A Multi-Lingual Web Service for Drug Side-Effect Data

Steven D. Bedrick, BA1, Alejandro Mauro, MD2
1Oregon Health & Science University, Portland, OR, USA; 2Hospital Italiano de Buenos Aires, Argentina

Abstract

In this paper, we describe a system that provides drug side-effect data for use as a component in service-oriented architectures. Our system uses “Web 2.0” techniques to collect data from a variety of public sources, and can provide its output in a variety of human languages (e.g. Spanish and Arabic). To demonstrate our tool’s versatility and the ease with which it may be integrated into larger systems, we present several front-ends that use our system, including SMS (“text message”), “instant messenger”, and iPhone interfaces. We enlisted a panel of Argentinean clinicians to review and rate the quality of our system’s Spanish-language output in order to investigate whether freely-available general-purpose machine translation technology (Google’s translation API) is adequate for consumer medical applications. Our raters found that Google’s translation quality varied greatly among drugs, and we conclude that it is better used as a starting point than as a complete translation solution.

Introduction

Today, developing a medical application is almost invariably an exercise in data integration. Modern applications must access, retrieve, and combine data from a wide variety of sources and in a similarly wide variety of formats. In the larger world of software development, there is an increased recognition of the importance (and potential benefits) of creating applications that are expressly intended to be used in combination with other systems. For a demonstration of this, we need look no further than the growing popularity and utility of so-called “Web 2.0” services such as the photo-sharing website Flickr, both of which are examples of systems which are designed to be easy to embed within other web pages or applications. Web 2.0 is a software design philosophy that emphasizes light-weight programming models, integration of disparate resources via the Web, and building social features into applications.

The number and variety of integration-friendly data sources and tools available to bioinformatics application developers has exploded in recent years. Researchers in public health have also been experimenting with Web 2.0 techniques for data integration with promising results. In medicine, there has been much talk in recent years about potential medical and professional benefits of the social aspects of Web 2.0. There have also been several significant clinical software tools developed following the Web 2.0 model, notably including the newest version of the National Library of Medicine’s UMLS Knowledge Source Server, (see Cheung, et al’s review for an excellent overview of the subject).

This paper describes one such tool: a web service that provides drug side-effect information from a curated public source in a machine-consumable format. Our system is designed to integrate seamlessly into other applications, services, or tools. These tools could include decision-support applications, consumer information websites, or pharmacy systems, to name only a few. Our system’s data comes from the National Library of Medicine’s MedlinePlus consumer health information website.

An important secondary goal of our system is that it provide multi-lingual functionality. Internet users in non-English speaking countries often search for and prefer to access medical information in their native languages; our system is designed to enable developers of non-English applications to access drug side-effect data originally written in English. Currently, our system depends on Google’s public translation tools, and is therefore limited by their capabilities and performance. However, as we describe below, our system is designed such that other translation systems may be used with minimal effort. This paper includes a discussion of a preliminary evaluation of the English–Spanish translations that Google provides for our system.

Methods

Back-end System Design

The back-end architecture of our system is illustrated in Figure 1. The application is written in Ruby, and connects to two different Internet services to obtain its data: MedlinePlus, and Google’s Language API (Google Inc., Mountain View, CA).

In its current form, the back end consists of two tiers: a library tier and a web tier. The library tier consists of three main modules: one to resolve arbitrary input strings into MedlinePlus drug identifiers; another to connect to MedlinePlus, retrieve a drug’s data, and parse out its side-effects; and a third to connect to Google to provide translation services.

The library tier’s modules each access web-based resources, although they do so in different ways. MedlinePlus does not provide machine-readable data feeds of its contents, so the modules that connect to the NLM resources must use the an HTML parsing library such as Nokogiri to “screen-scrape” the raw HTML found in the relevant websites. This approach, while effective, is very

1http://www.flickr.com
2http://delicious.com/
brittle: small changes in the layout of the source web pages will cause the parser to fail. Our system therefore includes a full suite of automated unit tests for each NLM-dependent module. While this will not prevent breakage should the NLM change their page layouts, our test-driven approach can and does ensure that we (the system’s developers and maintainers) will discover such breakage quickly and respond appropriately.

