The Boas II Named Entity Elicitation System

Marjorie McShane, Ron Zacharski, Sergei Nirenburg, Stephen Beale

Working Paper 08-05

March 8, 2005

Institute for Language and Information Technologies
University of Maryland Baltimore County

Abstract. The Boas II Named Entity System collects information about the structure (i.e., syntax) and components of named entities in any language with the goal of permitting computer programs to correctly extract (interpret, etc.) named entities from texts written in that language.

1. Overview

The Boas II Named Entity Extraction System (referred to hereafter as Boas II) elicits information about named entities in any language (L) from informants who are speakers of that language. Based on that information, the system generates named entity extraction capabilities for L. The languages targeted are low-density languages for which no custom-made named entity recognition (NER) system exists. Boas II places primary emphasis on names of people, but also covers, at a coarser grain-size, names of companies, institutions, buildings, locations, geographical names and events (e.g., World War II). We call the system Boas II since its rationale and methodology derive from the original Boas system (see, e.g., Nirenburg 1998, McShane et al. 2002, McShane and Nirenburg 2003a, 2003b, McShane et al. 2004). The two systems are, however, fully independent.

This is a Web-based system that can be worked on through any web browser. The system runs off a local web server, which is installed when the system is loaded. We take a pattern-matching approach to named-entity recognition with no machine learning, since the latter typically requires large tagged corpora which are difficult and expensive to develop for low-density languages. However, the knowledge acquired during use of this system can support stochastic methods, as described in Section 2. An important aspect of the work is compiling—by means of iterative corpus-based methods—inventories of named entity components (e.g., personal names, family names). These inventories both support higher-level corpus work and improve the overall functioning of the named entity recognizer.

Configuring NER capabilities for a language 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 named entities, the syntactic patterns in which they participate, and some detection heuristics (e.g., capitalization, morphological triggers); he or she 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 named entity heuristics – like Chinese, which lacks capitalization – will require large inventories of name components before good results are achieved (we do not have any magic bullets for NER in such languages, nor do we 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 NER. 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 for NER. 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 in English is certainly not part of a named entity).

- **The breadth of the system.** Any subset of named entities can be the focus of a NER 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.

What we pursued in this project was the elicitation of information that we understood to be useful for machine processing. We did not incorporate many interesting factoids about name use in different languages, which have more sociolinguistic than computational linguistic import: for example,

- 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;
- 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 do reasoning based on cross-referencing family members: e.g., a system might using 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.

Before turning to an in-depth description of the system in Section 3, we present a brief overview of the state of the field in order to orient our work in this currently active realm of research.

2. Background on Named Entity Recognition

Named entity recognition attracted much attention in response to the MUC-6 "NER task", which included – in addition to the types of entities covered by Boas II – times, percentages and monetary amounts. The rules of the game of the MUC competitions significantly affected the approaches to NER selected by participants, since participants were provided with large tagged corpora – a prerequisite for most stochastic approaches. However, tagging corpora is expensive: it has been reported that tagging 100,000 words requires at least 33 hours by trained taggers (Bikel,

Schwartz, and Weischedel 1999); and 100,000 words isn't even very big for stochastic training. Therefore, the dominance of stochastic systems over pattern-matching ones should 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. However, both types of systems generally require a significant amount of static knowledge, in addition to morphological and/or syntactic processors. The following snapshots of some notable systems will serve as examples. The resources required by each are mentioned as they are of special import to the present discussion.

- BBN's IdentiFinder (Bikel, Schwartz, and Weischedel 1999) uses a hidden Markov model and a minimum of 100,000 words of training data to learn NER for a language; however it performs much better with a million words of training data (i.e., months of tagging). This group uses bigrams since the use of trigrams requires "exponentially more training data".
- An experiment in the supervised learning of NER in Greek involved bootstrapping from English (Karkaletsis et al. 1999). Resources required were: a tokenizer, a sentence splitter, a part-of-speech (POS) tagger, a gazetteer, a named-entity parser, and a large hand-tagged corpus.
- Another bootstrapping experiment involved Catalan (Marquez et al. 2003), 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.
- The NER system developed by Mikheev et al. 1999 makes relatively less use of gazetteers than most systems because of the known limitations of strong reliance on gazetteers: a) there are too many entities to list exhaustively; b) the list is always growing; c) there is overlap between, e.g., Washington as a place and as a person; and d) 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 (supplied through MUC) and a small inventory of names to learn NER.
- NYU's NER system for MUC-6 (Grishman 1995) reflected substantive changes from earlier implementations. For five MUCs, NYU used full syntactic then semantic analysis (based on a large grammar and lexicon of English) in a pattern-matching approach – but all of this machinery did not produce the desired results. So for MUC-6 they cut back and concentrated on specifics of the NER task: a gazetteer, various specialized dictionaries, scenario-specific terms, a POS tagger, and NP rules that were targeted to NER.
- A high-quality pattern-matching based NER system is SPARSER (McDonald 1996), which relies on both internal evidence (evidence from within the sequence of words and characters) and external evidence (evidence from the context of the phrases adjacent to the name). The required resources include: a lexicalized grammar; a closed-class lexicon; an open-class lexicon; 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 (Wakao et al. 1996). LaSIE is an all-purpose environment that includes syntax, semantics, ontology and discourse. NER processing in LaSIE relies on 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 semantic types of entities.

- A system that, to our knowledge, exceeds all others in its breadth and depth of coverage for English is the FUNES system developed by Coates-Stephens (1993). This system goes beyond the relatively restricted confines of the MUC task specification in pursuing all manner of named entities and their coreference relationships in English.

The common property of all the systems mentioned above is their reliance on a lot of pre-prepared data and/or programs – which is natural for systems that target languages that have long been the object of NLP, like English. However, if one needs to ramp up NER systems for low-density languages, the cost and feasibility of all prerequisites must be considered.

A recent trend in NLP has been to begin all work on a new language by tagging a corpus and applying machine learning to it. An NER system that follows this paradigm is SRA's RoboTag (Bennett et al. 1997), which is a tagging tool and machine learner for any L that requires a preprocessor, morphological analyzer and lexicon. Its goal is to allow the end user to build a tagging system for a language by giving examples of what should be tagged, rather than making the user learn a pattern language. (The rationale is that learning a pattern language is too difficult; we do not agree, as shown by the design of our system.)

Another system that targets low-density languages and requires even less static knowledge and user input is the Hopkins NER system (Cucerzan and Yarowsky 1999). Developers seek 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 texts with more or less informant input.

The Hopkins system has one crucial feature in common with Boas II: although more seed information is a plus, the NER engine can function with very little seed information as well – as expected, at a lower level of accuracy. One difference between the two systems concerns the lower limit of informant time: typically fifteen minutes for the Hopkins system, to amass about 100 seed words and a few heuristics, as contrasted with several hours for Boas II, to indicate syntactic components, patterns and heuristics and compile seed component inventories. Another difference concerns the approach – stochastic for Hopkins as contrasted with pattern matching for Boas II. There is, however, clear room for cooperative use of these systems. The Hopkins team 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. Boas II is strong on inventory building (in addition to matching patterns), so its output could be input to a system like Hopkins' – or any NER system that combines knowledge and machine learning.

3. Knowledge Elicitation in Boas II

Knowledge elicitation in Boas II is divided into three parts, as reflected in the Navigation page of Boas II, shown in Figure 1:

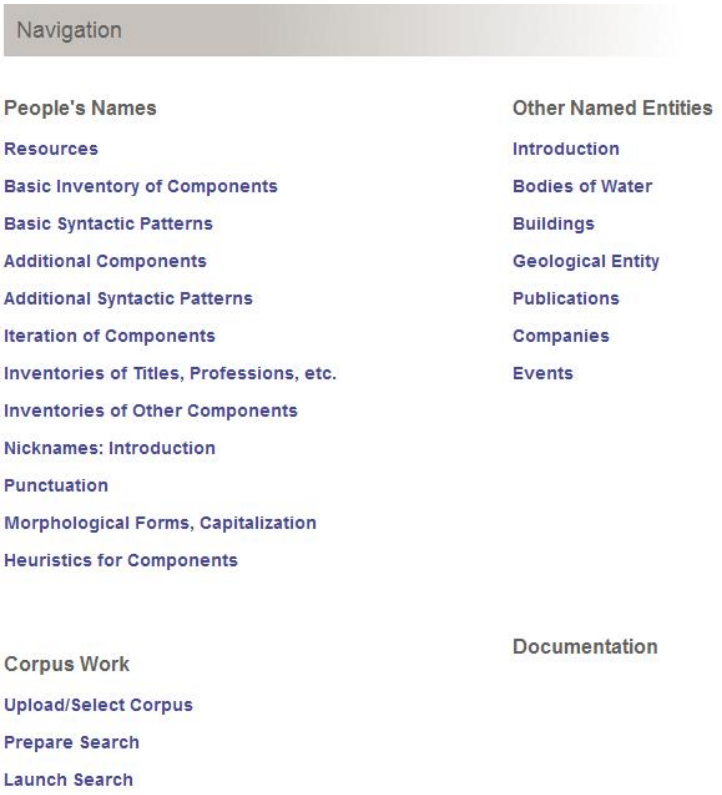


Figure 1. The navigation page of Boas II.

The first section (upper left quadrant: People's Names) collects information about people's names, the second section (upper right quadrant: Other Named Entities) collects information about other types of named entities, and the third section (lower left quadrant: Corpus Work) uses the information from either or both of those sections to drive corpus work, during which inventories of components are expanded, missing syntactic patterns are detected (to later be added), and approved corpus matches are used to populate a database of complex named entities.

Practically all subtasks are optional, though a few have self-evident prerequisites: e.g., one cannot search for people's names in a corpus without providing a corpus and at least one syntactic pattern for people's names; and one cannot provide an inventory of syntactic patterns for people's names without providing an inventory of components that can participate in those patterns. However, such constraints are few: far more tasks can be performed in any order, can be performed quickly at first then returned to later, or can not be performed at all: such decisions depend on how the NER data for L is to be incorporated into a larger text processing system.

The elicitation process begins with a login, in which the user inputs his/her name, email address, password, and language to be described (Figure 2).

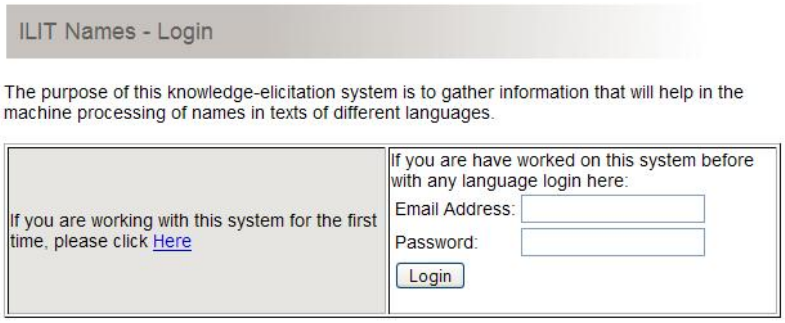


Figure 2. Login page for Boas II.

The combination of email and password represents the key to the given language profile. The subsections below briefly describe each subtask of Boas II.

3.1 People's Names

Introduction (Figure 3)**.** This is a description of the NER task and the Boas II system. It is geared toward novices.

