A Web Application for Dialectal Arabic Text Annotation

Yassine Benajiba and Mona Diab

Center for Computational Learning Systems
Columbia University, NY, NY 10115
{ybenajiba,mdiab}@ccls.columbia.edu

Abstract

Design and implementation of an application which allows many annotators to annotate data and enter the information into a central database is not a trivial task. Such an application has to guarantee a high level of security, consistent and robust back-ups for the underlying database, and aid in increasing the speed and efficiency of the annotation by providing the annotators with intuitive GUIs. Moreover it needs to ensure that the data is stored with a minimal amount of redundancy in order to simultaneously save all the information while not losing on speed. In this paper, we describe a web application which is used to annotate many Dialectal Arabic texts. It aims at optimizing speed, accuracy and efficiency while maintaining the security and integrity of the data.

1. Introduction

Arabic is spoken by more than 300 million people in the world, most of them live in Arab countries. However the form of the spoken language varies distinctly from the written standard form. This phenomenon is referred to as diglossia (Ferguson, 1959). The spoken form is dialectal Arabic (DA) while the standard form is modern standard Arabic (MSA). MSA is the language of education in the Arab world and it is the language used in formal settings, people vary in their degree of proficiency in MSA, however it is not the native tongue of any Arab. MSA is shared across the Arab world. DA, on the other hand, is the everyday language used in spoken communication and is emerging as the form of Arabic used in web communications (social media) such as blogs, emails, chats and SMS. DA is a pervasive form of the Arabic language, especially given the ubiquity of the web.

DA varies significantly from region to region and it varies also within a single country/city depending on so many factors including education, social class, gender and religion. But of more relevance to our object of study, from a natural language processing (NLP) perspective, DA varieties vary significantly from MSA which poses a serious impediment for processing DA with tools designed for MSA. The fact is that most of the robust tools designed for the processing of Arabic to date are tailored to MSA due to the abundance of resources for that variant of Arabic. In fact, applying NLP tools designed for MSA directly to DA yields significantly lower performance (Habash et al., 2008; Benajiba et al., 2008) making it imperative to direct research to building resources and dedicated tools for DA processing.

DA lack large amounts of consistent data due to several factors: the lack of orthographic standards for the dialects, the lack of overall Arabic content on the web, let alone DA content. Accordingly there is a severe deficiency in the availability of computational annotations of DA data.

Any serious attempt at processing real Arabic has to account for the dialects. Even broadcast news which is supposed to be MSA has non-trivial DA infiltrations. In broadcast news and talk shows, for instance, speakers tend to *code switch* between MSA and DA quite frequently. Figure 1 illustrates an example taken from a talk show on Aljazeera[1] where the speaker explains the situation of women who are film makers in the Arab world. The DA word sequences are circled in red and the rest of the text is in MSA. In (Habash et al., 2008), the authors show that in broadcast conversation, DA data represents 72.3% of the text. In weblogs, the amount of DA is even higher depending on the domain/topic of discussion where the entire text could be written in DA.

Language used in social media pose a challenge for NLP tools in general in any language due the difference in genre. Social media language is more akin to speech in nature and people tend to be more loose in their writing standards. The challenge arises from the fact that the language is less controlled and more speech like where many of the textually oriented NLP techniques are tailored to processing edited text. The problem is exacerbated for Arabic writing found on the web because of the use of DA in these genres. DA writing lacks orthographic standards, on top of the other typical problems associated with web media language in general of typographical errors and lack of punctuation. Figure 2 shows a fully DA text taken from an Arabic weblog[2].

Our Cross Lingual Arabic Blog Alerts (COLABA) project aims at addressing these gaps both on the resource creation level and the building of DA processing tools. In order to achieve this goal, the very first phase consists of gathering the necessary data to model:

- Orthographic cleaning and punctuation restoration (mainly sentence splitting);
- Dialect Annotation;
- Lemma Creation;
- Morphological Profile Creation.