Unlike the NLM, Google generally provides machine-readable interfaces to its various resources, including its translation system: input to the system comes in the form of HTTP request parameters, and output comes in the form of a JSON-encoded HTTP response. JSON (the Javascript Object Notation format) is a human-readable serialization format for simple data structures, and is very commonly used by light-weight web services that need to communicate across programming languages and operating systems. Google’s translation API currently supports approximately 40 different languages, although experience suggests that the quality of translation varies from language to language.

The information flow is as follows: a program invokes the library, and provides a string containing a drug name along with a language identifier specifying the desired language of the results. The library attempts to resolve the string to a normalized drug name, and then retrieves the side-effect information for that drug from MedlinePlus. The result of a successful retrieval is a data structure containing a normalized drug name, some number of lists of side-effects, and any special hazard warnings that the drug’s MedlinePlus article may contain. MedlinePlus groups drug side-effects according to the suggested patient response (e.g., "mention to your doctor", “go to hospital immediately”, etc.); each side-effect list in the web tier’s output contains this descriptive sentence as well as the associated side-effects (see Figure 2). If necessary, the library then invokes Google’s translation system to translate the side-effect lists into the specified language. Finally, the resulting data structure is returned to the calling program.

The back-end’s web tier provides an extremely simple wrapper around the library tier’s functionality which any number of front-end clients may connect to via the Hyper-Text Transfer Protocol (HTTP). It was built using the Ramaze web framework, and accepts as input a drug name and an optional target language. The web tier’s output is simply a JSON-encoded representation of the data structure returned by the library tier. While it is technically a “web service”, it follows the light-weight Representational State Transfer (REST) design philosophy rather than the heavier-weight SOAP or XML-RPC approaches to web service development. As such, it is extremely easy to integrate and use as a component within other applications, as described below.

Evaluation

We evaluated two aspects of our system: the flexibility of its back-end, and the quality of the translations it produced. We evaluated our back-end system’s flexibility by building several front-end interfaces using a variety of software platforms, and enlisted the assistance of human raters to assess the quality of our system’s translated side-effect data.

Front-end Interfaces

To demonstrate our system’s versatility, we have constructed a variety of front-end interfaces to access its functionality. In this paper, we describe three: “Instant Message” (Jabber/XMPP), Short Message Service (SMS, or “text message”), and an iPhone/Web application. As previously mentioned, our back-end system is accessible via the Web and sends its output as JSON-encoded data. Virtually every modern programming environment and language contains functionality to load data via HTTP, and most contain tools to parse JSON-encoded data. We were therefore able to write several front-end interfaces to our system in a short period of time and with minimal programmer effort. These interfaces ranged in complexity from trivial clients implemented in a very few lines...
of JavaScript to more robust clients that involved several hundred lines of Ruby code and that interacted with several different remote services in addition to our own.

The “Instant Message” (IM) interface is a software program that acts as a simulated user (a “bot”) on a widely-used IM network (in our case, the Jabber/XMPP network, also known as “Google Chat”). Users of the network may add it to their contact lists and interact with it just as though it were a human user with whom they were “chatting”. To use our interface, a user must simply send an instant message containing a command, a drug name, and an optional language specifier. For example, to retrieve the side-effects for Mefloquine, a user would send the following message: side mefloquine. In this example, “side” is the command, and “mefloquine” is the drug name. The IM system then invokes the back-end system described above, and sends the results to the user as instant messages. Should the user desire to have Mefloquine’s side-effects translated into their own language, they need only include a standard two-letter ISO language code in their request. The command “side ar mefloquine”, for example, would retrieve Mefloquine’s side-effects in Arabic (specified using the “ar” language code).

The SMS interface allows users to retrieve drug side-effect information by sending a “text message” from their mobile device to our system containing a drug name. SMS messages have been used for a wide variety of clinical communications, including behavioral interventions, health promotion or education, and adverse-event reporting. We use the DOTGO service1 to receive and process these messages, and to send responses which the user receives as SMS messages. DOTGO is a free service that allows internet content providers to easily extend their sites’ functionality to include SMS features.

Our interface’s functionality is constrained by the technical limitations of SMS: each message may contain only 160 characters. We therefore split the output into two sets of messages. The first response a user receives from our system contains a numbered list of the MedlinePlus-derived descriptive sentences (“1. Call your doctor if these occur...”, “2. Seek immediate attention if you experience any of the following...”, etc.). This functions much in the same way as a text-driven menu interface: to retrieve the side effects corresponding to a particular descriptive sentence, the user replies to the menu message with the number of the desired list. The system then replies with its second set of messages, this time containing the requested side-effects (“Dizziness, muscle pain, headache”).