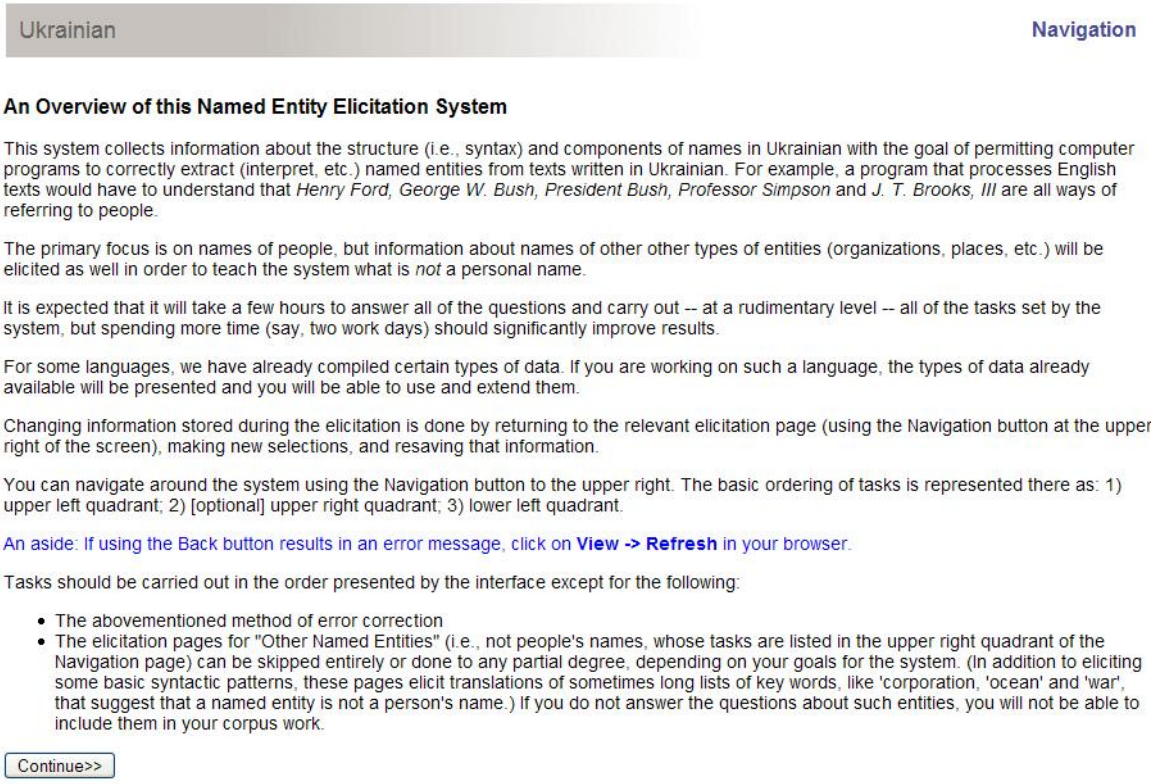


Figure 3. Introduction to Boas II during work on Ukrainian.

Resources (Figure 4)**.** This page lists resident resources, if any, for the selected language.

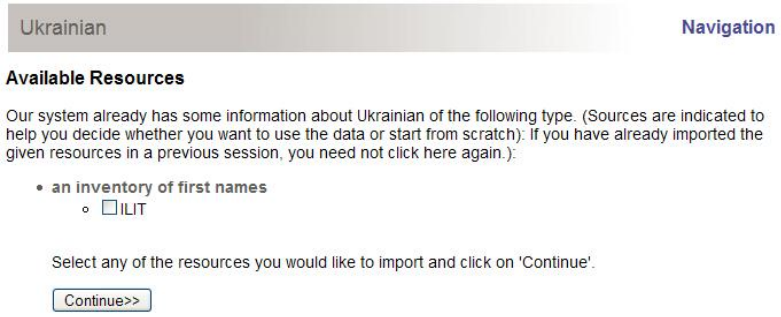


Figure 4. Resources page in Boas II during work on Ukrainian.

Lists of personal and family names for many languages were compiled by developers and are delivered with the system. The languages covered are as follows ($^{+}$ indicates personal names only; $^{\#}$ indicates family names only; no superscript indicates both): Afghany^{+}, African^{+}, Albanian, English, Arabic, Armenian^{+}, Basque^{+}, Bulgarian^{+}, Catalan^{+}, Chinese, Croatian, Czech, Danish, Dutch, Filipino$^{\#}$, Finnish, French, Georgian, German, Greek, Guyanese^{+}, Hausan^{+}, Hawaiian^{+}, Hispanic$^{\#}$, Hungarian, Icelandic^{+}, Igbo^{+}, Indian, Iranian^{+}, Irish^{+}, Italian^{+}, Japanese, Jewish^{+}, Korean$^{\#}$ Lithuanian, Mongolian, Norwegian, Pashto^{+}, Polish, Portuguese, Romanian, Russian, Scottish, Serbian^{+}, Slovak, Slovene^{+}, Swedish, Tamil^{+}, Thai, Tibetan^{+}, Turkish^{+}, Ukrainian^{+}, Welsh$^{\#}$.

There are, on average, several hundred personal names and a few thousand family names per language. In some cases, the lists are in transliteration. While this will not directly assist in NER for a language, transcribing the lists off-line might be worth the effort since this task is far easier than listing entities from scratch or attempting to find all such entities in corpus searches.

The resources that might be available for a language include ones compiled by developers (listed as ILIT, as in Figure 4) and ones compiled by other users of Boas II on the same server (listed under the individual's login name). So, if someone wants to do a profile of a language after someone else has worked on that language, all of the lists from the first person are available to the second.

Basic Inventory of Components (Figure 5). In this subtask, the user is presented with a list of category names, like Personal, Family, Initial and Conjunction from which he/she chooses the ones relevant for L. (Figure 5 shows only the first portion of category names; for the full list, see Appendix I.)

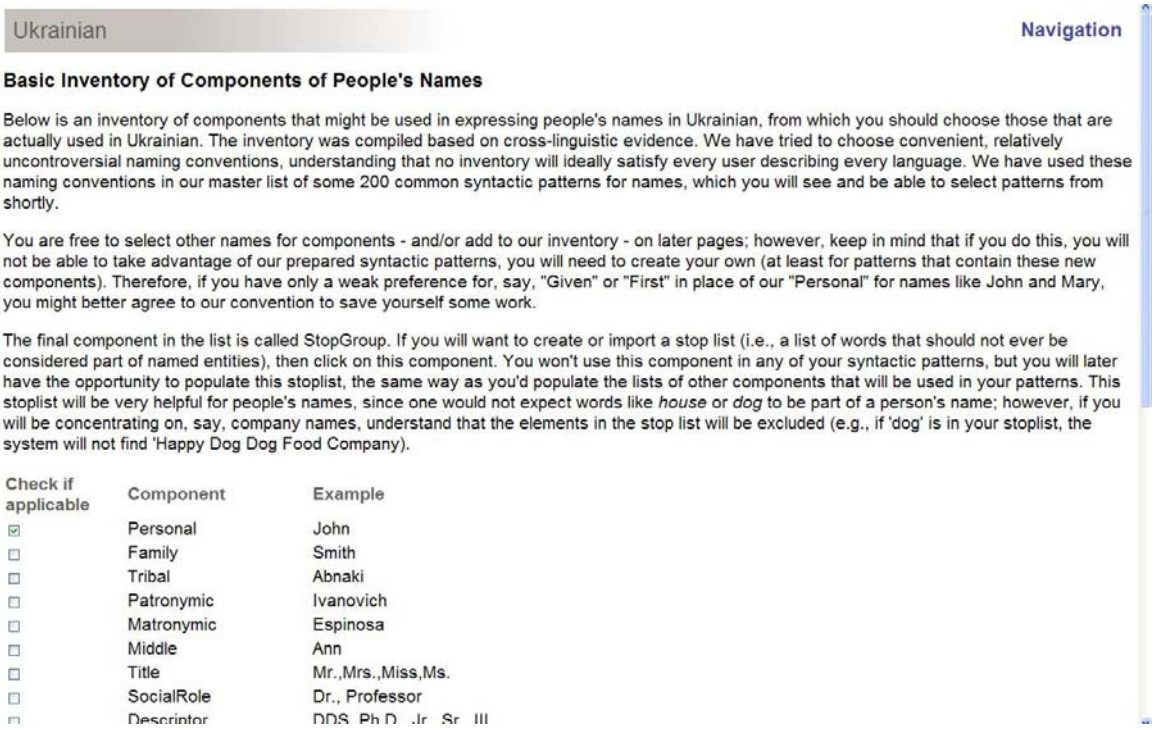


Figure 5. Elicitation of name components (an excerpt) page during work on Ukrainian.

As the instructions for this task describe, the inventory was compiled based on cross-linguistic evidence, which was gathered both by the developers and with the help of responses to a query to the Linguist List (for the results of that query, see http://www.ilit.umbc.edu). We tried to choose convenient, relatively uncontroversial naming conventions, understanding that no inventory will ideally satisfy every user describing every language. These naming conventions are used in our master list of some 200 common syntactic patterns for names (see next task). In later tasks, the user can add to our inventory, with additions most commonly reflecting types of components not covered in our list. However, additions may also reflect different names for components we already cover, if the naming convention we suggest does not suit a user's preferences. The disadvantage of renaming existing components is that the user cannot then take advantage of our prepared syntactic patterns and will need to create his or her own (this is not difficult, it just takes time; we did not have time to incorporate a special “renaming” function in the elicitation thread). Therefore, if users have only a weak preference for, say, "Given" or "First" in place of our "Personal" for names like John and Mary, they are advised to accept our conventions to save themselves some work.

Basic Syntactic Patterns (Figure 6)**.** 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/she will see will include Personal Family (Robert Jones), Personal Initial Family (Robert T. Jones), Personal (Robert), Family (Jones). The full inventory is shown in Appendix 2. The basic syntactic patterns do not include iteration of elements, which is elicited later.

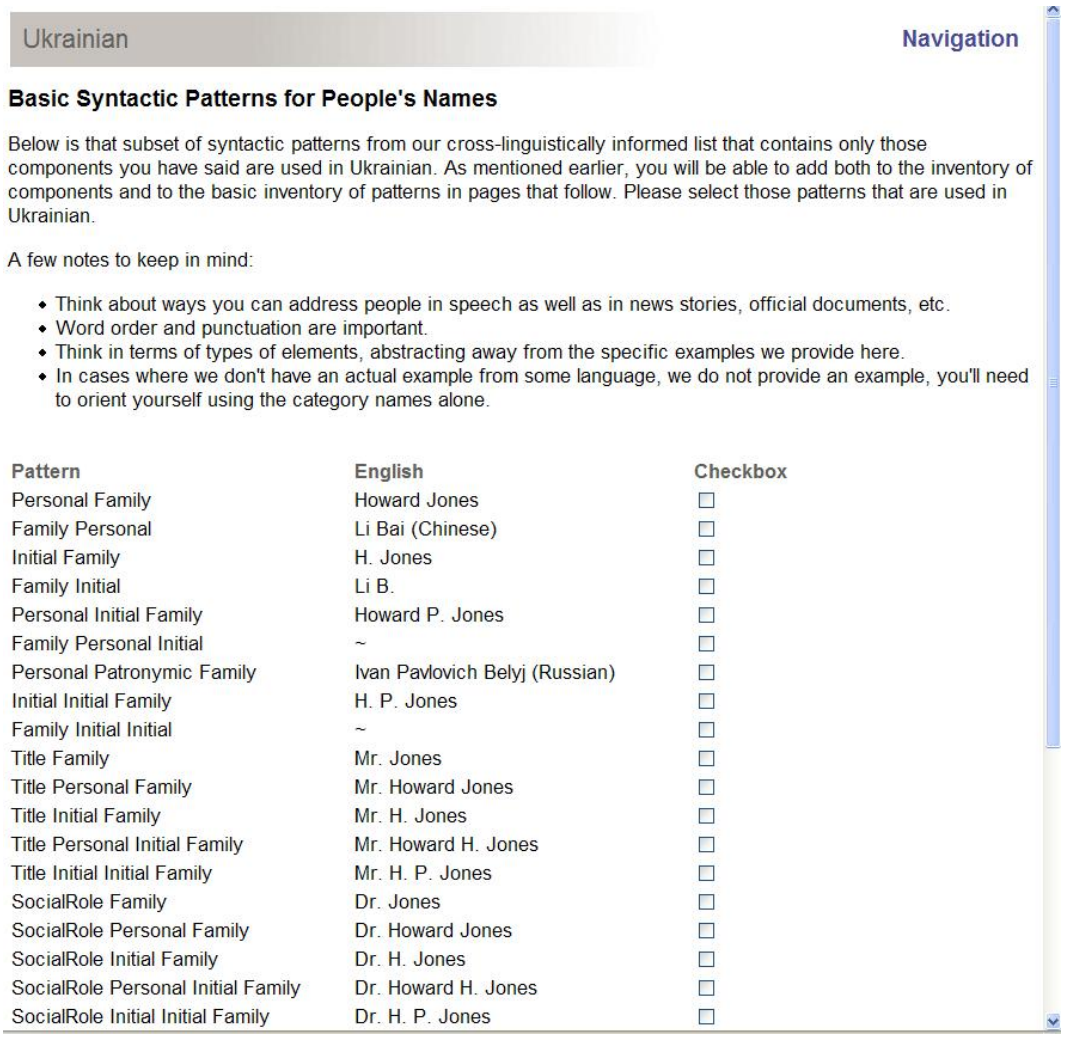


Figure 6. Elicitation of basic syntactic patterns during work on Ukrainian.