Across all these tasks, we have designed a new phonetic scheme to render the DA in a conventionalized internal orthographic form details of which are listed in (Diab et al., 2010b). We believe that creating a repository of

[1] http://www.aljazeera.net/

[2] $http://www.paldf.net/forum/$

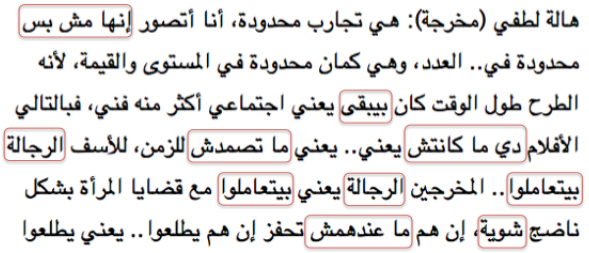

Figure 1: An illustrating example of MSA - DA code switching.

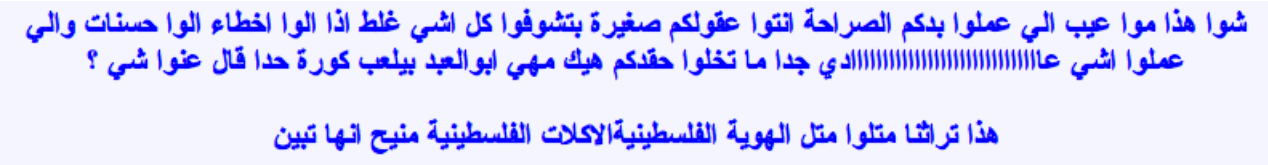

Figure 2: An illustrating example of a DA text on a weblog.

consistent annotated resources allows for the building of applications such as Information Retrieval, Information Extraction and Statistical Machine Translation on the DA data. In this project, we have targeted four Arabic Dialects, namely: Egyptian, Iraqi, Levantine, and Moroccan. And the harvested data is on the order of half a million Arabic blogs. The DA data is harvested based on manually created queries in the respective dialects as well as a list of compiled dialect specific URLs. Once the data is harvested it is automatically cleaned from metadata and the content part is prepared for manual annotation.

The application that we present in this paper, COLANN_GUI, is designed and implemented in the framework of the COLABA project.

COLANN_GUI is the interface used by the annotators to annotate the data with the relevant information. COLANN_GUI uses two different servers for its front-end and back-end components. It also allows many annotators to access the database remotely. It offers several views depending on the type of user and the annotation task assigned to an annotator at any given time. The decision to develop an annotation application in-house was taken after unsuccessfully trying to find an off-the-shelf tool which can offer the functionalities we are interested in. Some of these functionalities are:

- Task dependency management: Some of the annotation tasks are dependent on each other whereas others are completely detached. It is pretty important in our tasks to be able to manage the annotation tasks in a way to keep track of each word in each sentence and organize the information entered by the annotator efficiently. It is conceivable that the same word could have different annotations assigned by different annotators in different tasks whereas most the available tools do not have the flexibility to be tailored is such fashion; and
- Annotators' management: the tool should be able to allow the lead annotators to assign different tasks to different annotators at different times, help them trace the annotations already accomplished, and should allow them to give illustrative constructive feedback from within the tool with regards to the annotation quality.

Even though many of these annotation tools, such as GATE(Damljanovic et al., 2008; Maynard, 2008; Aswani and Gaizauskas, 2009), Annotea(Kahan et al., 2001) and MnM(Vargas-Vera et al., 2002) among others, have proven successful in serving their intended purposes, none of them was flexible enough for being tailored to the COLABA goals.

The remainder of this paper is organized as follows: We give an overview of the system in Section 2.; Section 3. illustrates the detailed functionalities of the application; Section 4. describes each of the annotation tasks handled by the application; We give further details about the database in Section 5. and finally, some future directions are shared in Section 6..

