process of knowledge elicitation for which the system itself takes much of the initiative for what work is carried out and how it is carried out. When a relatively small amount of information has been provided about a language, the system automatically configures a proper name recognition system, which can then be improved through iterative corpus-oriented trial and error. The important point for NLP developers involved with low- and middle-density languages is that the same system can be used for any language, and no external resources (apart from a corpus) or external processors are required.

4. GeoMatch: Multilingual Processing of Place Names

As we have just seen, lists of proper names are a very useful resource for the task of proper name recognition. While such lists can never be expected to contain *all* proper names, they can provide a simple and accurate means of detecting and categorizing many proper names.

If we extend proper name processing to cross-linguistic applications, it would be very useful to have a cross-indexed multi-lingual, multi-script database of proper names. That is, is would be useful to be able to look up how the capital of Russia is rendered in every language using both that language's native script and any other scripts that one might expect to encounter. The system we describe in this section, called GeoMatch, takes a step toward building just such a resource. It is not yet a knowledge elicitation system like Boas or Boas II, but the approaches developed for the initial subset of languages could easily be applied to other languages using Boas-like methodologies. Like Boas and Boas II, GeoMatch was implemented as a proof-of-concept system. Unlike the other systems, it builds upon an existing knowledge resource called the Geographic Names Database (GNDB).

The GNDB contains approximately 5.5 million geographic names and 4.0 million features, covering all countries of the world. For the most part, the names in GNDB are in the native language of the country where they are located: for example, names in Russia are in Russian. However, **all names are rendered as Latin strings** – they are not in the original script of the given language. So, for example, the Russian city known in English as Krasnodar will be listed as *Krasnodar*, which is the transliteration of the native Cyrillic *Краснодар*; and the Russian city known in English as Moscow will be listed as *Moskva*, which is the transliteration of the Cyrillic *Москва*. These examples highlight a noteworthy point about the cross-lingual rendering of names: in some cases, as with Krasnodar, proper names are *transliterated* between languages using rule-based correspondences (which may or may not be univocal), whereas in other cases, as with Moscow, they are *translated* idiosyncratically. We will continue differentiate between *transliteration* and *translation* based on whether the process is productive and rule-driven or idiosyncratic.

The GNDB was developed over decades, largely manually, and before the time when non-English language support in computers was readily available. As such, there are no companion databases that contain the "original" native script versions of entities. GNDB is currently available both as a search function over the Web and as text files for those wishing to incorporate its contents into computer systems (http://gnswww.nga.mil/geonames/GNS/index.jsp).

The specific goals of the GeoMatch project were to make the GNDB more useful as a resource (a) for people searching it over the Internet, and (b) for multi-lingual text processing applications. We were asked to approach this project using knowledge-based rather than stochastic methods because the results of prior attempts to use fuzzy-matching to improve searches for Arabic place names were deemed to be of insufficient quality.

The exclusive use of Latin script in the database leads to certain deficiencies, whether the GNDB is used in the Internet search application or is incorporated into an NLP system. For example, there are often many ways to transliterate a non-Latin string into Latin script and only one of them is recorded in the GNDB, meaning that if a person or system uses another, the entity will not be found. For example, in Russian, when the vowel *a* follows a palatalized consonant it is written as *я*, which has three canonical transliterations into English – *ya* (popular), *ja* (scholarly), *ia* (Library of Congress); GNDB uses only the first. Moreover, as discussed above, names are not always transliterated between languages, they can also be translated, in which case the cross-linguistic forms can be very different. So, ideally, a repository of geographical names would include the rendering of each geographical name in each language and script, and robust transliteration engines would permit users to search for strings using various search strategies.

Covering the geographical entities in every country using every language and script was out of scope for this exploratory project, so three languages and their respective country databases were selected: Russian/Russia, Ukrainian/Ukraine, Polish/Poland. As will be shown, the algorithms developed for these can be applied directly to more languages and country databases, providing the potential to turn the already information-packed GNDB into an even more robust resource to support NLP and geographic research needs.

4.1. The Seed Database

The data in the GNDB cover all countries of the world and are freely available as text files, divided by country, with no licensing requirements or restrictions. The data include both linguistic and extra-linguistic information, totaling 25 fields.[6] The fields of particular interest for language processing are:

UFI unique feature identifier – uniquely identifies the entity using a six-digit number

UNI unique name identifier – uniquely identifies a given string (which can be used to identify more than one place) using a six-digit number

FC feature classification – identifies the general type of the entity (e.g., populated place, vegetation, hydrographic) using one of nine (multi-)letter codes

DSG feature designation code – identifies the specific type of the entity (e.g., populated place, river, canal) using over 500 multi-letter codes; many DSGs can be realized in language as keywords, with the DSG-keyword associations being central to our linguistic processing[7]

LC language code – indicates the language of the given database entry; the field is rarely filled in even if the given entity is not in the native language of the country (an instance of a knowledge gap in the resource); most database entities, however, are in the language of the given country

Generic indicates the keyword, if any, in the string; the field is rarely filled in, even for entities that contain a keyword (another knowledge gap)

Full_Name the full name of the entity, written in Latin script and including diacritics, if applicable

Full_Name_ND the full name of the entity, written in Latin without diacritics, typed using QWERTY, the visible English keyboard

The combination of UFI (the actual place) and UNI (the string that renders it) uniquely identifies each database entry.

[6] The full inventory of DSG codes is available at http://gnswww.nga.mil/geonames/Desig_Code/Desig_Code_Help.jsp

[7] In some cases, we have found more than one keyword corresponding to a given DSG; in others, we have not yet found any keywords for a given DSG. Our inventory of keywords reflects analysis of the data in the GNDB as well as dictionary searches.

The GeoMatch work comprises two parts: enhancing the content of the databases and improving the search capabilities. Each is discussed in turn.

4.2. Database Enhancement

The process of enhancing the databases for Russia, Ukraine and Poland is shown in Figure 1, each step of which is briefly described below.