Additional Components (Figure 7)**.** The user is presented with his or her current inventory of components and permitted to supplement it, if necessary.

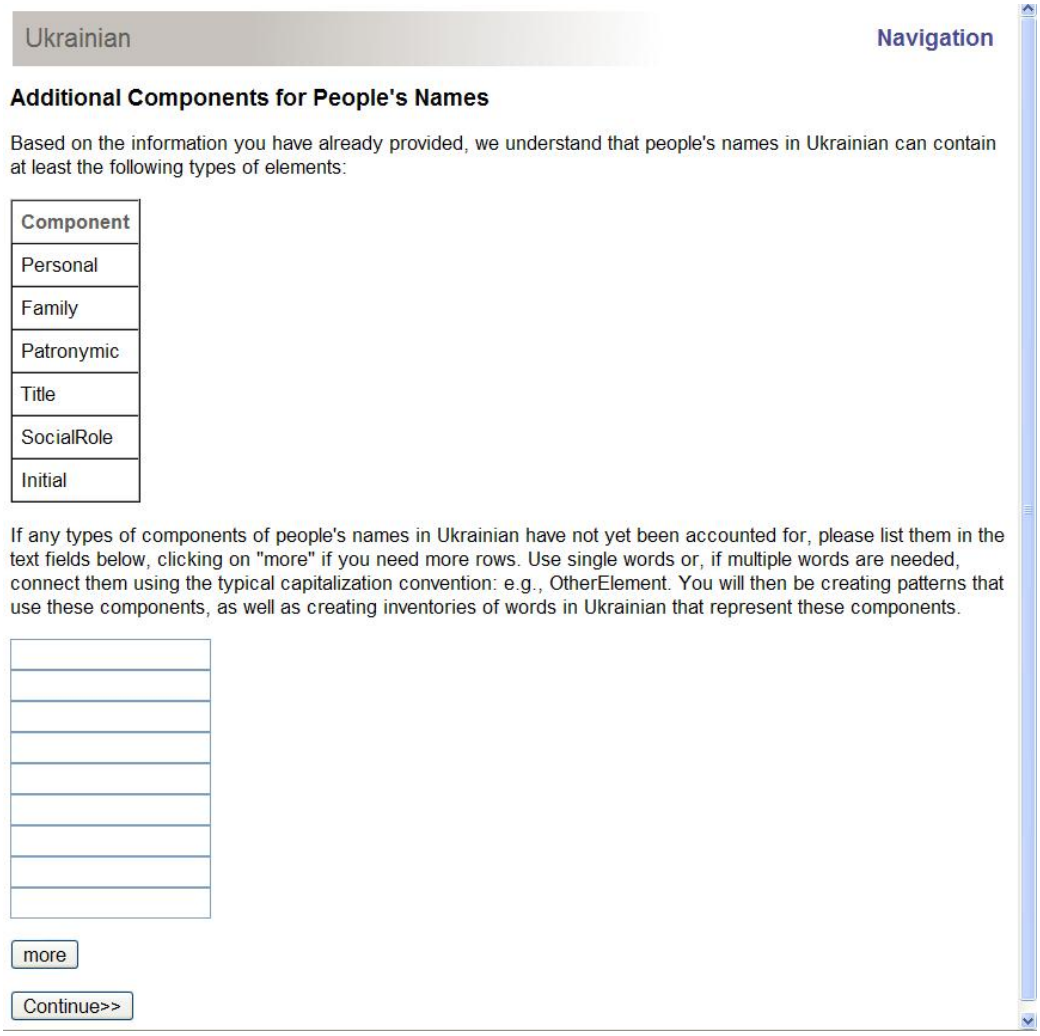


Figure 7. Elicitation of additional components during work on Ukrainian.

Punctuation (Figure 8)**.** 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 syntactic patterns). This inventory is used for parsing corpora.

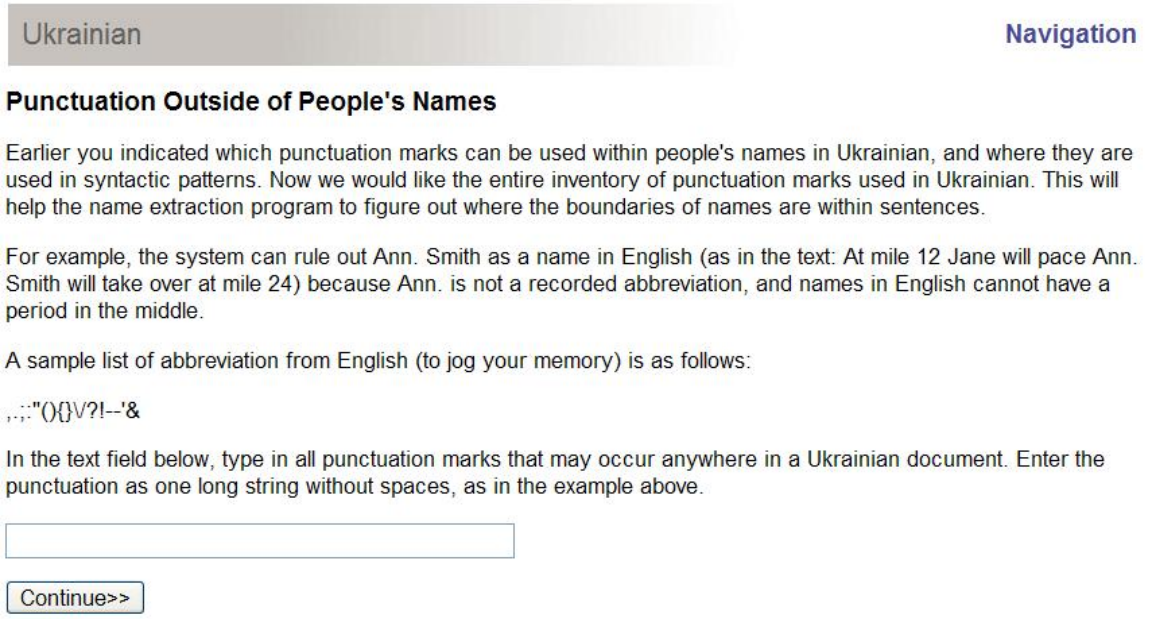
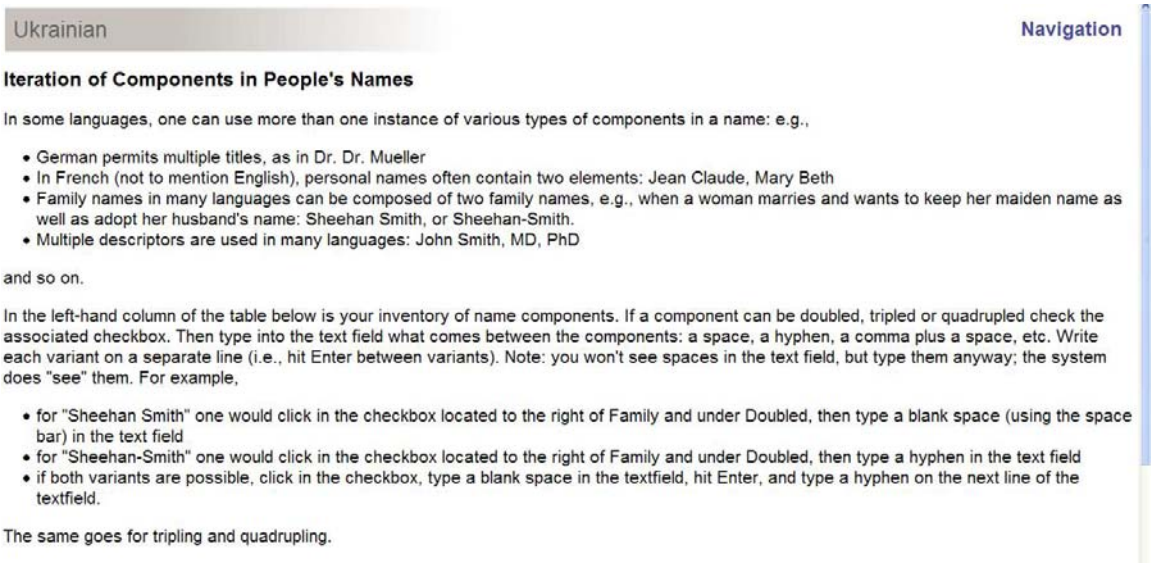
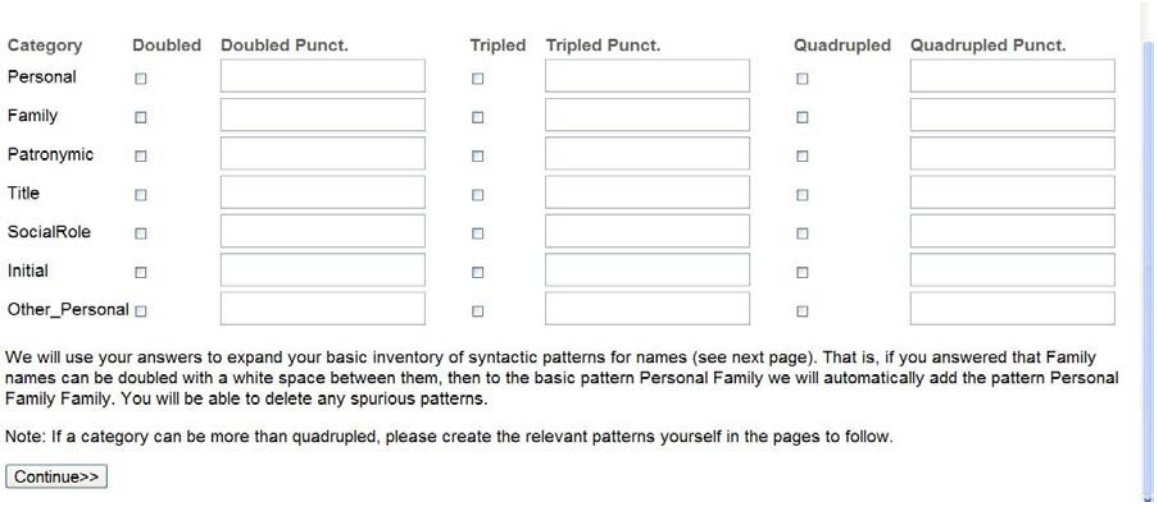


Figure 8. Elicitation of punctuation during work on Ukrainian.

Iteration of Components (Figure 9)**.** 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). The screen shot of this elicitation page is broken into two parts for reasons of space.

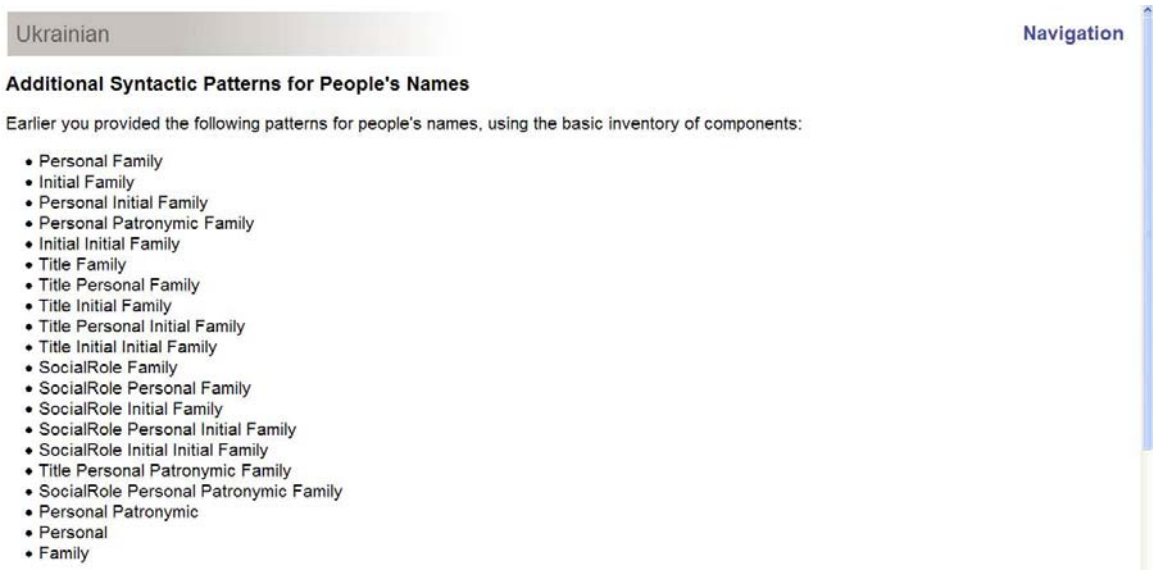




Category	Doubled	Doubled Punct.	Tripled	Tripled Punct.	Quadrupled	Quadrupled Punct.
Personal	☐		☐		☐	
Family	☐		☐		☐	
Patronymic	☐		☐		☐	
Title	☐		☐		☐	
SocialRole	☐		☐		☐	
Initial	☐		☐		☐	
Other_Personal	☐		☐		☐	

Figure 9. Elicitation of iteration of components during work on Ukrainian.