2. Overall System View

COLANN_GUI is a web application. We have chosen such a set up, in lieu of a desktop one, as it allows us to build a machine and platform independent application. Moreover, the administrator (or super user) will have to handle only one central database that is multi-user compatible. Furthermore, the COLANN_GUI is browser independent, i.e. all the scripts running in the background are completely browser independent hence allowing all the complicated operations to run on the server side only. COLANN_GUI uses PHP scripts to interact with the server database, and uses JavaScripts to increase GUI interactivity.

Safety and security are essential issues to be thought of when designing a web application. For safety considerations, we employ a subversion network (SVN) and auto-

matic back-up servers. For security considerations we organize our application in two different servers, both of which is behind several firewalls (see Figure 3).

3. COLANN_GUI: A Web Application

As an annotation tool, we have designed COLANN_GUI with three types of users in mind: Annotators, Lead Annotators, and Super User. The design structure of COLANN_GUI aims to ensure that each annotator is working on the right data at the right time. The Super User and Lead Annotator views allow for the handling of organizational tasks such as database manipulations, management of the annotators as well as control of in/out data operations.
Accordingly, each of these different views is associated with different types of permissions which connect to the application.

3.1. Super User View

The Super User has the following functionalities:

1. Create, edit and delete tables in the database
2. Create, edit and delete lead accounts
3. Create, edit and delete annotator accounts
4. Check the status of the annotation tasks for each annotator
5. Transfer the data which needs to be annotated from text files to the database
6. Generate reports and statistics on the underlying database
7. Write the annotated data into XML files

3.2. Lead Annotator View

The Lead Annotator view shares points 3 and 4 of the Super User view. In addition, this view has the following additional functionalities:

1. Assign tasks to the annotators
2. Check the annotations submitted by the annotators
3. Communicate annotation errors to the annotators
4. Create gold annotations for samples of the assignment tasks for evaluation purposes. Their annotations are saved as those of a special annotator
5. Generate inter-annotator agreement reports and other types of relevant statistics on the task and annotator levels

3.3. Annotator View

The annotator view has the following functionalities:

1. Check status of his/her own annotations
2. Annotate the assigned units of data
3. Check the overall stats of other annotators' work for comparative purposes
4. An annotator could check the speed of others (anonymously and randomized) on a specific task once they submit their own
5. View annotations shared with them by the Lead Annotator

4. Annotation Tasks

A detailed description of the annotation guidelines goes beyond the scope of this paper. The annotation guidelines are described in detail in (Diab et al., 2010b). We enumerate the different annotation tasks which our application provides. All the annotation tasks can only be performed by a user of category *Annotator* or *Lead Annotator* for the creation of the gold evaluation data. In all the tasks, the annotator is asked to either save the annotation work, or submit it. If saved they can go back and edit their annotation at a later time. Once the work is submitted, they are not allowed to go back and edit it. Moreover, the annotators always have direct access to the relevant task guidelines from the web interface by pressing on the information button provided with each task.
The annotation tasks are described briefly as follows:

1. *Typo Identification and Classification and Sentence Boundary Detection:* The annotator is presented with the raw data as it is cleaned from the meta data but as it would have been present on web. Blog data is known to have all kinds of speech effects and typos in addition to a severe lack of punctuation.
   Accordingly, the first step in content annotation is to identify the typos and have them classified and fixed, in addition have sentence boundaries identified.