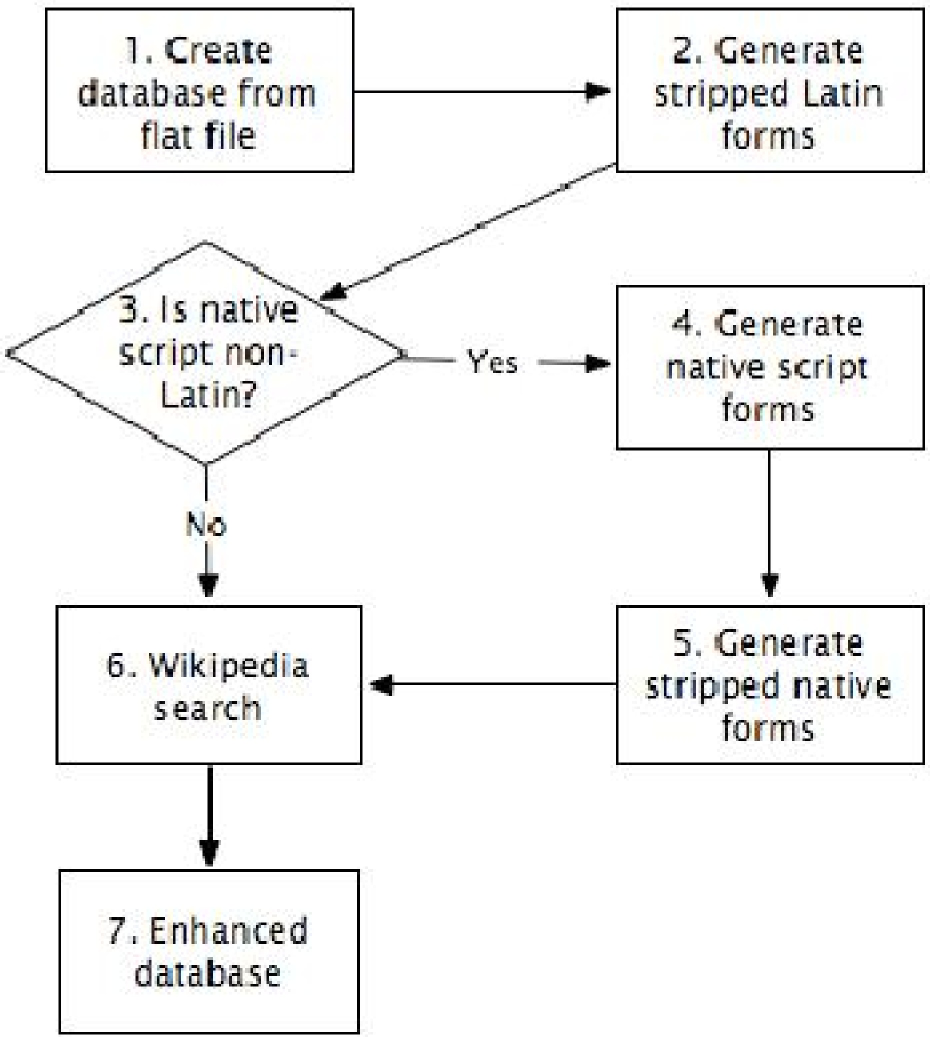

1. Create database from flat file
2. Generate stripped Latin forms
3. Is native script non-Latin?
Yes
4. Generate native script forms
No
5. Generate stripped native forms
6. Wikipedia search
7. Enhanced database

Figure 2. The algorithm for enhancing the GNDB.

Step 1. Although the GNDB exists as a database for use by the Internet search application, the database itself is not publicly distributed. Instead, flat files containing its data are distributed. The first step, therefore, is to create a database for each country from the corresponding flat file that can be downloaded from the Internet.

Step 2. This step is devoted to keyword recognition. We define keywords as strings like *mountain* and *river* that are the part of a proper name that identifies the type of entity: e.g., *Mississippi River* is a river. We call proper names stripped of their keywords "stripped" forms.

It is important to recognize keywords, and to recognize the stripped forms of proper names, for the following reasons:

- Entities that can include keywords sometimes appear in the GNDB with their keyword and sometimes without their keyword: for example, what we know in English as *Lake Baikal* might have appeared as *Ozero Baikal* (where 'ozero' means 'lake') or just as Baikal. Whereas the lack of an overt keyword might seem like an oversight, it actually might not have been because the acquirers of the GNDB unfailingly indicated the nature of the entity by selecting the relevant feature designation code, like LK for lake. So entering Baikal [DSG: LK] says, unambiguously, that this is a lake; entering it as Ozero Baikal is perfectly fine, but not strictly necessary. However, if a person or NLP system wanted to find namely the string *Ozero Baikal*, Baikal would not be a match.

- If an entity is recorded in GNDB with a keyword, sometimes the keyword is in the native language (e.g., Russian for the Russia database), and sometimes it is in English, German or some other language. For example, the database might contain *Lake Baikal* rather than *Ozero Baikal*. In this case, the keyword must be understood as being separate from the actual "proper name" part of the string.
- When a user is searching the database, he can either include or not include a keyword or DSG code. That is, he might type in *Baikal* and indicate that the DSG code is LK; or he might type in *Lake Baikal* and expect the system to understand that he wants only lakes named Baikal. No matter how the user types in the entity and how the entity happened to be entered into the GNDB originally, we want the desired correspondences to be found.

In order to support fast keyword processing at runtime, we supplemented the GNDB by what we call Stripped_Latin forms, which correspond to the Full_Form and Full_Form_ND fields except without the keywords. Whenever a search string is entered, it is parsed and only the proper name parts, minus the keywords, are searched for in the "stripped" columns of the database.

Automatically identifying keywords is not trivial: e.g., 'lake' in *Lake Louise* is not a keyword (this is a city) but 'lake' in *Lake Baikal* is (this is a lake). Our keyword stripping algorithm relies on two types of information to determine whether what looks like a keyword is actually functioning as one: the feature designation (DSG) of the entity and the inventory of keywords and their corresponding DSGs that we compiled into a new Keywords-database.

The Keywords-database covers several hundred keywords in four languages (Russian, Ukrainian, Polish and English) and includes all possible variants of each language's keywords, like every reasonable transliteration of Russian and Ukrainian keywords into Latin script. It also includes a smattering of keywords from other languages, like German and Czech, that were used occasionally in the databases. Creating this inventory was labor-intensive because, as mentioned above, the Generic field is most often empty in the original data; and even when a keyword string is identified, all of its "meaningful" DSG correspondences must be detected: e.g., the keyword 'sea' might be used as a keyword relating to entities of type BAY and LK (lake), but certainly not entities of type PPL (populated place).

The keyword stripping algorithm is as follows:

1. If the entity is composed of only 1 word, there is no keyword.
2. If the entity is composed of >1 word, search the Keywords-database for any of the component words, considering any matches potential keywords. Keywords in all languages are considered since English and German keywords are common in all files, Russian keywords are common in the Ukraine file, etc.
3. If there is > 1 potential keyword
    And if one is at the beginning of the string and the other is at the
    end of the string,
    Then the first one is the candidate keyword.
    Else the last one is the candidate keyword.
4. If the any of the DSGs of the candidate keyword, as recorded in the Keywords-database, matches the DSG of the entity in question, the candidate

keyword is considered an actual keyword and is stripped; else nothing is stripped.

This algorithm has been shown to work robustly for the languages in question. If this approach is expanded to other languages, two modifications might be necessary: the number of languages in which potential keywords are sought (step 2) should be restricted to avoid false positives (e.g., one might not want to look for Chinese keywords in the Russian database), and expectations regarding the linear ordering of keywords in multiple-candidate scenarios (step 3) must be parameterized.

The only errors so far in the stripping process have been due to the failure to include some keyword in the Keywords-database or the omission of a necessary DSG association for a keyword (e.g., the keyword 'river' is used for entities described as STM (stream), not only RV (river)).

Steps 3-4. The next step was to generate exactly one Cyrillic rendering of each Latin string in the Russia and Ukraine databases because Russian and Ukrainian are written in Cyrillic, making the Cyrillic forms the true native forms. In the general case, it would have been impossible to do this because there are several ways that certain Cyrillic characters can be rendered into Latin and, when converting the Latin back into Cyrillic, some ambiguities arise. However, the nature of the original GNDB data helps significantly because a single transliteration convention was largely used throughout (despite some inconsistencies that must be expected in a resource of this size). Therefore, we created a transliteration engine that take the GNDB Latin forms and posits a single Cyrillic variant for each one. Despite a few errors associated with unforeseen letter combinations, which could fixed globally, the results were quite good. There were, however, some residual ambiguities that would have had to have been fixed manually: e.g., the Latin apostrophe can indicate either an apostrophe or a soft sign in Ukrainian, with context providing little power of disambiguation. We did not carry out such manual correction.