Additional Syntactic Patterns (Figure 10)**.** The user is presented with his or her current inventory of syntactic patterns and permitted to supplement it, if necessary (again, the screen shot is divided into 2 parts).



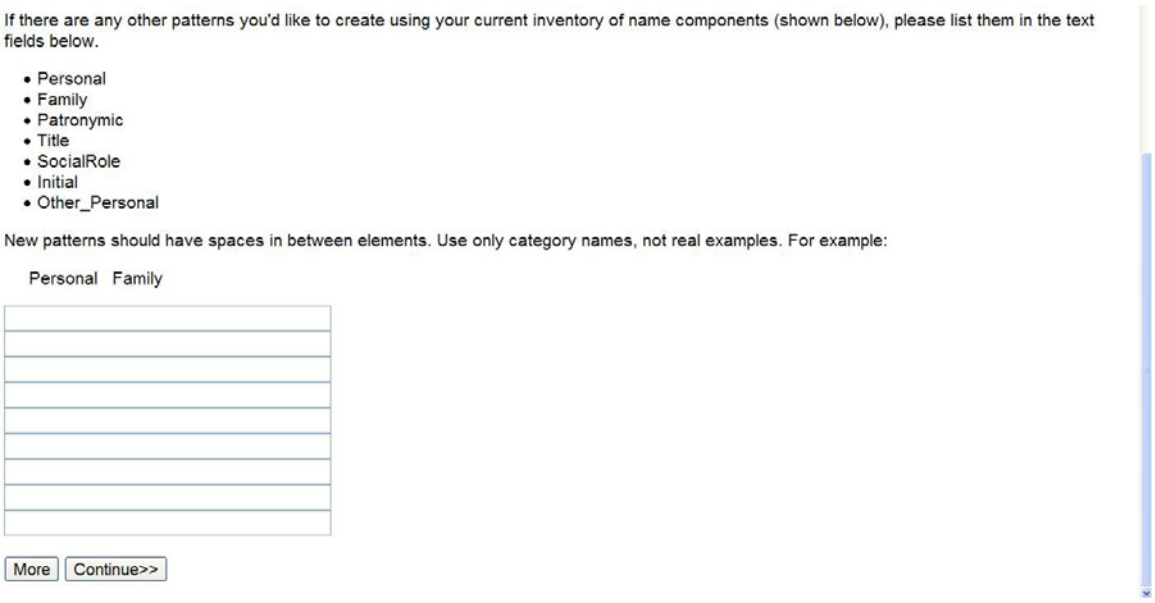


Figure 10. Elicitation of additional syntactic patterns during work on Ukrainian.

Morphological Forms, Capitalization (Figure 11)**.** The user is asked if components of names can occur in non-base morphological forms (e.g., plural, or with a case-marking apart from the citation form). The answer to this question will alert developers if they need to incorporate external morphological analysis. The elicitation of morphological analysis and the construction of a morphological analyzer for low-density languages was part of the original Boas system and can ultimately be a plug-in to this stand-alone supplement. Information about capitalization, by contrast, is used by the system.

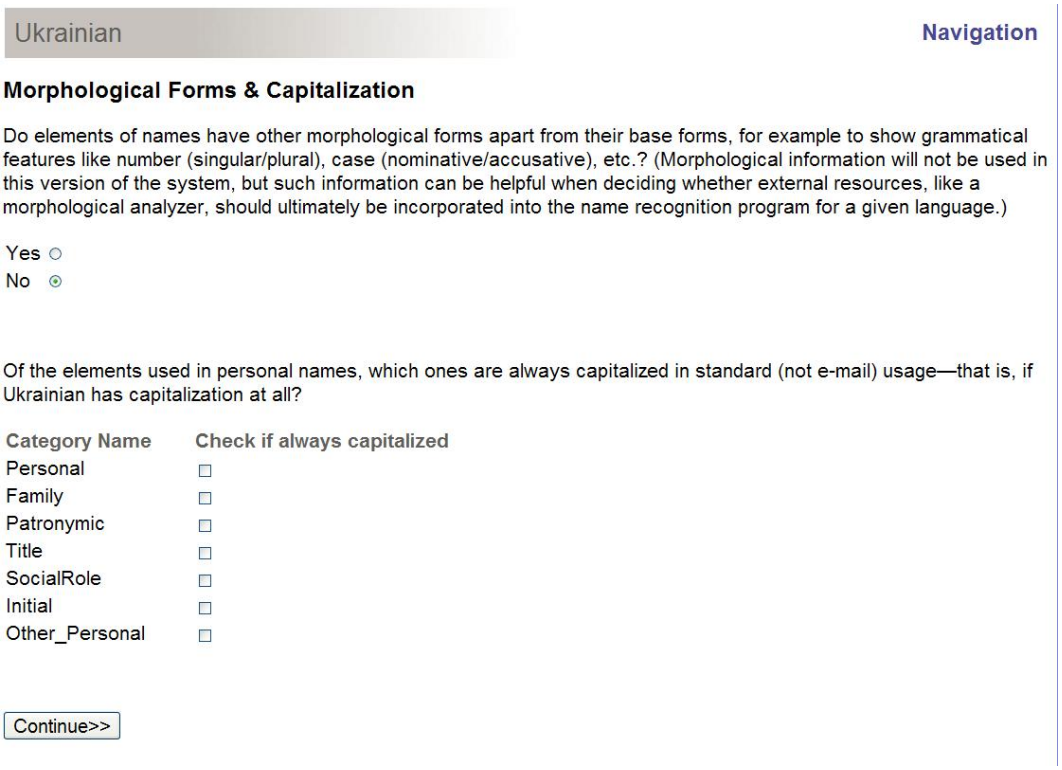


Figure 11. Elicitation of morphological forms and capitalization during work on Ukrainian.

Inventories of Titles and Social Roles (Figure 12)**.** The user is asked to translate whichever elements from our resident list of titles and professions might be useful for NER in L. For convenience, the list is loosely grouped into the categories Titles, General Professional, Military, Royalty, Political, Business, Medical, Academic, Entertainment/Communication, Family Role, Legal Role and Other (Figure 12 shows only the beginning of the list).

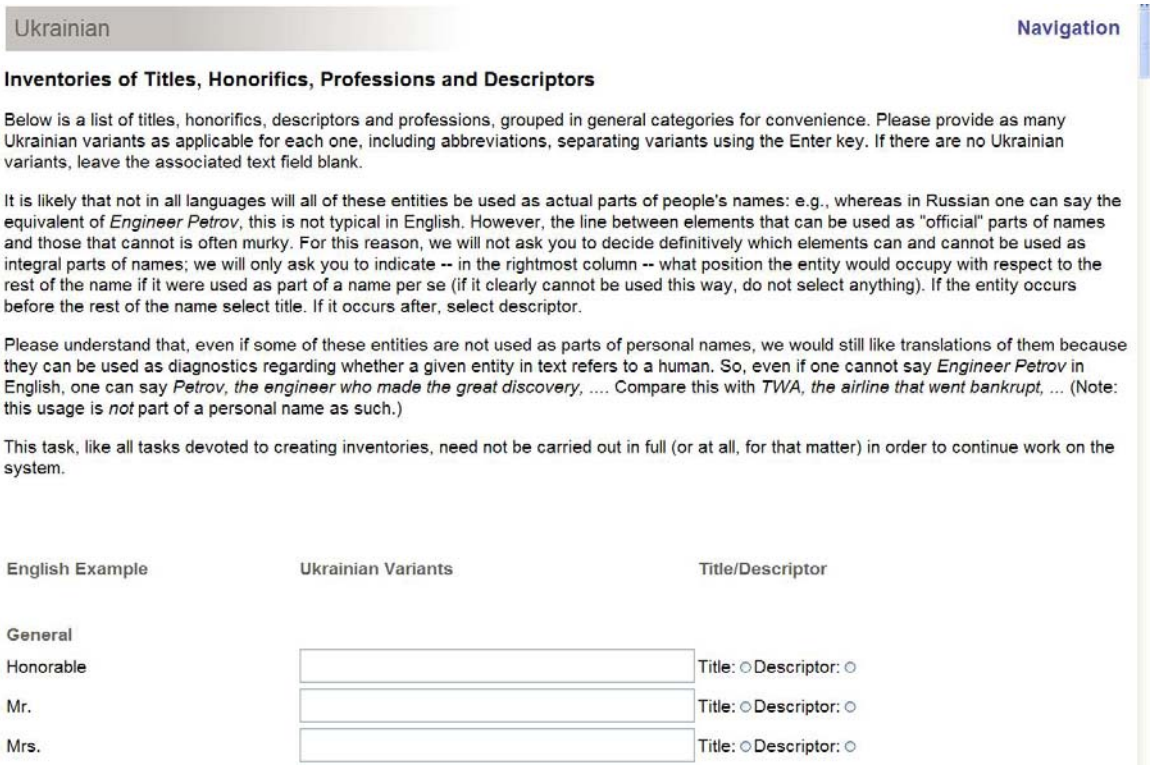


Figure 12. Elicitation of titles, etc., during work on Ukrainian.

Inventories of Other Components (Figure 13)**.** The user is asked to create at least small (twenty words, if possible) seed inventories of the other components of names in L, like Personal name, Family name, Patronymic, and so on.

Fig

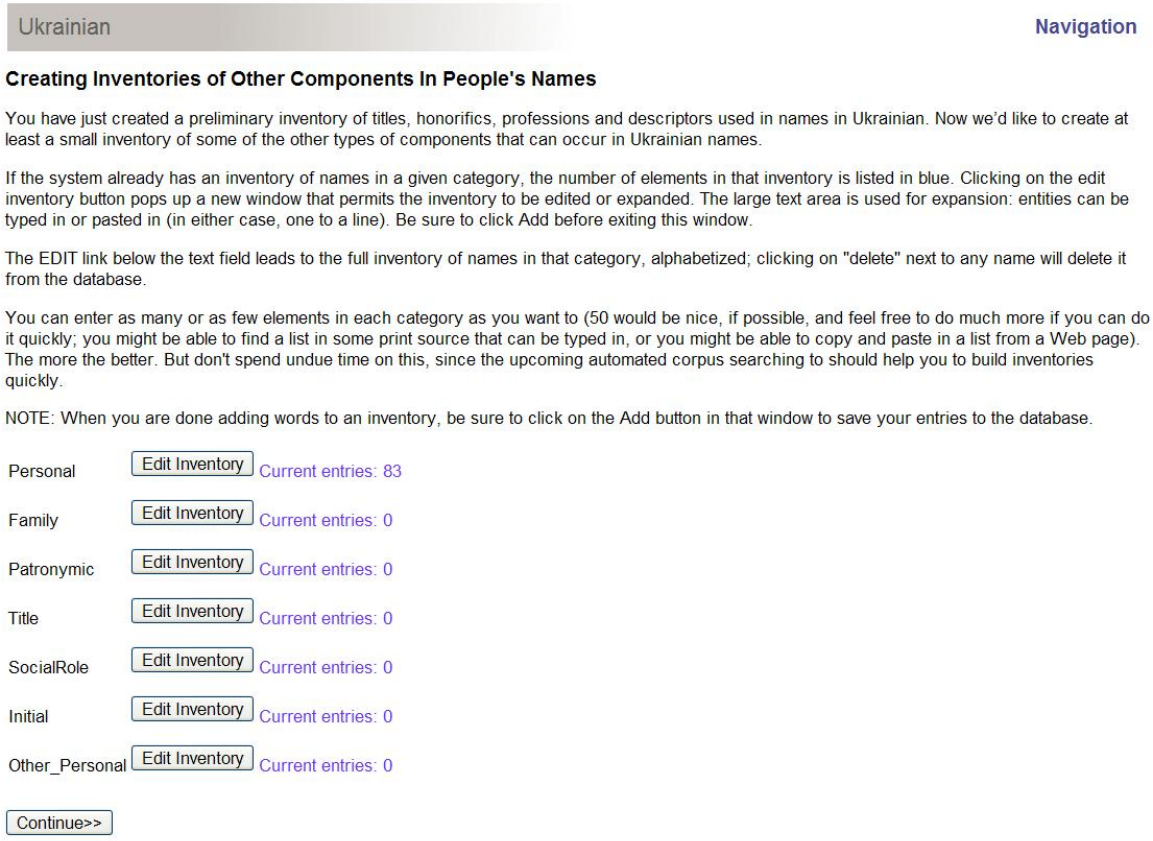


Figure 13. Elicitation of other components of people's names during work on Ukrainian.