   The typos include: (i) gross misspellings: it is recognized that DA has no standard orthography, however many of the words are cognates/homographs with MSA, the annotator is required to fix misspelling of such words if they are misspelled for example المساجذ *AlmsAj*, "the mosques" would be fixed and re-entered as المساجد*AlmsAjd*; (ii) speech effects: which consists of rendering words such as "Goaaaal" to "Goal"; and (iii) missing spaces. The annotator is also asked to specify the kind of typo found. Figure 4 shows a case where the annotator is fixing a "missing space" typo.
   The following step is sentence boundary detection. This step is crucial for many of the language tools which cannot handle very long sequences of text, e.g. syntactic parsers. In order to increase the speed and efficiency of the annotation, we make it possible to indicate a sentence boundary by clicking on a word in the running text. The sequence of words is simply split at that click point. The annotator can also decide to merge two sequences of words by clicking at the beginning of a line and it automatically appends the current line to the previous one. It is worth noting that all the tasks that follow depend on this step being completed. Once this task is completed, the data is sent to a coarse grained level of dialect identification (DI) pipeline described in detail in (Diab et al., 2010a). The result of this DI process is the identification of the

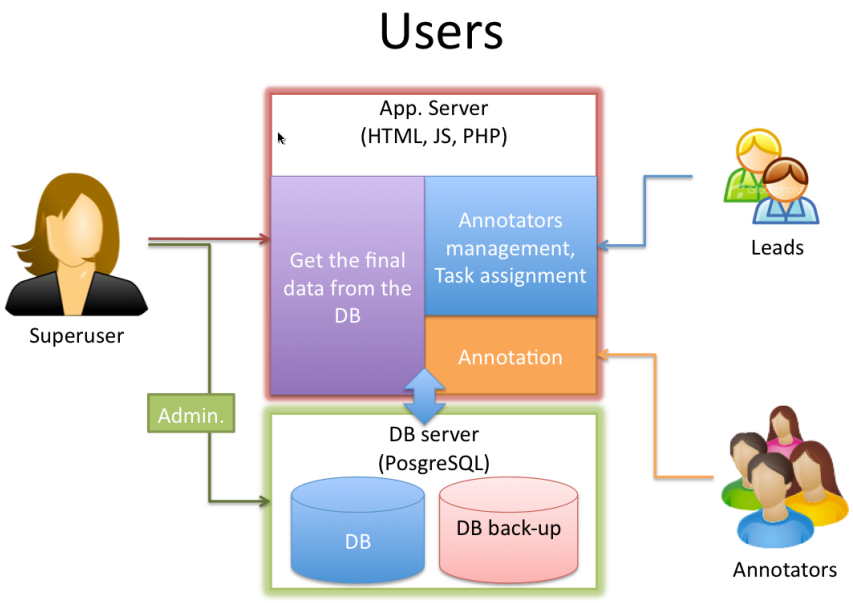

Figure 3: Servers and views organization.

Figure 4: Typo Identification and Fixing.

problem words and sequences that are not recognized by our MSA morphological analyzer, i.e. the words don't exist in our underlying dictionaries.[3]

2. *Dialect annotation:* For each word in the running text (after the content cleaning step mentioned before), the annotator is asked to specify its dialect(s) by picking from a drop down menu. Moreover they are requested to choose the word's level of dialectalness on a given scale. Finally, they are required to provide the phonetic transcription of word as specified in our guidelines on rendering DA in the COLABA Conventional Orthography (CCO).

The GUI at this point only allows the annotator to submit his/her annotation work when all the words in the text are annotated. The annotators are given the option to mark a word as unknown.

Another functionality that we have added in order to help the annotators speed up their annotation in an efficient way is a color coding system for similar words. If the annotator enters the possible dialects, relevant annotation, and the phonetic CCO transliteration for a surface word w_i. The annotated words change color

[3]It is important to note that we run the data through the morphological analyzer as opposed to matching against the underlying dictionary due to the fact the design decision we made early on that our dictionaries will have lemmas and rules associated with them rather than exhaustively listing all possible morphological forms which could easily be in the millions of entries.

to red. This allows the annotator to know which words have already been annotated by simply eye balling the words colored in red in the overall document undergoing annotation. Second, the script will look for all the words in the text which have the same surface form as w_i, i.e. all instances/occurrences of annotated w_i, and it will color each of these instances in blue. The annotator then, can simply skip annotating these words if s/he judges them to have the same annotation as the original, so it ends up being a revision rather than a new annotation. It is easy to understand how this simple change of color coding can facilitate the annotation job and increase the efficiency of the annotation process by an example. In a long Arabic blog text, frequent function words such as مش, *m$*, "not", will only need to be annotated once.