The SMS interface is currently English-only. Our SMS gateway service is unable to send messages containing accented or non-Latin characters via SMS, although it is our hope that these limitations will be removed in the near future.

The IM and SMS interfaces are intentionally simple to use, but they also have several non-obvious features. For example, if the system is unable to resolve an input string to a drug name, it uses Google to attempt to suggest a lexically similar word to the user. If a user asks the system for the side-effects for “asprin” (note the incorrect spelling), the system will reply with a suggestion: “Couldn’t find a drug named ‘asprin’. Did you mean ‘aspirin’?” This feature is especially important in the context of SMS messages, as it is often difficult to enter complex drug names on a mobile phone keypad.

The iPhone interface takes advantage of the device’s flexible user interface to offer a richer and more interactive experience. The user is presented with a search box, in which they may enter a drug name. The interface then queries our back-end system and presents any resulting side effect lists to the user using the standard iPhone user-interface conventions: a scrollable list of side effects, any of which the user may touch to receive further information. Touching/selecting a specific side effect from the list initiates a MedlinePlus search and directs the user to the search’s results. For example, should a user select “muscle pain”, they would be directed to a list of search results containing articles on muscle aches, images demonstrating why muscles hurt, and resources for dealing with chronic pain. The iPhone interface is the newest of our various front-ends, but offers the most promise of future expandability.

Translation Evaluation

To evaluate the quality of the translations that Google’s translation tools provide to our system, we used a very direct approach. Using a combination of clinician interviews, EHR record analysis, and literature review, we selected 14 common drugs that covered a broad range of side-effects and clinical uses (see Table 1 for a complete list). We asked two experienced clinicians (one nurse and one physician) to suggest drugs that they personally considered to have complex or difficult-to-remember side-effects. We also consulted the electronic prescription records from one author’s (AM) hospital to identify several of the most frequently-prescribed drugs. Furthermore, we referred to a recent review of ambulatory medication alert override behavior to identify several additional “problematic” drugs. Our intention was not to build a rigorous and exhaustive list of medications, but rather to select a small number of clinically-relevant drugs familiar to most clinicians.

After constructing our list of medications, we used our system to obtain a Spanish-language collection of side-effects lists for each drug. We then enlisted a panel of 25 Spanish-speaking clinicians in Buenos Aires, Argentina to review each drug’s translated list of side effects and rate it as being either “adequate” or “inadequate” for use as part of a medical decision. We specifically instructed the raters to mark a translation as “inadequate” if there were serious grammatical errors, mis-translated words, or other problems that would prevent the raters or their patients from being able to use the translation in a medical context. Raters were also able to enter free-form comments about each translation’s contents or quality, as required.

In addition to translation adequacy judgements, we collected basic demographic data from each participant.

1http://dotgo.com/
(age, sex, and medical specialty), and gave participants the opportunity to give us feedback on any aspect of the translations or survey instrument. We used the electronic survey tool SurveyMonkey\(^{3}\) to collect the responses, and analyzed the results using SPSS 16 (SPSS Inc, Chicago, IL) and Mathematica 7 (Wolfram Reserach, Champaign, IL).

Results
A total of 25 clinicians began the translation rating process; 16 reviewed all 14 translations. We limited our analysis to the ratings of the 16 clinicians that completed the entire survey. The mean age of the respondents was 31.8 years (standard deviation: 5.7 years); there were 11 males and 5 females. There were no differences in mean age between the group that completed the survey and the group that did not (two-sided \(p\)-value = 0.94). The group that completed the survey contained more males than females, whereas the group that did not complete the survey had more females than males. However, Fisher’s Exact test\(^{21}\) indicates that sex was not associated with completion of the survey (two-sided \(p\)-value = 0.098).

Our raters represented a wide variety of medical specialties, including Toxicology, Pediatrics, Gastroenterology, Medical Informatics, Obstetrics, and Family Medicine.