This is done by typing words into text fields (Figure 14). If lists of components were imported at the start of work, their numbers are reflected here: e.g., for Ukrainian, Boas II contains 83 Personal names from the outset. Lists can be added to or elements can be deleted from them at any time.

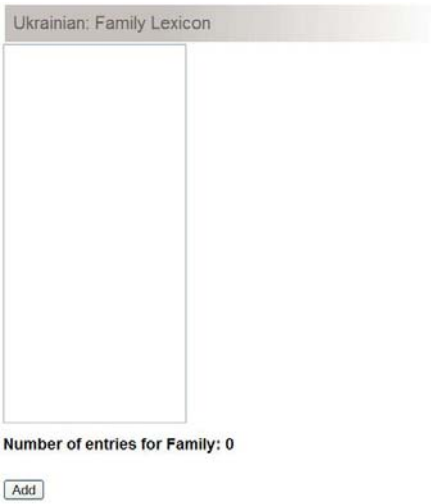


Figure 14. Text field for adding to the list of family names in Ukrainian.

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 this can lead to some missed matches (e.g., if *Washingon* were listed only as a Family name but the string Washington Erving was encountered), it is a better solution than the extensive false positives encountered when not using such a filter.

The Inventories page also includes the option of compiling a stop list: i.e., 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 concentrating on 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.

Nicknames (Figures 15 and 16)**.** The user is given the option of providing nickname equivalents for the current inventory Personal names, since such correspondences can be important for systems that carry out coreferencing of 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.

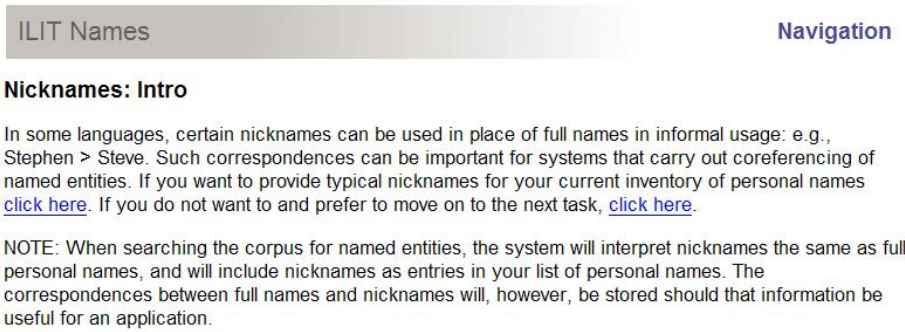


Figure 15. Introduction to the elicitation of nicknames during work on Ukrainian.

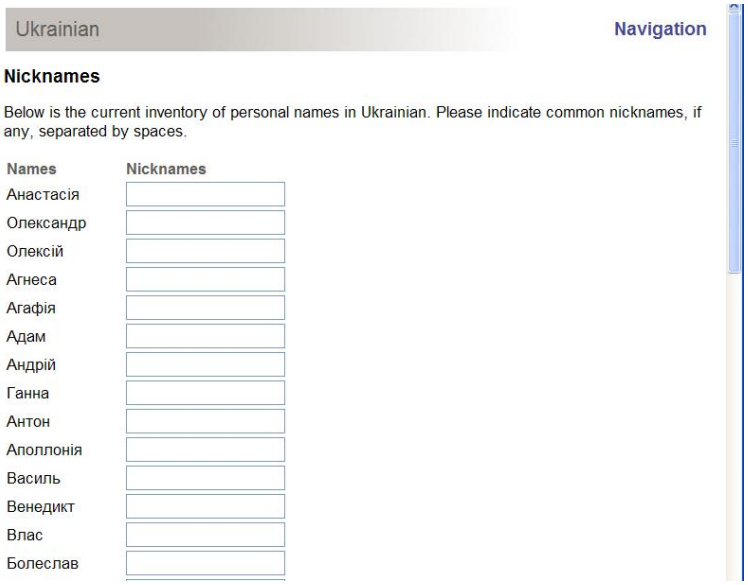


Figure 16. Providing nickname equivalents for Ukrainian personal names.

Heuristics for Components (Figure 17)**.** The user is asked to provide prefixes and/or suffixes that suggest that a given string is a certain type of personal name component (e.g., the suffix *-ovich* in Russian strongly suggests a patronymic). This information can be used to constrain the search program (see below).

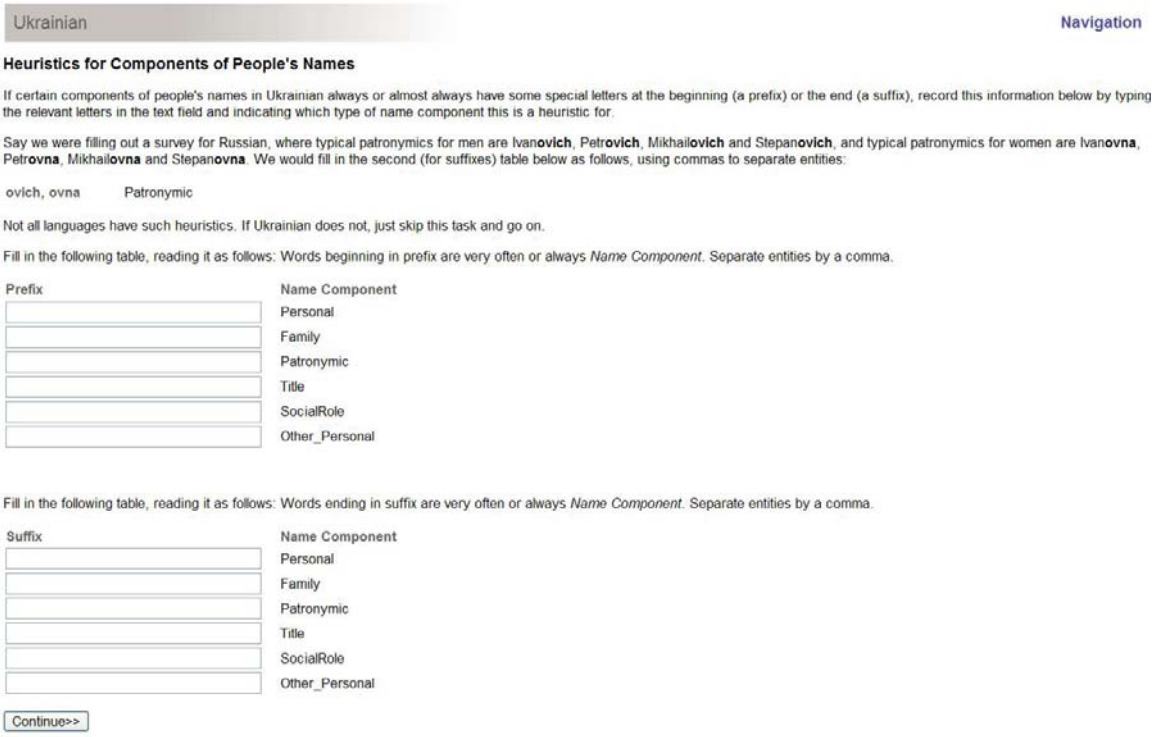


Figure 17. Eliciting heuristics for name components during work on Ukrainian.

3.2 Other Named Entities

In this section, the user is asked to provide translations for typical key words that suggest that a named entity is a *body of water, building, geological entity, publication, company name* or *event*: e.g., *X Ocean* in English is most likely an ocean, and *Y War* is most likely a war, and neither is most likely a person's name, despite the capitalization.

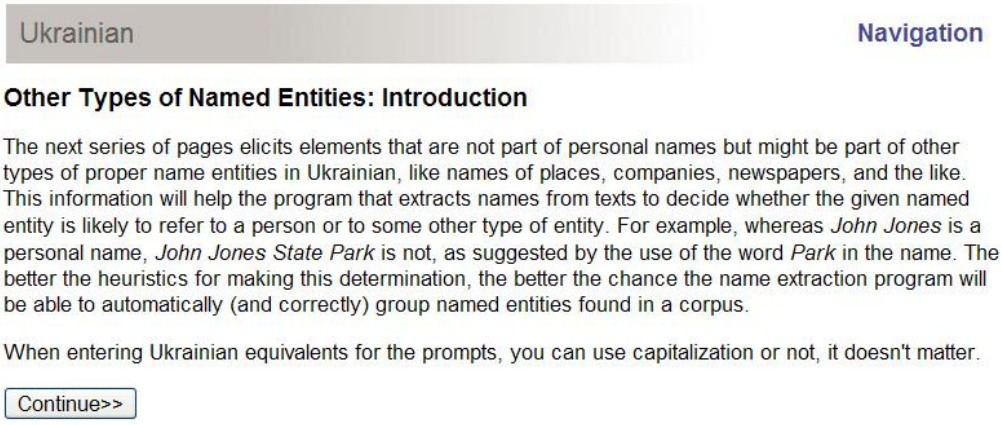


Figure 18. Introduction to the elicitation of other named entities during work on Ukrainian.

After the introduction (Figure 18) the user is asked to indicate the typical patterns in which the key word is used in a proper name: e.g., *X Ocean* and *Ocean of X* might be fine, but *Ocean X* might be impossible. The elicitation of these patterns is rather rough-grained. (As was shown in Coates-Stephens 1993, among other sources, the variety of patterns for company names is staggering. But, as we mentioned earlier, we were tasked to focus on people's names, so we are using these patterns primarily as blocking heuristics.) We show the elicitation pages for bodies of water (Figure 19) and publications (Figure 20) as examples; the elicitation of other subclasses is similar.

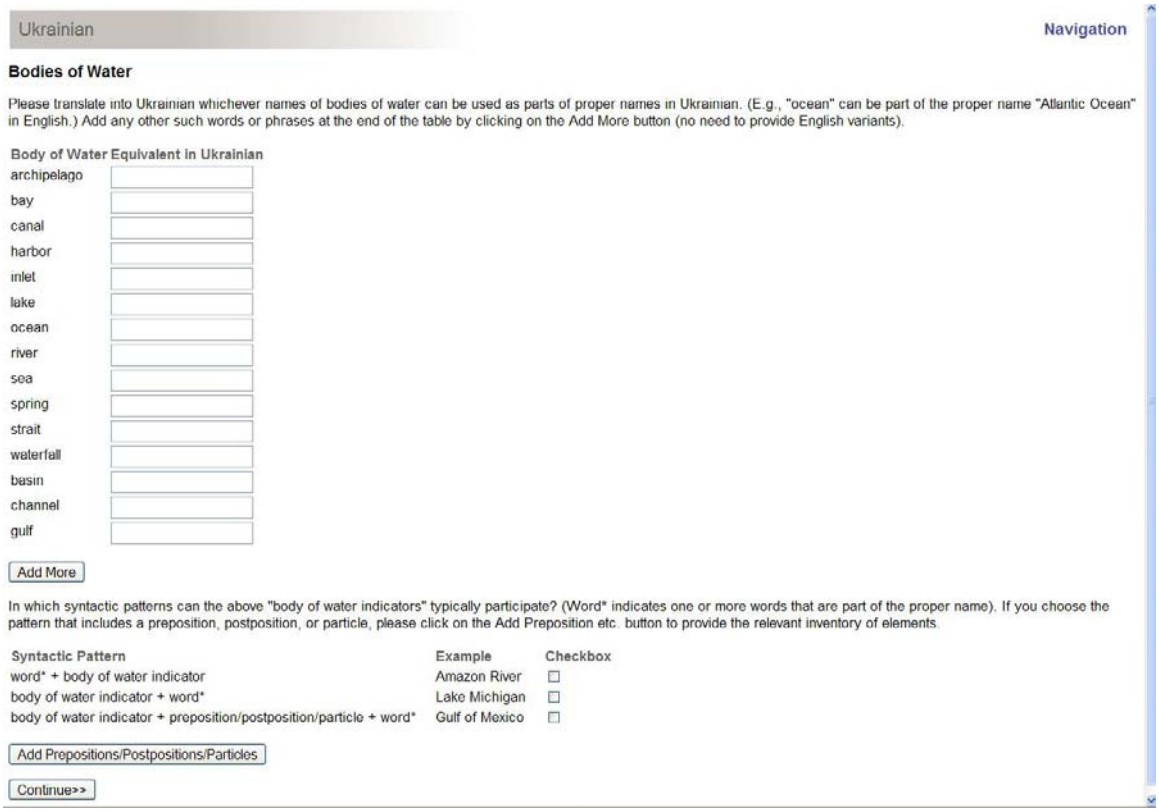


Figure 19. Elicitation of bodies of water during work on Ukrainian.