All transliteration in this system is carried out using the same engine, whether the transliteration is used for database population or for processing search strings in the ways discussed below. The input to the transliteration engine is a table of correspondences between letters and letter combinations in a source language and a target language. There are no language specific rules and no contextual rules, apart from the ability to indicate the beginning of a string and the end of a string. In addition, longer input strings are selected over shorter strings. As in Boas and Boas II, we intentionally kept the implementation language-neutral so that the component resources could be applied to any alphabetic language.

An example of a row in the English-to-Russian table is as follows. This says that the Latin letters yy at the end of a word ($) are to be rendered as Cyrillic ый.

yy$	ый

For the task of populating the Russia and Ukraine databases with a single Cyrillic variant for each entity, we used special one-to-one transliteration tables, whereas for the search application described below we used one-to-many tables. (Details on this are below.)

Prior to transliterating the entities in the Full_Form field into Cyrillic, we stripped any non-native keywords from them since, e.g., an English or a German keyword used in a supposedly Russian string should not be transliterated.

Step 5. The same as Step 2 except that Cyrillic keywords are stripped rather than Latin ones.

Step 6. The objective of this task is to mine Wikipedia (a) to attest our posited Cyrillic variants of geographical names and (b) to extract multilingual variants of found names. We save extracted multi-lingual variants to a Wikipedia Database that was cross-indexed with our linguistically embellished GNDB using the entity's unique combination of UFI and UNI values.

Our search and extraction engine mimics the search function in Wikipedia, leveraging the fact that the Web address for each entry is predictable based on the head entry (i.e., the head word or phrase for the entry). Each head entry is stored on the page using a strict naming convention: e.g., Krasnodar in English, Spanish and Russian is found at:

http://**en**.wikipedia.org/wiki/**Krasnodar**
http://**es**.wikipedia.org/wiki/**Krasnodar**
http://**ru**.wikipedia.org/wiki/**Краснодар**

The links to related pages in other languages are encoded in a highly structured manner, making them readily detected automatically. The links to the Spanish and Russian pages for Krasnodar from the English page are:

```
<li class="interwiki-es">
<a href="http://es.wikipedia.org/wiki/Krasnodar">Español</a></li>

<li class="interwiki-ru">
<a href="http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B0
%D1%81%D0%BD%D0%BE%D0%B4%D0%B0%D1%80">
Русский</a></li>
```

Since the Russian string for Krasnodar requires non-ASCII characters, it is encoded using percent-escape notation, in which each character is represented by a pair of percent-escapes (e.g., Cyrillic *a* is represented as %D0%B0). Percent-escape notation permits UTF-8 characters to appear in Web addresses (see [8] for a concise overview and [2] for a more in-depth treatment).

Our engine creates a list of Web addresses to search for from our inventory of geographical entities: if a Web page for the given address exists, then the engine follows the links to corresponding pages in other languages, opening up each page and searching for the "meta" tag with the first parameter-value pair *name = "keywords"*. The first value for the parameter *content* within that same tag is always the headword as rendered in the given language. In the underlying html for the Spanish of our Krasnodar example, the tag looks as follows:

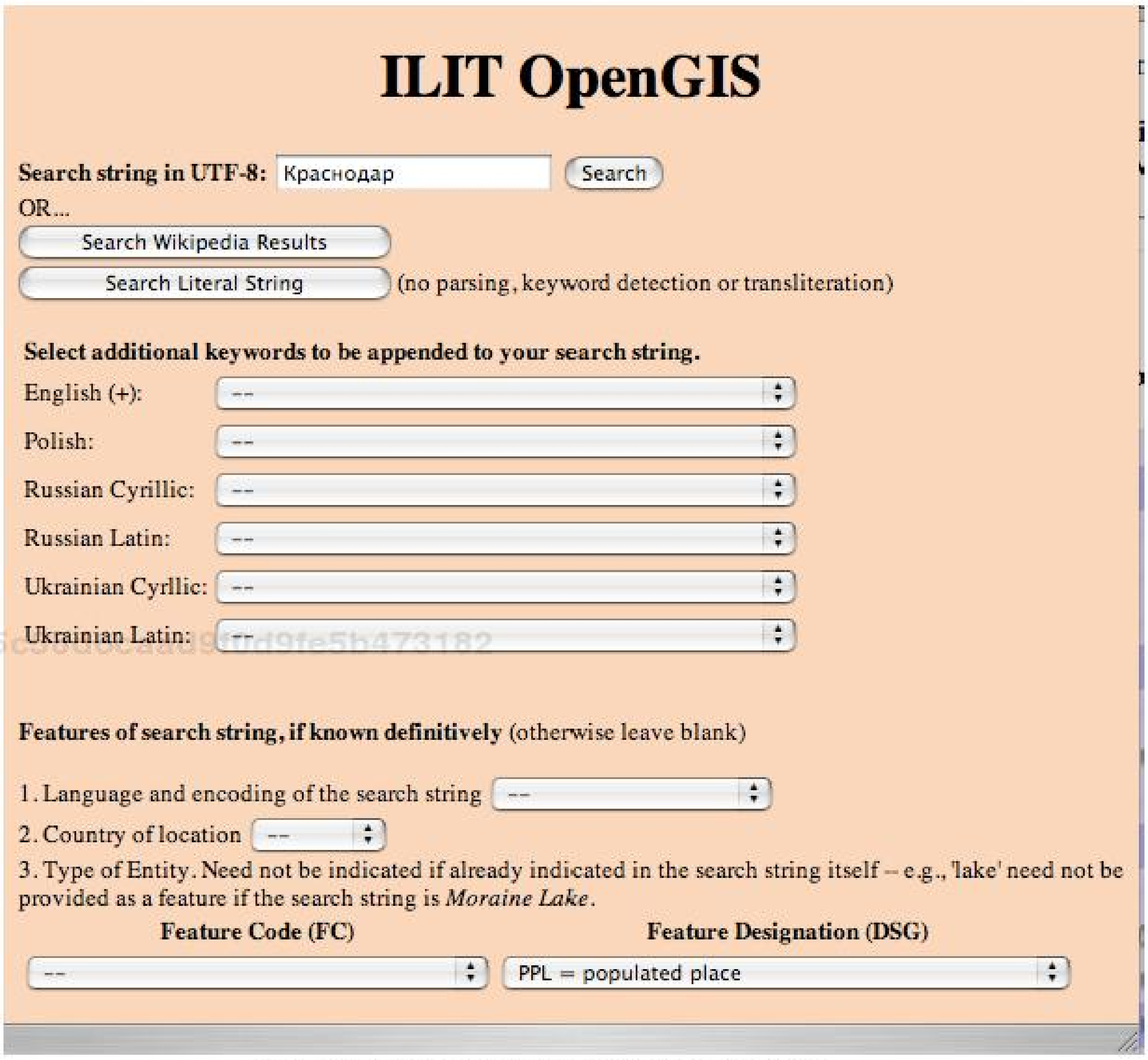
<meta name="keywords" **content="Краснодар**,Нетребко,
Анна Юрьевна,Бондарь Александр,[…] " />

Our engine currently does only light parsing of the input data, as by inserting underscores between multi-word entities and removing parentheses. We did not download Wikipedia prior to carrying out our experiments, although in retrospect that would have been well advised.