Figure 6 shows an illustrating example of the dialect annotation process via a screenshot of the task.

3. *Lemma Creation:* In this task, the annotators are asked to provide the underlying lemma forms (citation forms) for surface DA words. The lemmas constiture the dictionary entry forms in our lexical resources. The resource aims to have a large repository of DA lemmas and their MSA and English equivalents as well as DA example usages as observed in the blog data in the COLABA project. Accordingly, the annotator is provided with a surface DA word and instances of it's usage from example sentences in the blogs and they are required to provide the corresponding lemma, MSA equivalent, English equivalent, gross dialect id. Once they provide the lemma, they have to identify which example usage is associated with the lemma they created. All the lemma information is typed in using the CCO transcription scheme that COLABA specifies. It is worth noting that this task is completely independent from the Dialect Annotation task. Hence annotators could work directly on this task, i.e. after fixing typos and sentence boundaries are identified and the DI process is run.

   Accordingly, after the data undergoes the various clean up steps mentioned earlier, the data goes through the DI process as follows:

   (a) Transliterate the Arabic script of the blogs into the Buckwalter Transliteration scheme (Buckwalter, 2004) after the previous content clean up tasks of typo and sentence boundary handling. This process also identifies the foreign word character encoding if they exist in the text;

   (b) Use the DI pipeline to identify the DA words within each document;

   (c) Build a ranked list of all the surface DA words observed in the input document set based on their frequency of occurrence, while associating each surface word with the sentences in which it occurred in the document collection;

   Thus, we have grouped the DA words by surface form and used them as key entries in our database allowing us the ability to access them easily with their recurrent examples which are in turn identified uniquely by sentence number and document number. For instance, let us consider all the sentences where the surface word مركبه, *mrkbh*, appears in our data. For illustration in this paper, we provide the English translation and the Buckwalter transliteration, however in the actual interface the annotators only see the surface DA word and associated examples in Arabic script as they occur in the data, but after being cleaned up from meta data, html mark up, typos are fixed and sentence boundaries identified.

ها ها ها ها لو يعطوني بليار والله **مركبه** يماااااااه ...
Buckwalter Transliteration: *hA hA hA hA lw yEtwny blyAr wAllh mrkbh ymAAAAAAAAh ...*
English Gloss: hahaha even if they give me a billion I wouldn't **ride it** muuuuuum
...انجرف **مركبه** الى وادي ذيبان بسبّ ارتفاع
Buckwalter Transliteration: *A*njrf mrkbh AlY wAdy *ybAn bsbb ArtfAE ...
English Gloss: his boat drifted to the Dhibane valley because of the increase of the level ...
ده سامي مركبه فيلبلكونه
Buckwalter Transliteration: *d*h sAmy mrkbh fylblkwnh
English Gloss: Samy has it **s**et up in the balcony

   These sentences are shown to the annotator and s/he is asked to identify the number of lemmas for this surface word. For instance, in the second example, we find sentences where *mrkbh* appears as "his boat" and in the first example it appears as "ride". Accordingly in these examples, the annotator should indicate that there are three different lemmas for the surface form *mrkbh* rendered in CCO transliteration scheme as *rikib*, *markib*, and *merakib*, respectively.

   Figure 5 shows an illustrating example.

4. *Morphological Profile Creation:* Only for those words which have been already annotated with the lemma information in the previous step do we proceed for further annotation. For those lemmas in the database already, we add more detailed morphological information. The annotator is shown one lemma at a time with a set of example sentences where the surface form of the lemma is used. Thereafter the annotator is asked to select a part of speech tag (POS-tag). Figure 7 shows that when a POS-tag is selected the interface shows the type of information required accordingly.