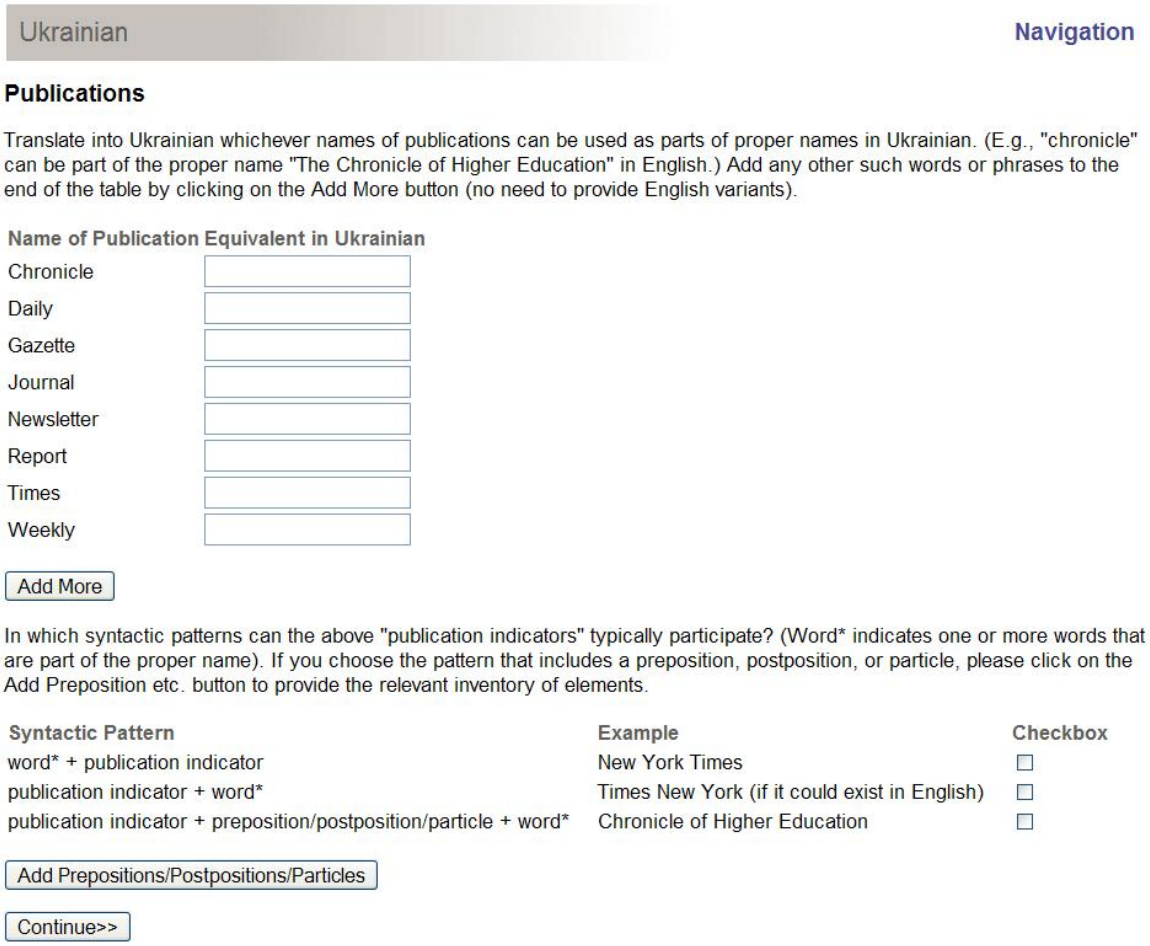


Figure 20. Elicitation of publications during work on Ukrainian.

3.3 Corpus Work

Corpus work is important in Boas II. Since this system does not involve machine learning, it will carry out NER exactly as well as it is taught to. If the user misses a syntactic pattern, that pattern will never be recognized. However, corpus testing of the nascent system is an effective method of making sure all necessary patterns are covered, all necessary stop-list words are input, all available instances of name components are recorded in their respective lists, and so forth. Corpus work proceeds in three steps.

Upload/Select a Corpus (Figure 21). The user uploads one or more corpora, following the instructions to convert them into UTF-8, then selects one to work with.

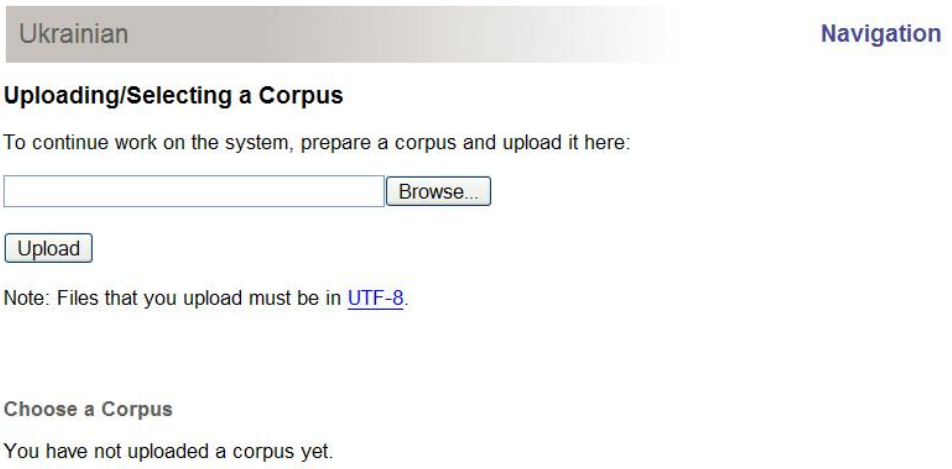


Figure 21. Uploading a corpus during work on Ukrainian.

Prepare Search. The user is asked to select one syntactic pattern at a time from the inventory 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 added. For example, say the user agrees that *Polly Jones* represents the pattern Personal Family, and say that *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 DB).

Patterns are searched for individually precisely 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 the user as he/she chose.) The inventory of patterns presented includes any necessary expansion 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*.

Search strategies are important in Boas II. We describe the related issues using the description presented to users of the system (we repeat the text rather than use a screen shot for readability; {language} is the variable for the language the person is working on).

********** *Start excerpt from Boas II* **********

Preparing to Search the Corpus

The next step in this process has two goals:

a. to test how well the system can find named entities in a corpus of {language}, and
b. 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 database. 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" will be added to the list of Family names for English.
4. Then you will launch the search again, choosing a different type of pattern to search for: e.g., **Personal Patronymic Family**. You will keep repeating this process until a) the system is finding most of the relevant syntactic patterns for names and b) the system's inventory of elements belonging to each name components is quite large or c) 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.

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 every single capitalized word 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.

********** *End excerpt from Boas II* **********

Launch Search (Figure 22)**.** The user selects from among the syntactic patterns he/she has already established for the language.

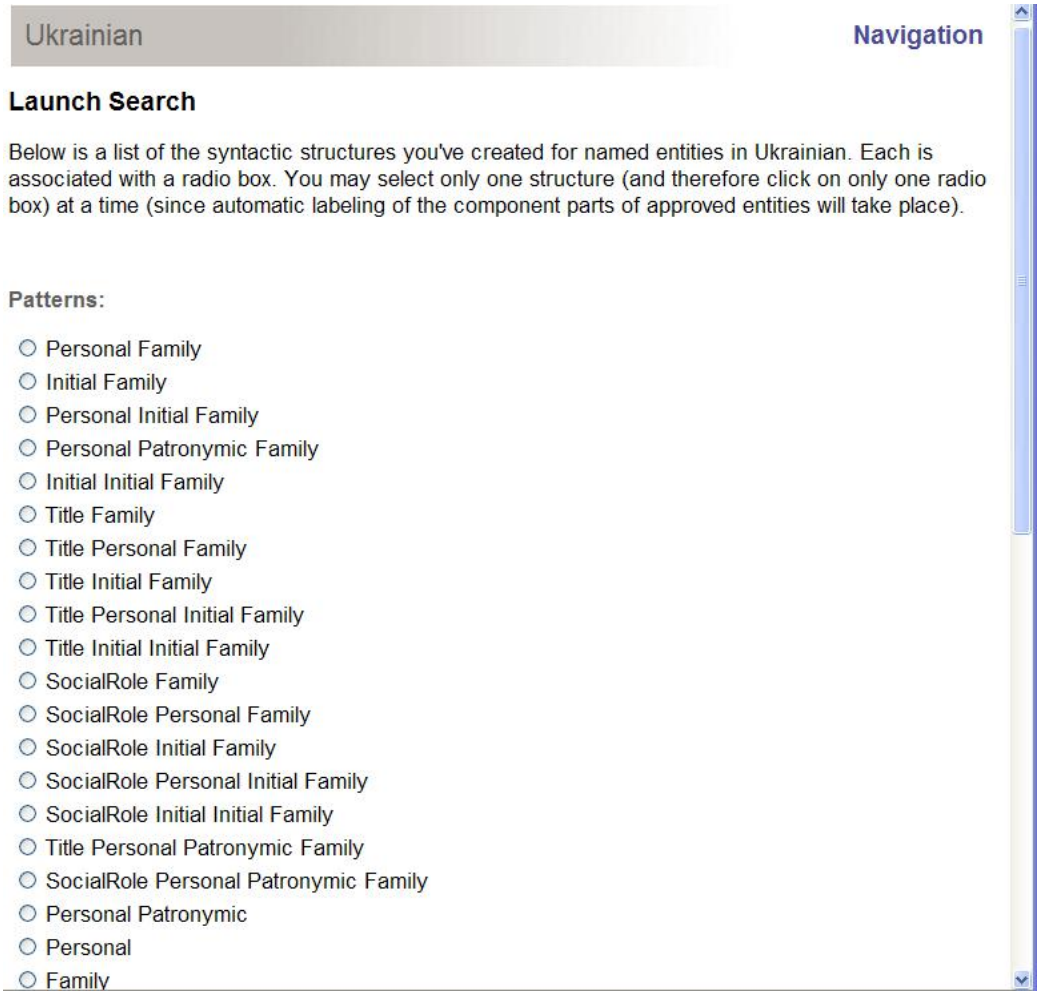


Figure 22. Selection of a pattern to search for during work on Ukrainian.

For each pattern selected, heuristics can be attached to component elements (Figure 23).

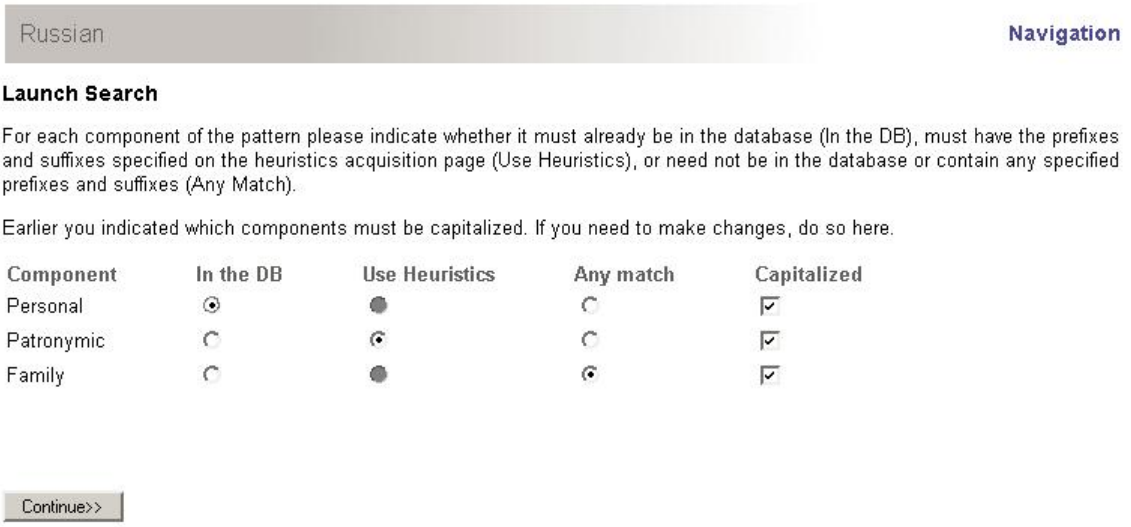


Figure 23. Specifying heuristics for components of syntactic patterns during work on Ukrainian.