Step 7. The final enhanced database contains, in addition to the 25 original fields: (a) a **Native** field containing Cyrillic variants for Russia and Ukraine; (b) a **Stripped_Latin** field containing the Full_Form (in Latin) without keywords; (c) a **Stripped_Bare** field containing the Full_Form_ND (diacritic-free Latin) without keywords; (d) a **Stripped_Native** field containing the Cyrillic form without keywords; (e) a new **Keywords-database** that includes multi-lingual keywords with their DSG correspondences; and (f) a new **Wikipedia-database** that includes the multi-lingual variants of all found entities along with their language attributions and explicit link (via UFI/UNI) to their original database anchor.

4.3. The Search Interface

The application we used to test the utility of our database supplementation and multi-lingual transliteration engine is a search engine that is similar to the one that currently accesses the GNDB but contains additional search features. The interface is shown in Figure 3.



ILIT OpenGIS

Search string in UTF-8: Краснодар

(no parsing, keyword detection or transliteration)

Select additional keywords to be appended to your search string.

English (+): --

Polish: --

Russian Cyrillic: --

Russian Latin: --

Ukrainian Cyrllic: --

Ukrainian Latin: --

Features of search string, if known definitively (otherwise leave blank)

1. Language and encoding of the search string --

2. Country of location --

3. Type of Entity. Need not be indicated if already indicated in the search string itself -- e.g., 'lake' need not be provided as a feature if the search string is *Moraine Lake*.

Feature Code (FC): --

Feature Designation (DSG): PPL = populated place

Figure 3. The GeoMatch Search Interface.

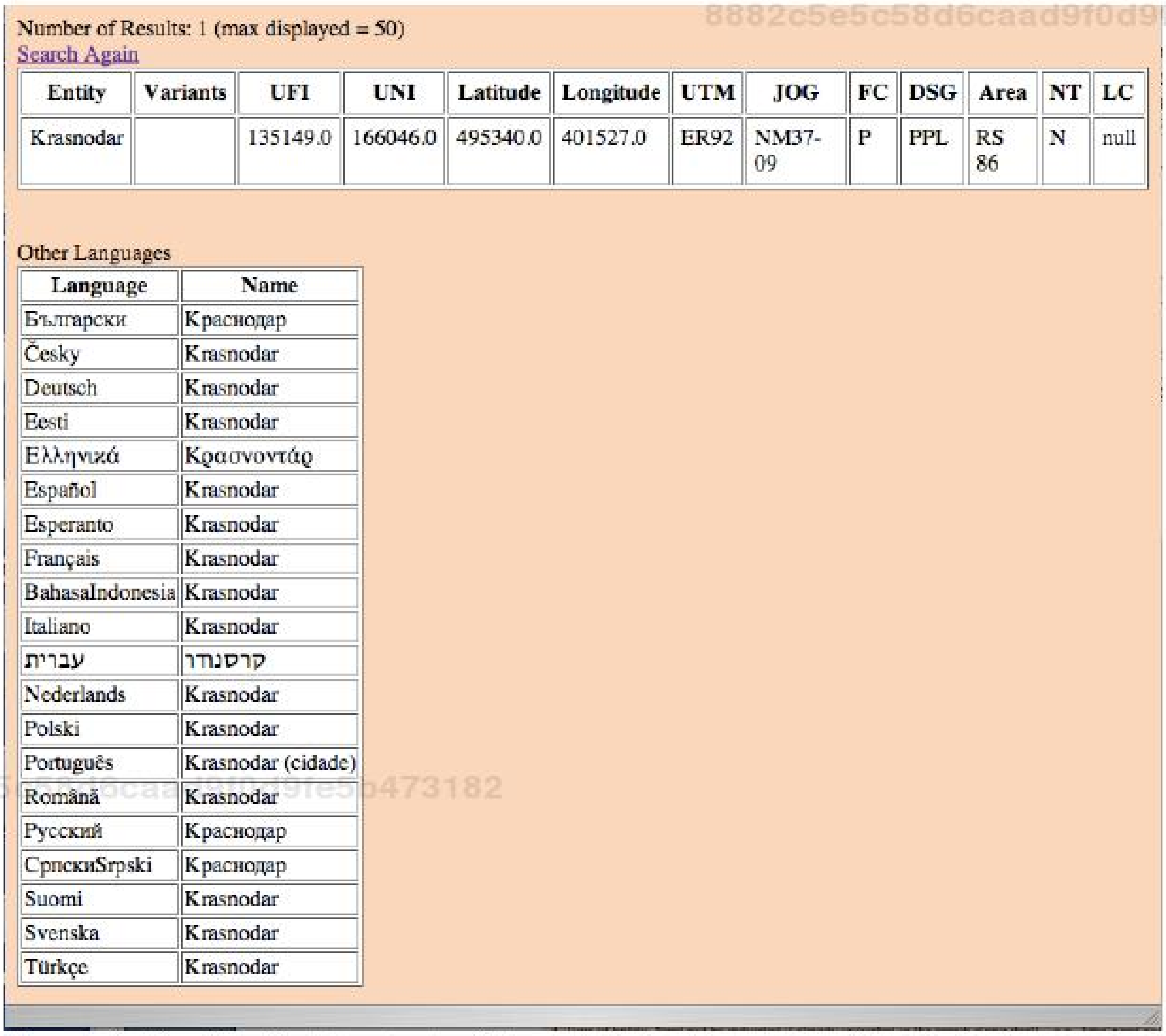
A search string can be in entered in any language and script as long as it is in UTF-8 encoding. In the figure, the search string is in Russian. There are three search buttons, all of which return a select inventory of properties of the entity, drawn from the GNDB, as well as any multi-lingual variants found in Wikipedia.

Search Searches the Russia, Ukraine and Polish databases using the main search algorithm, described below.

Search Wikipedia Results Searches the Wikipedia database for strings in other languages.

Search Literal String Searches the Russia, Ukraine and Poland databases on the string literally as typed, with no transliteration or keyword processing.

The output of the search in Figure 3 is shown in Figure 4. The Wikipedia results are below the main database information. In this prototype, we display only a subset of features for each entity (due to screen real estate), and we permit the user to constrain the search using only select features (e.g., DSG but not latitude or longitude); however, it would be trivial to expand the display and feature selection to include all features.



Number of Results: 1 (max displayed = 50)
Search Again

Entity	Variants	UFI	UNI	Latitude	Longitude	UTM	JOG	FC	DSG	Area	NT	LC
Krasnodar		135149.0	166046.0	495340.0	401527.0	ER92	NM37-09	P	PPL	RS 86	N	null

Other Languages

Language	Name
Български	Красnодар
Česky	Krasnodar
Deutsch	Krasnodar
Eesti	Krasnodar
Ελληνικά	Κρασνοντάρ
Español	Krasnodar
Esperanto	Krasnodar
Français	Krasnodar
BahasaIndonesia	Krasnodar
Italiano	Krasnodar
עברית	קרסנודר
Nederlands	Krasnodar
Polski	Krasnodar
Português	Krasnodar (cidade)
Română	Krasnodar
Русский	Краснодар
СрпскиSrpski	Краснодар
Suomi	Krasnodar
Svenska	Krasnodar
Türkçe	Krasnodar

Figure 4. The GeoMatch output of the search for Краснодар (Krasnodar).