In all these tasks, the application is always keeping track of the time that took each annotator for each task unit. Such information is necessary to compare speed and efficiency among annotators and also for the annotators themselves to be able to compare themselves to the best and the worst across a task.

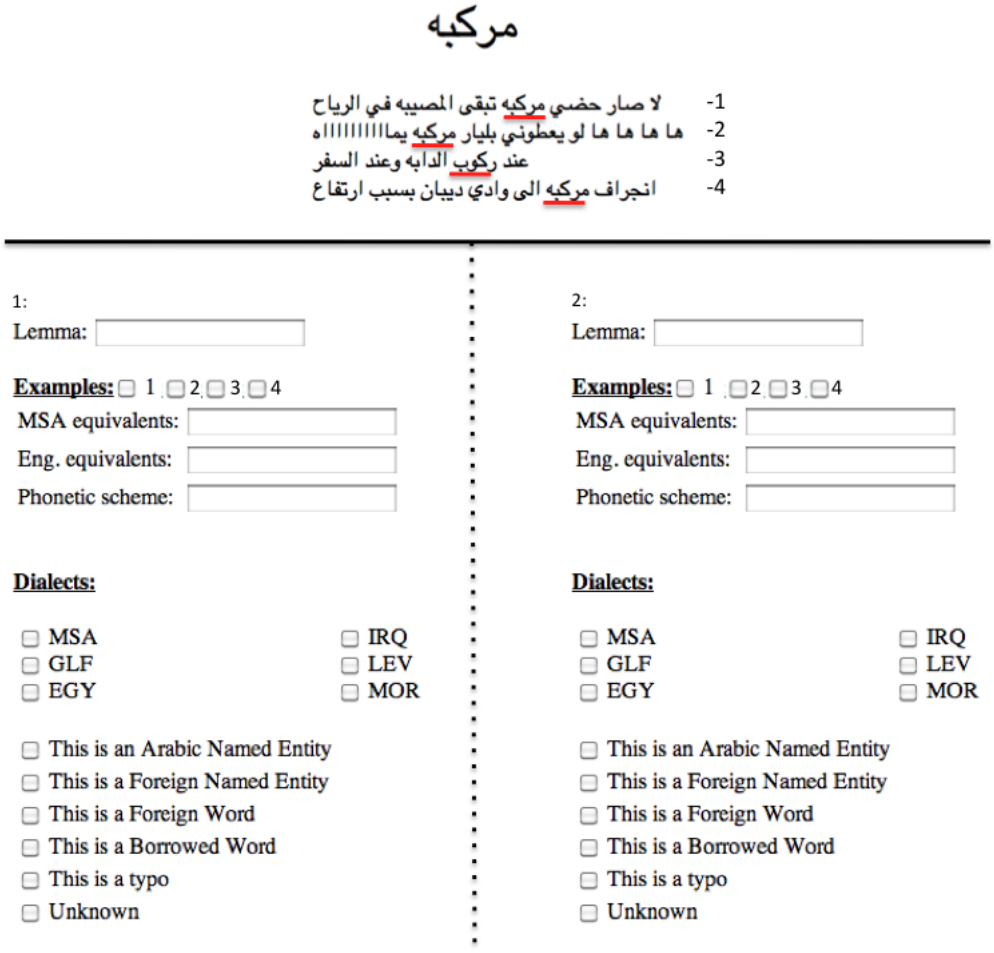

Figure 5: Illustrating example of lemma creation.