The pattern selected during work on the Ukrainian profile shown in Figure 23 was *Personal Patronymic Family*. The user has selected to require that the Personal name be in the database and capitalized; that the patronymic conform to heuristics previously supplied (in this case, suffixes typical of Russian patronymics) and be capitalized; and that the Family name simply be a capitalized word. The goals of this particular search can be summarized as follows:

- seek actual instances of this pattern in a Ukrainian corpus, with each correct instance being saved to file as a complex entity;
- seek more Patronymics that conform to the listed heuristics, which will then be added to the inventory of patronymics used in later searches;
- seek more Family names, which will be added to the inventory of Family names used in later searches;
- seek errors that suggest that certain data must be added to the system: e.g., since *Personal Patronymic* is a valid naming convention in Russian, and since other words besides Family names can be capitalized (e.g., place names), element X in a candidate like *Petr Pavolvich X* need not be a Family name, as in example (1):

(1) Ivan Pavlovič Venoj zanimaetsja.
Ivan$_{NOM}$ Pavlovich$_{NOM}$ Vienna$_{INSTR}$ study$_{3SG}$
'Ivan Pavlovich is studying Vienna.'

Such a result would suggest that *Vena* 'Vienna' needs to be added to the list of TerritorialDesignations in the Russian system and, as such, be in the stop list for Personal names. By extension, this false positive should encourage the user to compile a more extensive list of geographical entities as a stop list.

The results of a search are presented back to the user in tables like the following (Figures 24 and 25), which represent searches for Initial Family and Personal Family, respectively. Ten candidates are shown per page and all are marked by default as "good" using a checkmark. Candidates that are not good must be unchecked before Continue is clicked on, since Continue both shows the next page and commits the entries on the given page to the database. (The rejection of incorrect candidates is shown in Figure 25 for candidates that mean *Great Britain* and

Two Russias, respectively.) Instead of rejecting faulty candidates outright, the user can also edit them so that they are valid, since the components are presented in editable text fields. The search can be abandoned at any time (e.g., if there are too many false positives) by navigating out of the task without clicking on Continue (e.g., by using the back button, the menu, etc.). This release of the system does not support changing the number of candidates viewed per page or choosing whether they are initially checked or unchecked.

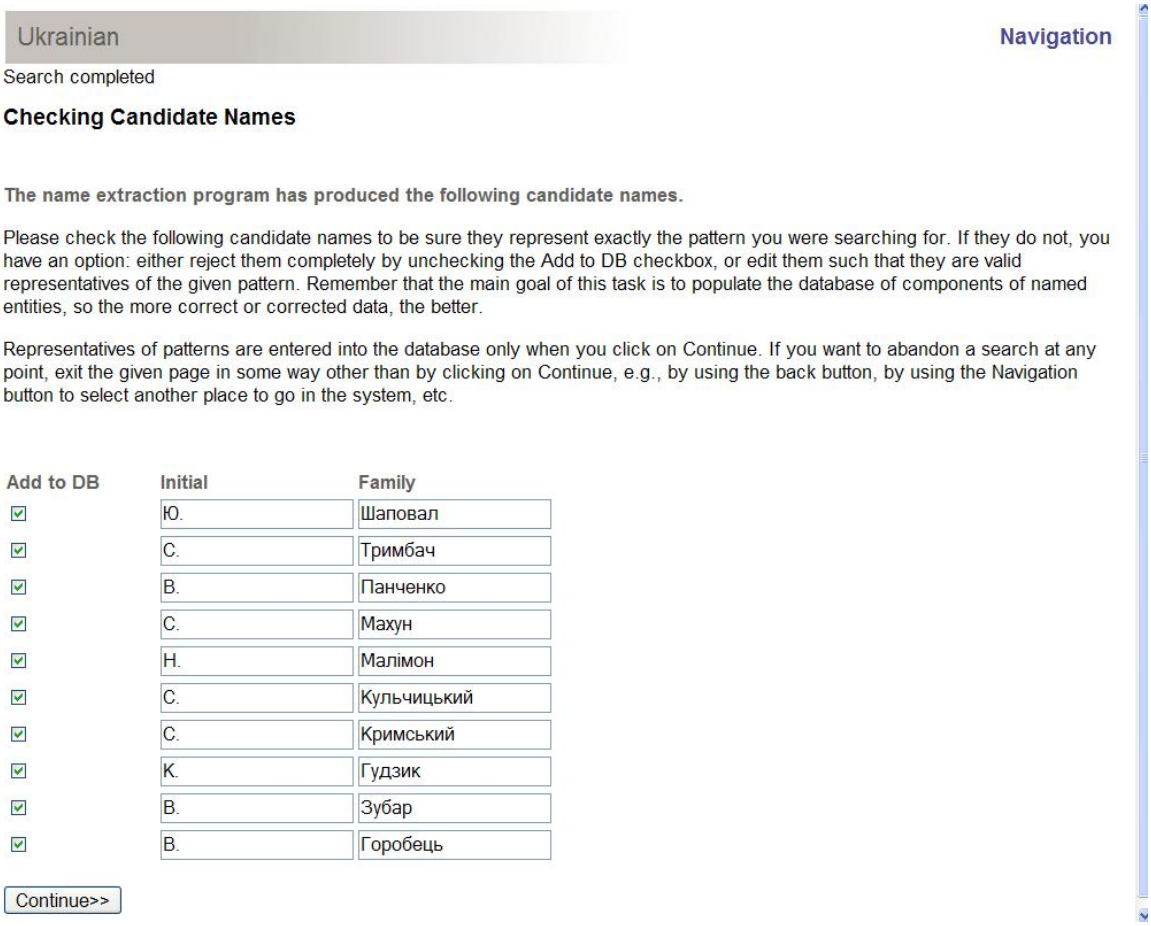


Figure 24. The first page of results for the pattern Initial Family in a Ukrainian corpus.

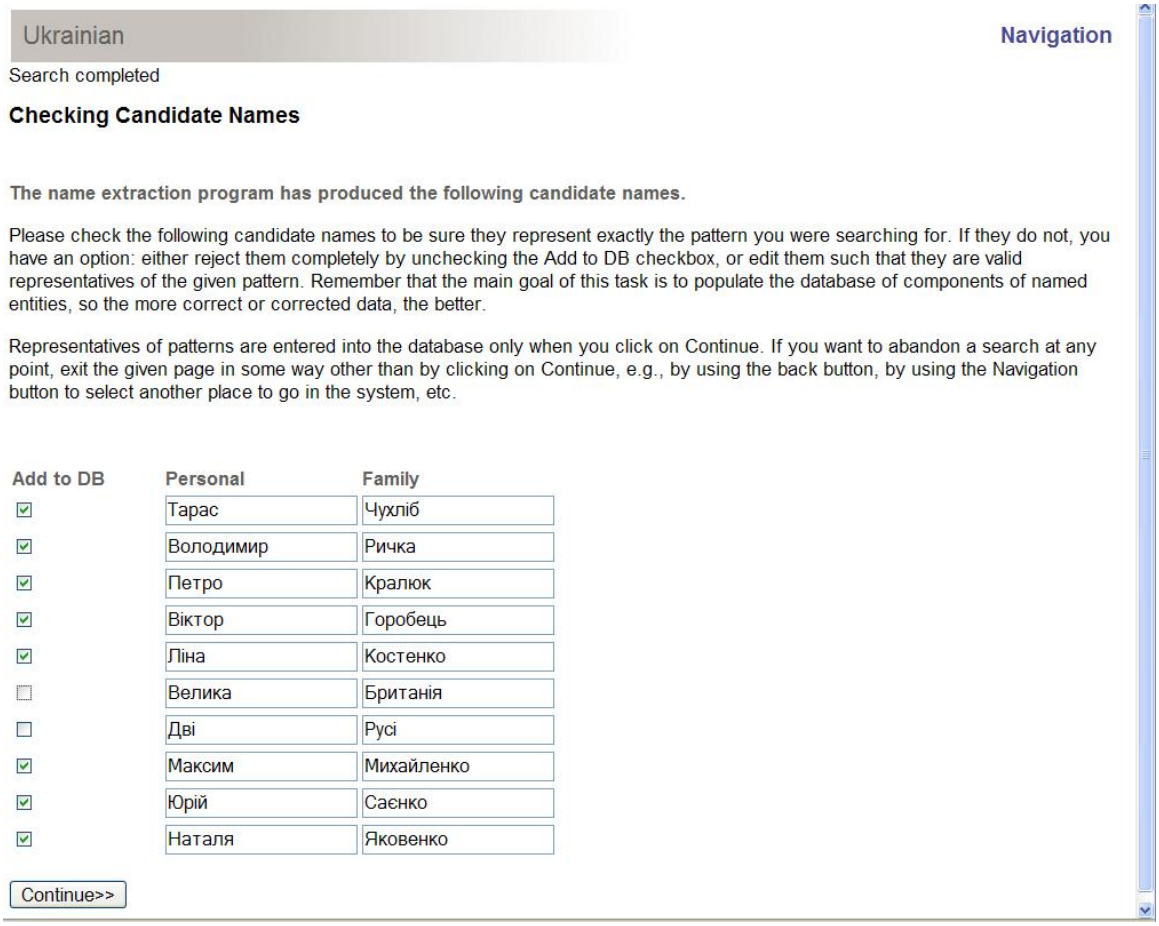


Figure 25. The first page of results for the pattern Initial Family in a Ukrainian corpus.

4. Implementation

This is a Web-based system that is worked on through a Web browser (it has been tested on Internet Explorer, Netscape, Mozilla and Firefox). The user installs the system on his/her computer and uses that computer as a local host. The software requires that the following applications be installed on the computer (which can be Linux, Windows, or Mac OSX): Apache 2.0 or newer, PHP 4.3.7 or newer (with Multibyte string support), MySQL 5 or newer, and Java 1.5.0.

The majority of this application is written in PHP, with all programming logic for the web pages being in PHP. All the user data is saved in the MySQL database. Developers' access to the data is through the MySQL server at http://localhost/phpmyadmin/. Here, one can view tables of stored data or search results, or export them to Excel or XML files. Due to problems in PHP's handling of UTF-8 regular expression strings, the search algorithm was written in Java. One positive side effect of having the search in Java is that it is relatively fast.

The information collected during elicitation is stored in tables (with the same name as the elicitation pages but with a different extension) in the subdirectory C:\php5xampp-dev\mysql\data\test: the MYI file is an index file; the MYD is the data file; and frm is the format or schema file. Below is a brief description of these tables.

langinsta table

Field	*Type*	*Info*
Id	Int – auto increment	Unique id
Lang	Int	The language code
User	Int	The code for the user
Patterns	Utf8 string	List of name patters
Components	Utf8 string	List of components for the language
Punc	Utf8 string	List of allowable punctuation
Water	Utf8 string	List of water patterns
Builds	Utf8 string	List of building patterns
Geos	Utf8 string	List of geographical patterns
Pubs	Utf8 string	List of publication patterns
Comps	Utf8 string	List of company patterns
events	Utf8 string	List of event patterns
Morph	Tiny int	Whether names have morphology

This table specifies the components, patterns, and punctuation of a language. It associates a particular user and language with a specific language instance. For example if Ann has a user code of 3 and Aniu's code is 54 we can search this table to find the particular language instance id (the id field). This instance id is used in a number of other tables.

Component Table

Field	*Type*	*Info*
id	Int – auto increment	Unique id
lang	Int	Language code (foreign key to
name	String	The name of the component (e.g., Personal)
example	String	An example of the component (e.g., Ann)
cap	Tiny int	Whether instances of the component are capitalized (0, 1)
Spec	Int	Whether the component represents punctuation (0, 1)

This table contains the components specified for all languages. The full list of components for all languages, which is displayed on the components.php page in the interface, have a lang value of -1. So, if you want to add entries to the components.php page you would add entries to this table with a lang value of -1. If you would like to see the components for a particular instance of a language (for example, Ann's version of Ainu) you'd search for the lang id associated with that language instance (see the langinsta table below).