4.4. The Main Search Strategy

In describing the search algorithms, we first concentrate on the main Search function, which targets the three databases and four languages (including English) treated in this system. When launching a search, the user may choose to specify values for any of the following features. None are required and if none are entered, all relevant algorithms are called in turn.

1. The language and script of the search string: *English; Polish, Latin; Polish, Extended Latin; Russian, Cyrillic; Russian, Latin; Ukrainian Cyrillic; Ukrainian, Latin*.
2. The location of the entity: Russia, Poland or Ukraine.
3. The feature specification (FS), which can be one of 9 values.
4. The feature designation (DSG), which can be one of over 500 values.

The user can insert keywords into the search string by selecting them from the menu of keywords for each language/script combination. For the Russian Latin and Ukrainian Latin keywords, only one of the many transliterations understood by the system is listed. Typing in a keyword is equivalent to selecting the associated feature designation code (DSG).

A flowchart for the main search strategy is presented in Figure 5.

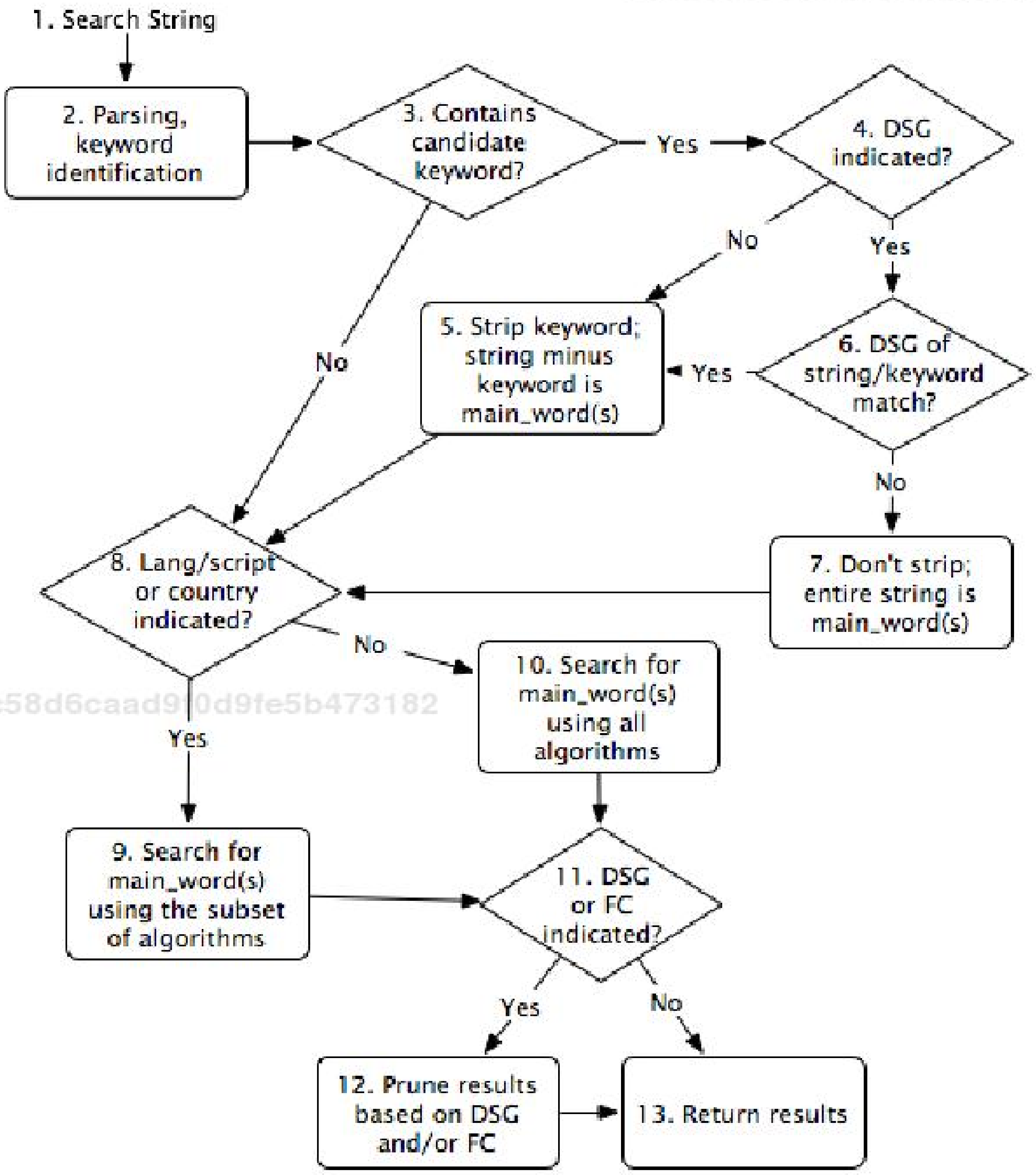

1. Search String
2. Parsing, keyword identification
3. Contains candidate keyword?
Yes
4. DSG indicated?
No
Yes
5. Strip keyword; string minus keyword is main_word(s)
Yes
6. DSG of string/keyword match?
No
No
7. Don't strip; entire string is main_word(s)
8. Lang/script or country indicated?
No
10. Search for main_word(s) using all algorithms
Yes
9. Search for main_word(s) using the subset of algorithms
11. DSG or FC indicated?
Yes
No
12. Prune results based on DSG and/or FC
13. Return results

Figure 5. The main search strategy.

Step 1. The search string is input with optional feature selection.

Steps 2-7. The string is parsed and keyword identification and stripping is carried out using the same methods as described in the database population task, Step 2.

Steps 8-10. The search algorithms are divided by language/script and country, with one algorithm devoted to each pair, making 21 algorithms in all. If either the language/script or the country is explicitly provided by the user, then the number of algorithms that have to be launched is decreased accordingly. If no features are provided, all algorithms are launched in turn and all results are returned.

Our search algorithms attempt to cover *any* reasonable transliteration of a string. We do, however, assume that when strings are transliterated into Latin, they will be transliterated by an English speaker; therefore, we use, for example, *v* to indicate phonetic [v], not *w*, as would be done by German speakers. Of course, Polish also uses *w* for [v], but this and other special Polish orthographic details are accounted for explicitly in our transliteration tables for language pairs that include Polish.

A thumbnail sketch of the main search algorithms for our countries and languages of interest is below, divided into conceptual classes. Recall the contents of the following fields that we have added to the original database:

Stripped_Native: Cyrillic forms with keywords stripped
Stripped_Latin: Latin (with diacritics) forms with keywords stripped
Stripped_Bare: Latin (without diacritics) forms with keywords stripped

The following abbreviations are used in the algorithms: en (English); ru (Russian); uk (Ukrainian); pl (Polish).

The language of input is the native one for the country of location: Russian/Russia, Ukrainian/Ukraine, Polish/Poland.

For Russia and Ukraine

If the input is in Latin, transliterate using the en-ru or en-uk engine, then search in the Stripped_Native field. The reason we transliterate into Cyrillic rather than just searching for the Latin is that there are many possible Latin variants for many of the entities, and only one is recorded in the database. Rendering the string back into Cyrillic neutralizes this problem.

For Poland

Search the Stripped_Latin and/or Stripped_Bare fields; if the script was indicated by the user (Extended Latin or Basic Latin) only one of these fields need be searched.

The language of input is not the native one for the country of location.