5. The Database

For this application we need a sophisticated database. We use the freely available `Postgresql` management system[4]. The database system is able to save all the annotations which we have described in Section 3.. But it also holds other information concerning the users and their annotation times. Our database contains 22 relational tables, which can be split into the following categories:

- *Basic information*: We have created a number of tables which save basic information that is necessary for all the tasks. By saving such information in the database our application becomes very easy to update and maintain. For instance, if we decide to add a new POS-tag we just have to add it in the appropriate table.
- *Annotation*: These tables are the core of the database. For each of the annotation tasks described in Section 4. it saves all the information entered by the annotator while keeping the redundancy of the information at a minimum. For instance, for each sentence in the data, we want to save all the information entered about the dialect, the lemmas and the morphological information of the dialectal words while saving only one instance of the actual sentence in only one table and relate all the annotation records to it. Only by doing so are we able to save information about millions of words in the database while keeping it easily and efficiently accessible. Finally, we also save the time (in seconds) taken by the annotators to complete the annotation tasks. This information is necessary for inter-annotator speed comparison.
- *Assignments*: These tables hold information about how many task assignments have been assigned to each annotator and how many of them have already been annotated and submitted. This is directly related to the assignment task of the lead annotators described in Subsection 3.2..
- *Users Permissions*: When a user is created, we assign a certain category to her/him. This information is used by all the scripts to decide on user privileges.
- *Connection*: Whenever a user is connected, this information is communicated to the database. By doing so we are able to deny a user to connect from two machines at the same time.

As mentioned in Section 1., our database is located on a separate server from the web server. This web server can

[4] http://www.postgresql.org/

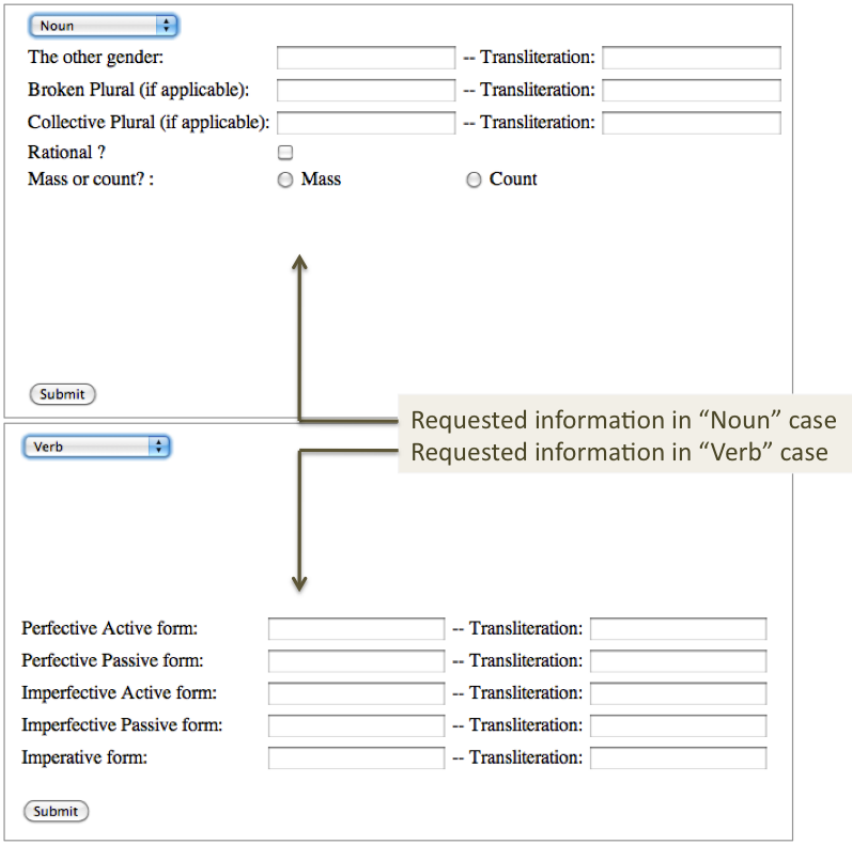

Figure 7: Illustrating example of the requested information when an annotator chooses the POS-tag Noun or Verb.