Docwords Table

Field	*Type*	*Info*
Id	Int – auto increment	Unique id
Type	Int	Component code (1= Personal, 2=Family, 3=Matronymic, 4=Patronymic)
Word	Utf8 string	The word in the source language

Field	*Type*	*Info*
Trans	Utf8 string	Transliteration (if any)
Gender	Utf8 string	
Lang	Int	The language (foreign key to Languages)

This table stores the inventories of name components that were compiled by developers before the system's release. If you would like to see all the Arabic Personal names you would search on the lang field equaling the Arabic code (from the languages table) and type = 1.

Items Table

Field	*Type*	*Info*
Id	Int – auto increment	Unique id
Lan	Int	The language (foreign key to Languages)
Category	Int	The component (PoS) foreign key to the components table
Word	Utf8 string	
Parent	Int	Translation code (key to items table id)

This table contains all the lexical items that are used as translation prompts for things like professions, titles, company names, etc. If you would like to find all the lexical entries for Ann's instance of Ainu you would search for the lan field being equal to the lang id associated with that language instance (see the langinsta table below). Complete names (for ex., *Dr. Mary Ann Smith, Rio Grande River*) are also stored in this table.

Itter table

Field	*Type*	*Info*
Id	Int – auto increment	Unique id
User	Int	User id
Parent	Int	The component that can be iterated. Foreign key to component table
Amount	Int	How many times can component be iterated
Punc	Utf8 string	Intervening punctuation

This table stores information about whether given types of name components (like Family names) can be iterated (e.g., Mary Smith-Sheehan has an iterated Family name with an intervening hyphen).

Language table

Field	*Type*	*Info*
Id	Int – auto increment	Unique id
Name	Utf8 string	Language name

This contains a list of languages and their codes.

Patterns table

Field	*Type*	*Info*
Id	Int – auto increment	Unique id
Lang	Int	Language id
Name	Utf8	Description of pattern
Example	Utf8 string	Example of pattern
Itt	Tiny int	Whether or not the pattern was created automatically from iteration info.

The table contains the inventory of patterns of named-entity components. These patterns are displayed, as applicable, on the patterns.php page (i.e., only those patterns that use elements of the given language are displayed).

Psearch table

Field	*Type*	*Info*
Id	Int – auto increment	Unique id
User	Int	User id
Pattern	Int	Pattern to search for
Components	Utf8 string	List of components (by id number)
Done	Int	Whether search is completed

This is a table used by the search function.

Results table

Field	*Type*	*Info*
Id	Int – auto increment	Unique id
Search	Int	Id of search
Matches	Utf8 string	Match found

This is another table used by the search function.

users table

Field	*Type*	*Info*
Id	Int – auto increment	Unique id
Name	Utf8 string	User name
Email	Utf8 string	
Pass	Utf8 string	password
Lang	Int	Language id

This is a table containing information about the users. The password (pass) is encrypted.

Through the interface at http://localhost/phpmyadmin/ a user or developer can export entire tables or search results to Excel files or XML files. For example, selecting the 'test' database in phpmyadmin (which is the name we gave to the DB that stores user data for Boas II; the other databases listed there come with MySQL) then clicking on the word 'components' in the left panel will provide information on that table in the main frame. One tab in that frame is 'export'. The export page permits files to be saved as CSV for Ms Excel or XML. An excerpt from a trial system is as follows:

```
"id","id","id","id","id","id"
"1","-1","Personal","John","1","0"
"2","-1","Family","Smith","1","0"
"3","-1","Tribal","Abnaki","0","0"
"4","-1","Patronymic","Ivanovich","0","0"
"5","-1","Matronymic","Espinosa","0","0"
"6","-1","Middle","Ann","1","0"
"7","-1","Title","Mr.,Mrs.,Miss,Ms.","1","0"
"8","-1","SocialRole","Dr., Professor","0","0"
"9","-1","Descriptor","DDS, Ph.D., Jr., Sr., III","0","0"
"10","-1","Particle","von, de","0","0"
"11","-1","Initial","A.","0","0"
```

One prerequisite of corpus work is a tokenizer, which divides the text into a list of words and punctuation. During the acquisition process the user specifies a list of punctuation for the given language. During tokenization these punctuation characters are separated from words. For example, the string *Smith.* is divided into two tokens, one for *Smith* and one for the period. Using a Unicode character not associated with any natural language, the tokenizer keeps track of the attachment point of the punctuation. For example, the sequence of three tokens *Smith* . *He* does not indicate that the period attaches to the end of *Smith* and not the beginning of *He*. Adding the attachment point character (here signified by a smiley face) makes the point of attachment clear: *Smith* ☺ . *He* If a word has internal punctuation the word is split based on this punctuation. For example, if the user specifies a hyphen as a punctuation mark, the word *Kathy-Jane* is divided into three tokens and two attachment points: *Kathy,* ☺ - ☺, and *Jane.* Again, the ☺is used internally (not visible to the user) to denote points of attachment without intervening spaces. In this case it indicates that the hyphen immediately follows the *y* of *Kathy* (with no space) and immediately precedes the *J* of *Jane.* As a final example, the string *The sign said "Ann Beckwith, PhD". w*ould be rendered:

The Sign said " ☺ Ann Beckwith ☺, PhD ☺" ☺.

This tokenization step simplifies the search process.

When a search is launched, PHP saves the following user-specified information in a table in the MySQL database, indexed under an integer representing the user id:

- the corpus to search
- the pattern to search for
- for each component of the pattern, (a) whether that component needs to already be present in the dictionary, (b) whether it needs to be capitalized and (c) whether user-specified heuristics must apply.

PHP passes this user id to the Java search engine, which uses it to retrieve the relevant information from this table.

When PHP executes the Java search it passes the user name to Java. Java uses this name to extract the required information about the search from the database. In preparation for the search Java loads the lexicon, which may contain phrasal entries. The Java lexicon is stored in a hash table and contains the following information: a single word in the source language, the name of the component (i.e., *personal, family, body of water*), and, in case of phrasals, a hash table of possible continuations. Imagine we have the following bodies of water: *Red, Big, Big Green, Big Round,* and *Big Blue Ox*,

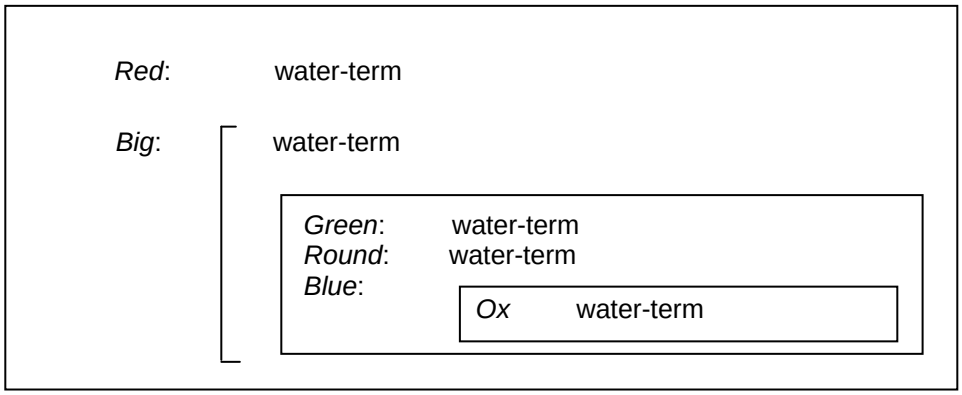


Figure 26. Hash tables in Boas II.

As shown in Figure 26, the embedded hash tables can be viewed as a simple network, and this architecture allows for efficient searching for phrasals.

For languages in which prefixes and suffixes are used as heuristics for some named entity components (see Figure 17), Java loads a lexicon of prefixes and suffixes. These affixes are stored character by character in a data structure similar to that described above for phrasals. For example, if language has the prefixes, *pre, pro, post,* and *ab* the hash table configuration would look as follows (Figure 27):

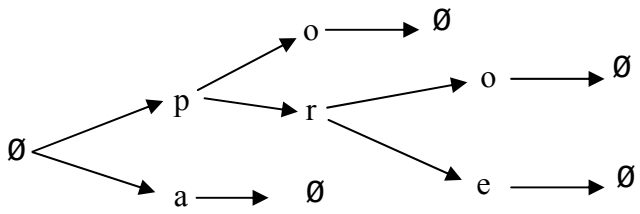


Figure 27. Data structure for prefixes and suffixes.

Java stores the search pattern as a list of component objects. The search proceeds sequentially through the tokenized text. Each component object of the search pattern attempts to match one or more tokens in the text.

Depending on search criteria specified by the user, the component object might do a lexical lookup on the token. If the lexical lookup identifies the token as the beginning of a phrasal the following token is examined. If the user has specified that a certain affix must be present, the component object runs an affix analyzer on the token using the affix lexicon described above. If the user specified that the component must be capitalized, the component object makes this check.

In our simplified approach to non-human named entities (e.g., companies and events), we permit there to be up to five words before or after the key word, with the ordering specified by the user. The iteration of non-key elements in such compound entities is indicated internally by a Kleene Star. During the search Java stores the search results in a linked list. At the end of the search Java writes the results to a database table.

5. Evaluation and Directions for Future Work

Boas II is a prototype system that could be enhanced in many ways, for example:

- by assigning probabilities to hypotheses of matches, such that users can focus on more probable analyses first. This would involve launching a corpus search for more examples using the same components to try to corroborate the analysis (in English, for example, if *Yvette Longman-Wilson* is hypothesized to be Personal + Family, the program could search for an instance of *Title* + *Longman-Wilson* to corroborate that *Longman-Wilson* is, indeed, a Family name);
- by incorporating active handling of rejected named entities, such that the same mistakes will not be made again: e.g., if a named entity that the system thinks represents Personal Family actually represents a compound place name, that place name should be listed in the gazetteer for L; this can already be done, but there is no specific interface support to make it fast and convenient;
- by incorporating more cross-linguistically relevant heuristics for certain types of name components;
- by integrating into Boas II morphological analysis capabilities from the original Boas or other sources, like the system reported in Cuzerzan and Yarowsky 1999;
- by providing support for integrating open- and closed-class lexicons for languages as stop lists into the system; this can be done manually now, and the original Boas system provided support for this type of procedure;
- by incorporating better handling of very large corpora;
- by providing more interface options to speed work. A small sample of the virtually limitless interface enhancements is as follows: a) if Personal and Middle names can be the same in a language, one should be able to copy the list from one category to the other; b) if a stoplist is valid for one pattern (e.g., 'dog' cannot be part of a person's name in English) but not for another pattern ('dog' can be part of a company name), the restrictions should be indicated and processed; c) if a person chooses to see more or fewer search results per page, or chooses to have all entities unchecked by default rather than checked, these options should be available; d) if it would save time for a user to import patterns and/or name lists from systems for other languages, this facility should be supported.

In short, optimization of the interface and the underlying search procedure were not pursued in this prototype system but should be revisited in the future.

We would also like to add to Boas II a Boas III system devoted to transliteration. Named-entity extraction and transliteration systems are related in that, ultimately, it would be very

helpful to be able to search for references to some named entity in resources in any language, which includes extraction, direct transliteration (as from Arabic to English) and indirect transliteration (as from Arabic to Russian to English). This could result in a large database of cross-indexed and type-labeled (e.g., SURNAME, CORPORATION) named entities, as well as programs to hypothesize coreference between those to be met with in the future.

Acknowledgements. This work was supported by NSF grant … This work was supported by National Science Foundation grant IIS-0242403.

Appendix 1: Full inventory of basic components for 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)

Appendix 2: Full Inventory of Basic Syntactic Patterns (examples are provided when available; when examples are not available but the pattern has been attested or suggested by informants, the symbol ~ is used in the examples column).

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	(Spanish); [note: one can also interpret this as Personal Family Family – doubled family names are elicited later; the only reason to split patronymic from matronymic is if they represent a different inventory 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	Mr. Howard H. Jones, Jr.

Descriptor	
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, singular feminine form)
Article Personal Family	la Antoni Badia (Catalan); also used in modern Greek
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 (in Tamil this

	represents: place + father's name, which we will consider a personal name unless the user wants to make a new category + personal name)
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

References Cited

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*.

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.

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

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.

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

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.

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.

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*, pages 21-39.

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.

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

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

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.

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

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.

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