Transliterate the input into the native language using the appropriate transliteration engine(s). This can comprise one or two stages of transliteration. For example,

One-stage transliteration: The string is in Cyrillic Ukrainian but the place is located in Russia. Use the uk-ru engine to generate a Cyrillic Russian variant and search for it in the Stripped_Native field of the Russia database.

Two-stage transliteration: The string is in Ukrainian Latin but the place is in Russia. Use the en-uk engine to generate a Cyrillic Ukrainian variant, then use the uk-ru engine on that output to generate a Cyrillic Russian variant. Search for the Cyrillic Russian variant in the Stripped_Native field of the Russia database.

An important aspect of our transliteration strategy is to permit many different transliterations of certain letters and letter combinations. This reflects the fact that: (a) a user might be searching for something heard, not written, in which case he will render it phonetically, and (b) a user cannot be expected to always follow canonical transliteration schemes, which can never be agreed upon anyway.

Consider the following Polish place names and how they might sound to a speaker of English [ ] or Russian { }.

Bóbrka	[Bubrka] {Бубрка}
Bartężek	[Bartenzhek / Bartezhek] {Бартенжек / Бартежек}
Bądze	[Bondze] {Бондзе}

If the user were conveying these place name based on what he heard, he would likely use the search strings above. However if he saw the name in print, he might decide simply to ignore the diacritics, ending up with a different inventory of search strings. For this reason, our transliteration tables contain many target possibilities for many of the source letters and letter combinations: e.g., Polish *ó* can be rendered as English *u* or *o*, and as Russian *у* or *о*; similarly, Polish *ą* can be rendered as English *on, om* or *o*, and as Russian *ом, он* or *о*.

Consider the following example of 2-stage transliteration. Russian Latin input is used to search for the Polish place name *Byczoń*. This ends with a palatalized *n,* which can be represented in Russian Latin as *n'* and in Russian Cyrillic as *нь*. However, it is common to leave out the apostrophe indicating palatalization when using Russian Latin (and many English speakers do not hear the palatalization to begin with), which means that Russian Latin *n* can be intended to mean either a palatalized (*нь*) or an unpalatalized (*н*) letter.

The algorithm called when a Russian Latin string is used to search for a place in Poland is first to transliterate from Russian Latin to Russian Cyrillic, then to transliterate from Russian Cyrillic to Polish. The possibility that palatalization will not be indicated in the original Latin string must be handled either in the Russian Latin to Russian Cyrillic transliteration, in which case every *n* can mean *н* or *нь*, or in the Russian Cyrillic to Polish transliteration, in which case every *н* must be understood as either *n* or *ń*. Clearly, if we insisted that people input every string "correctly", we could circumvent some such problems; however, this would be unrealistic and not in the service of users. In short, extensive system testing suggested the need for far more transliteration correspondences than those that would reflect typical, canonical transliteration schemes.

The reason why our search application does not suffer from a one-to-many transliteration scheme is that there is no need for exactly one ouput from the transliteration engine: all of the generated strings can be searched for the in database and typically only one of them is found. Many generated candidates represent impossible strings in the target language, which could be filtered out by language-specific contextual rules that we did not, however, develop for this prototype. If the approach were expanded to many more languages and countries, however, we might need to prune the output results in order to not generate false positives. In our testing so far we have not had problems with false positives, and even if we did, this search application has a person as the end user, and that person could filter out the false positives using the inventory of features returned for each hit. Here we touch on an important aspect of this – or any – application: it must be catered to what it is supposed to do, with development efforts targeted at namely those goals. For this application, robustly finding matches in the database is more important than generating a single answer for multi-stage transliteration.

Steps 11-12. If the FC or DSG features are provided, these are used to prune the search results. They could alternatively have been used to constrain the search at the outset.

4.5. Additional Search Strategies

The two additional search buttons permit searching the Wikipedia-database directly and searching the main database without keyword processing or transliteration. The latter is a slower search in which all relevant fields are searched: Full_Name, Full_Name_ND, Stripped_Latin, Stripped_Bare and, for Russia and Ukraine, Native and Native_Stripped. One situation in which the latter search strategy might be useful is the following: A user knows that his search string includes a word that looks like a keyword but is not; however, he cannot block keyword interpretation by entering the entity's correct DSG because he does not know it. In this case, seeking an exact string match is a better search strategy.

4.6. New Possibilities Provided by the GeoMatch Search Strategy

Using the GeoMatch search strategy a user has the following search support not provided by the GEOnet Names Server (GNS):

- He can provide a search string in one language for an entity located in a place having a different native language.
- He can provide a search string that contains the main search word(s) in one language/script and a keyword in another language/script and still have the appropriate keyword interpretation carried out.
- He can provide search strings in any language, even those not explicitly targeted in this application, since the Wikipedia-database results cover a wide range of languages.
- He can receive not only the geographical information from the original GNDB but also multi-lingual variants and their language attributions.
- He can constrain the search not only using the seed database features but also using language and script.

4.7. Evaluation

We attempted to evaluate GeoMatch by randomly selecting entities from each country database and searching for them using all possible language/script combinations; however, the results were not indicative of the progress made due to the nature of the data and the scope of the project. GeoMatch was a small prototype project aimed at developing algorithms, not cleaning databases. As with any large databases, those for

Russia, Ukraine and Poland presented challenges related to inconsistency and underspecification of data. For example:

- Although the Full_Form fields are supposed to contain strings in the language of the given country (albeit in Latin), strings in many other languages and scripts are scattered throughout, not indicated as such using the available Language Code (LC) field
- Compiling the inventory of keywords and their DSG correspondences was a big job, and we still have not achieved complete coverage (especially in terms of finding all "meaningful" DSG correspondences for each keyword)
- There are some outstanding errors in our initial one-to-one Cyrillic transliteration for population of the Native field that can only be hand corrected due to actual ambiguities.

Using less formal, glass-box evaluation methods, we became convinced that the algorithms show a lot of promise and that proof of concept was achieved. However, further evaluation will need to wait for a continuation of the project, when the abovementioned trivial but evaluation-affecting problems have been resolved.

As concerns the evaluation of the Wikipedia aspect of the work, numerically, the results might seem like a drop in the bucket, with around 1 in 25 being found. For each hit, a subset of the languages represented in Wikipedia provided a variant. However, it is important to note that most geographical entities that have multi-lingual *translations* (which are idiosyncratic) rather than *transliterations* (which follow rules) are the historically more important, well-known places (like *Moscow*), which are likely to be accounted for in Wikipedia, making the Wikipedia supplements extremely valuable. Moreover, the Wikipedia results show proof of concept that using on-line resources either to gather or to vet posited variants is realistic and useful.

However, even when an entity is found in Wikipedia, that is not a guarantee that it refers to the intended place. It is possible that translation/transliteration decisions will be different for different types of proper names that are rendered identically in English. Table 4 shows an example from Russian, in which the English string *Jordan* is translated/transliterated in three different ways for different types of entities.

Table 4. Various renderings of *Jordan* in Russian.

English	*Gloss*	*Russian (Cyrillic)*	*Russian (back transliterated)*
Jordan	a country	Иордания	Iordanija
Jordan	a river	Иордан	Iordan
Jordan	a person	Джордан	Dzhordan

We provide back transliterations in column 4, using one of the many Russian-English transliteration schemes, simply to orient readers not familiar with Cyrillic as to the type of morphological and phonetic distinctions being conveyed. This particular example will not prove problematic for our current engine because our current engine only accepts exact matches of Wikipedia head entries, and the head entries differ for each of the entities, as shown in Table 5.