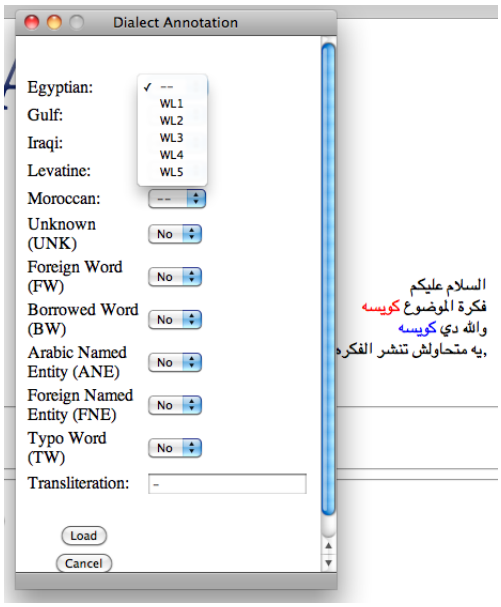

Figure 6: Illustrating example of dialect annotation.

send requests to the database server through an `ssh` tunnel which helps forward services between the two servers with encrypted data.[5]

6. Future Work

We are constantly updating our interface incorporating feedback from the annotators and lead annotators on the various tasks. The data that is annotated using our application is intended to build efficient models of four different dialects that cover all the major Arabic dialects. The models will be useful for several NLP applications:

- Automatic spelling correction;
- Automatic sentence boundary detection;
- Automatic dialect identification and annotation;
- Lemmatization and POS-tagging;
- Information Retrieval and Advanced search;
- Named Entity, foreign words and borrowed words detection;

However, it is not possible to aim at such advanced applications without a consistent annotation where the efficiency of the application which we describe in this paper plays a pivotal role.

Acknowledgments

This work has been funded by ACXIOM Corporation.

[5] http://www.ssh.com/support/documentation/online/ssh/adminguide/32/Port_Forwarding.html

7. References

N. Aswani and R. Gaizauskas. 2009. Evolving a general framework for text alignment: Case studies with two south asian languages. In *Proceedings of the International Conference on Machine Translation: Twenty-Five Years On*.

Y. Benajiba, M. Diab, and P. Rosso. 2008. Arabic named entity recognition using optimized feature sets. In *Proceedings of EMNLP'08*, pages 284–293.

T. Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Cat alog No.: LDC2004L02, ISBN 1-58563-324-0.

D. Damljanovic, V. Tablan, and K. Bontcheva. 2008. A Text-based Query Interface to owl Ontologies. In *Proceedings of the 6th Language Resources and Evaluation Conference (LREC)*.

M. Diab, N. Habash, O. Rambow, M. AlTantawy, and Y. Benajiba. 2010a. COLABA: Arabic Dialect Annotation and Processing. In *Proceedings of the Language Resources (LRs) and Human Language Technologies (HLT) for Semitic Languages at LREC*.

Mona Diab, Nizar Habash, Reem Faraj, and May Ahmar. 2010b. Guidelines for the Annotation of Dialectal Arabic. In *Proceedings of the Language Resources (LRs) and Human Language Technologies (HLT) for Semitic Languages at LREC*.

C. A. Ferguson. 1959. Diglossia. *Word*, 15:325–340.

N. Habash, O. Rambow, M. Diab, and R. Kanjawi-Faraj. 2008. Guidelines for Annotation of Arabic Dialectness. In *Proceedings of the LREC Workshop on HLT & NLP within the Arabic world*.

J. Kahan, M.R. Koivunen, E. Prud'Hommeaux, and R.R. Swick. 2001. Annotea: an open rdf infrastructure for shared web annotations. In *Proceedings of the WWW10 Conference*.

D. Maynard. 2008. Benchmarking textual annotation tools for the semantic web. In *Proceedings of the 6th Language Resources and Evaluation Conference (LREC)*.

M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. 2002. Mnm: Ontology driven semi-automatic and automatic support for semantic markup. In *Proceedings of the 13th International Conference on Knowledge Engineering and Management (EKAW)*.