Our raters reported very mixed opinions regarding the adequacy of Google’s translations; however, for 11 out of the 14 drugs, more than half of the raters found the translations to be adequate (see Table 1 for a drug-by-drug summary). The raters’ comments on the three “inadequate”-rated drugs (Haldol, Ambien, and Paxil) were illuminating. These three drugs had some of the longest and most complex lists of side effects of any of the study drugs, and the original authors of these drugs’ lists used more idiomatic English expressions than in many of the others. For example, one of Haldol’s side effects is listed as “blank facial expression”. Several raters commented on the inadequacy of Google’s translation: “expresión facial en blanco” (lit. “white facial expression”). Several raters noted Google’s mistranslation of Prednisone’s “inappropriate happiness” side-effect to “inadecuado felicidad” (lit. “inadequate happiness”).

Another common complaint from raters was irregular grammar. MedlinePlus lists “uncontrollable, rhythmic face, mouth, or jaw movements” as a possible side effect of Haldol. Google’s translation engine was unable to correctly translate the complex structure and unusual comma usage in this sentence fragment (in which “uncontrollable” and “rhythmic” are both adjectives modifying all three categories of movement). Google was able to correctly translate the individual words, but failed to adjust their order, rendering their meaning ambiguous and difficult to interpret: “incontrable, rítmica cara, la boca, la mandíbula o movimientos” (a correct translation would have been “movimientos incontrolables y rítmicos de la cara, la boca o la mandíbula”). Google also had difficulty resolving certain word-sense ambiguities. For example, two of the side effects of Lipitor are “gas” (i.e., flatulence) and “hives”. Google’s translation engine chose to translate these as “de gas” and “colmenas”, respectively: “gas-powered” and “hives” (as in bees, as opposed to the medical condition). More than half of our raters commented on these particular errors.

Conclusions
Clearly, Google’s translation engine (and therefore the translations provided by our system) is not yet sufficiently robust for unsupervised use in a medical setting. That said, it is important to note that, for the majority of the drugs in our panel, our clinician raters felt that Google’s translations were adequate, despite being partially flawed. This may be an example of a “better than nothing” situation, in which an informed user may be able to derive some benefit or utility from a system so long as that user keeps the system’s limitations in mind. At the very least, this study demonstrates that Google’s translation system represents a reasonable starting point for medical translation, and may be able to serve as a foundation for more advanced or specialized translation techniques.

There were some important limitations to our evaluation. First and foremost among these was sample size: 16 raters and 14 drugs are not sufficient to fully evaluate and characterize the performance of a translation engine. The purpose of this study was largely exploratory in nature, and in conducting it we have learned many valuable lessons about evaluating translation quality. Another limitation of our study was that we only involved Spanish speakers, when the system is capable of translating side effects into dozens of other languages. For example, it is entirely possible that Google’s English–Arabic translation system is able to correctly handle grammar or word choices that its English–Spanish system does not.

Furthermore, our study used a very simplistic binary outcome measure, and we instructed our raters to be unforgiving with their ratings of the translated side effect lists. A more robust measure would have given us a more nuanced understanding of the issues of comprehension.
surrounding automatically translated texts. In this instance, however, our choice of outcome was heavily influenced by our system’s intended purpose, which is to be used as a component within larger systems in a way that is completely transparent to the user. The consumers of data from our system may be completely naïve about where their side-effect lists are coming from or how they are translated (or even that they are translated), and may therefore not be not be informed, “better than nothing”-style users. Instead, they may be “all or nothing” users, without access to much of the context needed to accurately interpret confusing mistranslations. It was for this reason that we chose such a restrictive evaluation approach.

There are many possible directions for future work. We would like to attempt a similar evaluation of a different language pair, or with the same language pair but a different translation engine. We also intend to investigate ways in which MedlinePlus’s Spanish-language content may be integrated into our system, and compare its linguistic utility to that of automated translations of the same content. Finally, we plan to integrate our back-end system with the Unified Medical Language System. Currently, its output is in a machine-consumable format, but does not include any standardized disease or condition codes.

In spite of its linguistic limitations, our web service represents a potentially valuable tool for system developers around the world who wish to include drug information in their applications. It is our hope that others will find further useful and interesting applications for our tool.

Acknowledgments

The authors gratefully acknowledge and thank the clinicians who participated in this study. They also wish to thank the AMIA reviewers for providing valuable feedback on the manuscript. This work was funded by NLM training grant 2T15LM007088.

References