Table 5. Wikipedia head entries that include the word *Jordan.*

English, as in Wikipedia	*Russian (Cyrillic), as in Wikipedia*	*Russian (back transliterated)*
Jordan	Иордания	Iordanija
Jordan River	Иордан (река)	Iordan (reka)
Neil Jordan	Джордан, Нил	Dzhordan, Nil

However, we can imagine that there could be cases in which identical head entries – be they composed of a single word or multiple words – could have different renderings in a given language when referring to different types of entities. In addition, this problem will be met with more frequently when expand our Wikipedia matches to include substrings: for example, if Wikipedia did not have an entry for *Jordan* but did have an entry for *Jordan River*, our engine could hypothesize that the rendering of *Jordan* would be the same when used independently as when used in collocation *Jordan River*. While this strategy will very often work, it clearly will not always work and will require external attestation, as by corpus search.

The second problem is that authors of Wikipedia pages do not always precisely agree as to how to represent the head entries. Table 5 shows two such cases: the Russian equivalent of *Jordan River* has the word for 'river' in parentheses, and the Russian equivalent of Neil Jordan has the first and last names reversed with a comma in between. Another example is that the English entry for *Los Angeles* uses the head entry *Los Angeles, California* whereas the Russian equivalent just lists the name of the city, *Лос-Анджелес*, without the state. Parsing and semantic analysis of the head entries in each of the linked languages would be the optimal method of detecting such lacks of parallelism. The algorithms for such parsing and analysis are certainly less complex than those needed for the typical unconstrained named entity recognition task, in which detecting the span of the named entity in open text and determining its semantic class (e.g., person vs. organization) are central. The work needed to clean the results of the Wikipedia extraction task is, therefore, more a matter of development than research, since the parser and semantic analyzer for each language need to be parameterized to include the correct inventory of generic terms (not only for geographic entities), relevant word order constraints, and perhaps a search of other named entities in the language to detect things like the state 'California' being appended to the city 'Los Angeles' in the example above.

We did not attempt to vet place names using a traditional Web search engine – something that certainly could have been done. However, vetting variants that way would not have provided cross-linguistic variants, so finding entities in Wikepedia would be preferable.

As concerns comparing GeoMatch with other systems, the best locus of comparison is NewsExplorer (http://press.jrc.it/NewsExplorer/home/en/latest.html). NewsExplorer clusters around 15,000 news articles a day in 40 languages, extracting and matching up named entities across languages and using them to populate a large multi-lingual database [20], [21]. Although this system is very relevant to the work reported here, it does not supercede this work for three reasons. First, the NewsExplorer database is not publicly available, though some search functions are. Second, the reported methods are not sufficiently detailed to make them truly reproducible: e.g., only 9 of 30 "substitution rules" that were found useful for transliteration tasks are described. Third, the system does not solve several kinds of problems that our development efforts are seeking to address. For example, the methods implemented by NewsExplorer require that the search string be a multi-word entity in order to cut down on spurious results; however, many (perhaps, even, the majority of) geographical entities are single-word entities. So the goals pursued by GeoMatch and NewsExplorer are quite similar, but different methods are employed that exploit different available resources and processors.

4.8. How to Extend the Coverage of the GeoMatch System

Knowledge-based systems that cover only a subset of a larger problem bear the burden of proof that they can be expanded to cover the whole problem space in finite time and with finite resources. Let us consider the ideal for the environment under discussion here and how the work already accomplished will support that.

1. The geographical databases for all countries of the world should be automatically provided with a reasonably confident native-script variant that could be validated over time using digital resources. Our current transliteration engine can accept transliteration tables for any language pairs as long as they are in UTF-8. (Recall that it requires no language-specific rules.) Since the original Latinization of place names used when building the GNS resource was supposed to have been done using a single transliteration system for each language, there should not be too much spurious ambiguity. Relatively fast analysis of the output of automatic transliteration can be followed either by improvement of the transliteration tables and rerunning of the data, or by global changes to the transliterated variants to correct recurring problems. Idiosyncratic aspects will naturally need to be hand corrected.
2. Attested multi-lingual variants for entities in all countries should be extracted from resources like Wikipedia and stored as a database supplement. The success of this task, of course, depends entirely on what the world community decides to enter into Wikipedia or what can be found on sites reporting news, current events, etc.
3. An inventory of keywords in all languages, along with their valid DSG associations, should be compiled. Although creating a full inventory of keywords that might represent the hundreds of DSGs would take some time, particularly as one might have to be a specialist to tell them apart, covering the most prominent 100 or so would be very fast for a native speaker. Such an inventory could be expanded over time.
4. The keyword stripping algorithm should be amended, if necessary, to cover language-specific orderings of "meaningful" keywords (e.g., in the string *River ABC Meadow*, would River or Meadow be the keyword for the given language?).
5. Multi-lingual access to any of the country databases should be supported so that a user could, e.g., type in a string in Bulgarian when looking for a place name in Turkey. This task is the most complex, but it seems that the complexity could be moderated using the notion of Language Hubs, which would be not unlike airport hubs: just as one does not need to be able to fly directly from every city to every city, one does not need to have a transliteration engine from every language to every language. Certain languages could act as hubs, permitting "translation passage" to and from a large number of languages, not unlike what is done linguistically at the United

Nations. A given language might have access to one hub (e.g., R3) or more than one hub (e.g., A2).

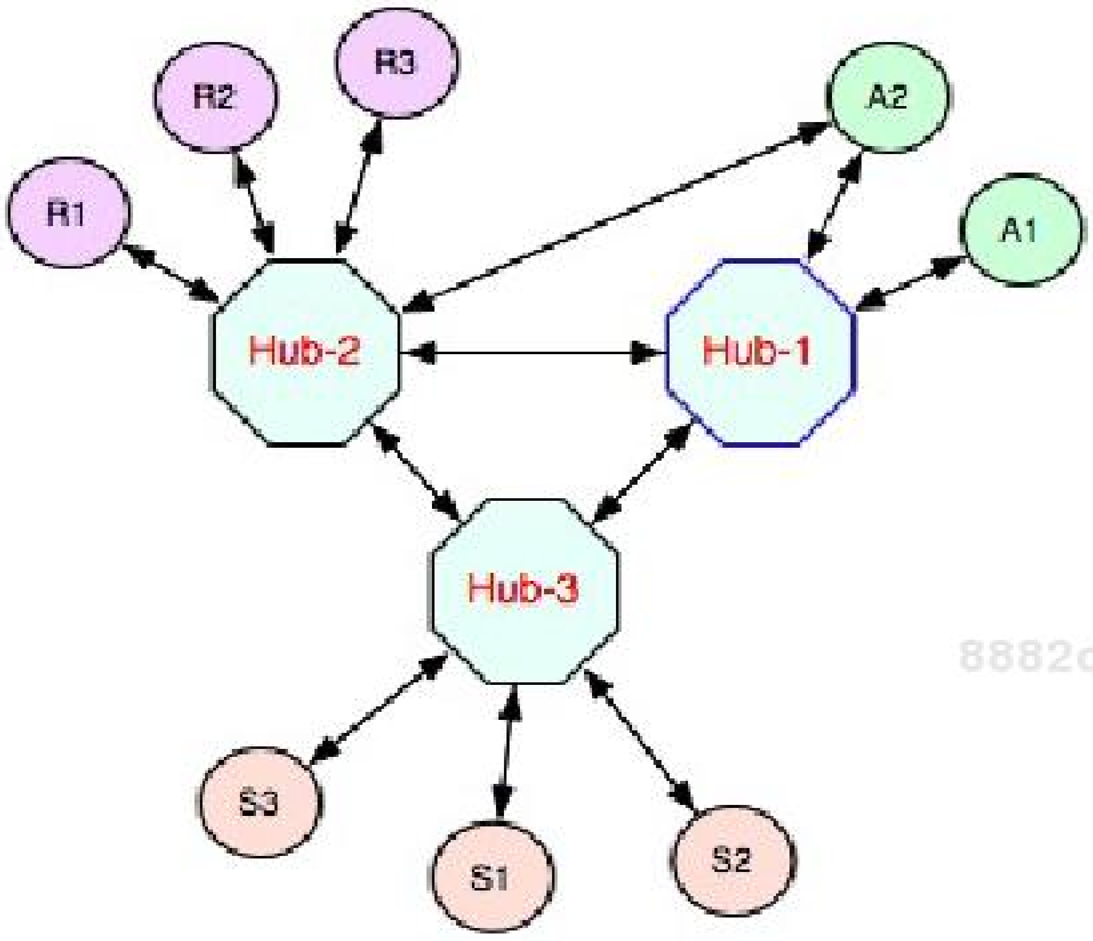


Figure 6. Using language hubs to expand GeoMatch.

Language hubs should be chosen with practical considerations in mind like the level of international prominence, how closely the spelling in the language reflects the phonetics, and how many other languages might readily feed into the given hub. Of course, this is simply a preliminary suggestion, the details of which would require further study.

5. Final Thoughts

The systems described above, Boas II and GeoMatch, seek to support proper name recognition for a wide variety of languages. Boas II provides the infrastructure to quickly ramp up a proper name recognizer for any language with no external resources needed. GeoMatch serves as an example of how an available knowledge base that was not initially developed to serve NLP can be expanded and leveraged to support multilingual language processing. These enabling technologies could be exploited in applications ranging from question answering to machine translation to the automatic generation of object or event profiles through the mining of multi-lingual text sources. For the purposes of this volume, the import of these systems lies in the fact that are as applicable for low- and middle-density languages as they are for high-density languages.

Acknowledgements

This work was supported by two grants from the United States Department of Defense.

References Cited

[1] Bennett, Scott W., Aone, Chinatsu Aone and Lovell, Craig. 1997. Learning to tag multilingual texts through observation. *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*.

[2] Berners-Lee, T., Fielding, R & Masinter. L 2005. Uniform Resource Identifier (URI): Generic Syntax. The Internet Society. Available at http://www.gbiv.com/protocols/uri/rfc/rfc3986.html.

[3] Bickel, Daniel M., Richard Schwartz and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning* 34(1-3): 211-231.

[4] Chinchor, Nancy. 1997. MUC-7 Named Entity Recognition Task Definition. Version 3.5, September 17, 1997. Available at http://www-nlpir.nist.gov/related_projects/muc/proceedings/ne_task.html.

[5] Coates-Stephens S. 1993. *The Analysis and Acquisition of Proper Names for the Understanding of Free Text*. Hingham, MA: Kluwer Academic Publishers.

[6] Cucerzan, Silviu and David Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence. *Proceedings, 1999 Joint SIGDAT Conference on Empirical Methods in NLP and Very Large Corpora*, pp. 90-99.

[7] Grishman, Ralph. 1995. Where's the syntax? The New York University MUC-6 System. *Proceedings of the Sixth Message Understanding Conference*.

[8] Ishida, R. 2006. An Introduction to Multilingual Web Addresses. W3C Architecture Domain. Available at http://www.w3.org/International/articles/idn-and-iri/.

[9] Karkaletsis, Vangelis, Georgios Paliouras, Georgios Petasis, Natasa Manousopoulou, Constantine D. Spyropoulos. 1999. Named-Entity recognition from Greek and English texts. *Journal of Intelligent and Robotic Systems* 26 (2): 123-135.

[10] Màrquez, Lluís, Adrià de Gispert, Xavier Carreras, and Lluís Padró. 2003. Low-cost named entity classification for Catalan: Exploiting multilingual resources and unlabeled data. *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition*, pp. 25-32.

[11] McDonald, D. 1996. Internal and external evidence in the identification and semantic categorization of proper names. In B. Boguraev and J. Pustejovsky, editors, *Corpus Processing for Lexical Acquisition*, pp. 21-39.

[12] McShane, Marjorie, Sergei Nirenburg and Ron Zacharski. Mood and modality: Out of theory and into the fray. 2004. *Natural Language Engineering* 19(1): 57-89.

[13] McShane, Marjorie and Sergei Nirenburg. 2003. Blasting open a choice space: Learning inflectional morphology for NLP. *Computational Intelligence* 19(2): 111-135.

[14] McShane, Marjorie and Sergei Nirenburg. 2003. Parameterizing and eliciting text elements across languages for use in natural language processing systems. *Machine Translation* 18(2): 129-165.

[15] McShane, Marjorie, Sergei Nirenburg, Jim Cowie and Ron Zacharski. 2002. Embedding knowledge elicitation and MT systems within a single architecture. *Machine Translation* 17(4):271-305.

[16] McShane, Marjorie and Ron Zacharski. 2005. User-extensible on-line lexicons for language learning. Working Paper #05-05, Institute for Language and Information Technologies, University of Maryland Baltimore County.
Available at http://ilit.umbc.edu/ILIT_Working_Papers/ILIT_WP_05-05_Boas_Lexicons.pdf.

[17] McShane, Marjorie, Ron Zacharski, Sergei Nirenburg, Stephen Beale. 2005. The Boas II Named Entity Elicitation System. Working Paper 08-05, Institute of Language and Information Technologies, University of Maryland Baltimore County.
Available at http://ilit.umbc.edu/ILIT_Working_Papers/ILIT_WP_08-05_Boas_II.pdf.

[18] Mikheev, Andrei, Marc Moens and Claire Grover. 1999. Named entity recognition without gazetteers. *Proceedings of EACL* '99.

[19] Nirenburg, S. 1998. Project Boas: "A linguist in the box" as a multi-purpose language resource. *Proceedings of the First International Conference on Language Resources and Evaluation*, Granada, Spain.

[20] Pouliquen, B, Steinberger, R, Ignat, C. & de Groeve, T. 2004. Geographical Information Recognition and Visualisation in Texts Written in Various Languages. *Proceedings of the 19th Annual ACM Symposium on Applied Computing (SAC'2004), Special Track on Information Access and Retrieval (SAC-IAR),* vol. 2, pp. 1051-1058. Nicosia, Cyprus, 14 - 17 March 2004.

[21] Pouliquen, B., Steinberger, R, Ignat, C., Temnikova, I, Widiger, A, Zaghouani, W & Zizka, J. 2005. Multilingual person name recognition and transliteration. *Journal CORELA - Cognition, Représentation, Langage*.

[22] Wakao, T., R. Gaizauskas and Y. Wilks. 1996. Evaluation of an algorithm for the recognition and classification of proper names. *Proceedings of COLING-96*.