Chinese to English
Introductory Cover Letter Generator

By Jason L. Stine
Liberal Arts and Engineering Studies
Chinese Language and Computer Science
Thursday, June 9, 2011

Dr. David Gillette and Dr. Michael Haungs, Advisors
California Polytechnic State University, San Luis Obispo

Table of Contents

An Idea is Born

Discussing Possible Project Ideas

As a Liberal Arts and Engineering Studies (LAES) major studying a combination of Chinese and Computer Science, deciding on what I was going to do for a senior project was challenging. I wanted to work on a project that was applicable to my studies, but at first there was no obvious answer as to what I could do. So like many students before me, I decided my best option was to attend various professor's office hours to seek their advice on this matter.

It was not until I met with Professor Clinton Staley of the computer science department that something really sparked my interest. There had always been something in the back of my mind hinting that it would be him who would be able to help me the most, mainly because of a privileged relationship as a classmate, not just student, I had with him in the Chinese language courses Cal Poly offers taught by Professor Sophia Chen. Professor Staley had wanted to learn Chinese as well, and as a fellow student in Professor Chen's classes, those of us studying computer science had many great opportunities to communicate with him about both academic subjects.

During his office hours something he said finally made me really excited. He told me that he and other computer science professors have been receiving an ever-increasing amount of emails from students in other countries abroad, especially from India and China. These students have many reasons to contact Cal Poly's computer science professors, from inquiring about general information about the professor's college to asking them for advice on projects they may be working on in their home countries. In order to do so, many of these students often have to give a basic introduction of themselves to provide necessary background information on where they are coming from so the rest of their letter makes sense. Unfortunately, in too many cases these email messages contained poorly written English. Professor Staley then asked the question which has since served as a basis for my senior project: “Wouldn't it be great if there was software that could generate these letters in *good* English?”

Narrowing It Down

After discussing the idea with other people including my academic advisor, Professor David Gillette of the English department and director of the LAES program, I narrowed my project's scope down to that of generating a basic introductory cover letter these foreign students could use to send to these professors in English-speaking countries. I would create a kind of automated translator for Chinese people who knew little to no English, thus allowing me to utilize the Chinese language side of my studies. After thinking hard about it, I realized this was a possible task. Creating any English letter of any topic would not be achievable in my case as the number of possibilities and combinations that arises in writing are infinite. However, as long as I limited the topic of this generated letter, it could be done. For instance, if one is to consider the ending of any cover letter asking for information about a certain institution or organization, it might end with “Thank you so much for spending the time to read what I have said above, and I am really excited in hearing back from you with more information regarding your organization.” As you can imagine, there are only so many ways one can say this and end such a letter. Therefore, by relying on this concept of limitation, I could see that writing software to produce such passages could be done.

Figuring Out the Specifics

Eventually I concluded that what I would do would be to translate very specific items entered in Chinese using Google Translate, and then take these English translations and plug them into "sentence templates" of English sentences written earlier which would be stored in a kind of database shipped with the software. Services like Google Translate often handle full sentences terribly and return to the user translated nonsense, but these tools actually do a fairly good job at translating very specific items like nouns and verbs standing alone. The software would intelligently assemble paragraphs by choosing relevant sentences which would make sense in the passage and not disrupt the "flow" of good, readable English. I would have control over what these sentence templates contained and when they could be chosen to be placed in a certain paragraph. In this way making a self-introductory cover letter in good English was possible.

I also did searches online in both Chinese and English to see if such language generation tools existed. I found a tool which could generate such letters in English, but the interface and instructions itself was in English, which would not help people who only understood Chinese. The generator also did not have a form of passage-assembly "intelligence": it was composed of drop-down menus with the sentence options, so in essence the user knew exactly what the passage would look like before it was generated. If the user knew enough English to use this tool, they could have created the letter perhaps even faster if they just typed it up themselves. Other than that, the only other items of interest I could find were human translation services which required a fee and full-fledged machine translators like Google Translate which have a track record of unreliable translations. I believe I also found software that could generate such passages automatically, but all I could find were extremely expensive packages that no ordinary Chinese user could afford, and therefore I was unable to try them out myself to see just how exactly the software worked.

Sequence of Events

The First Two Weeks: Using C++

Please see the Appendix section "Development Timeline" for a short summary of what follows below.

I started immediately on the project once I had free time during the spring break between the Fall and Spring quarters of 2011. At first I felt it would be appropriate to write the software in the C++ programming language, as that was what I thought I was most comfortable with since working on an earlier game project for the graphics classes at Cal Poly using that language.

I had to consider a crucial factor: how was I to create software that was cross-platform friendly? My main technical problem was that I primarily use the Linux operating system, and fully intended to use it to develop the application. However, my users would mainly be using a form of Windows, so whatever I made to work on Linux had to work on Windows as well. I did some preliminary research and determined that I should use the Qt graphical user interface system as it was apparently very easy to implement across various operating systems. This was the first time I learned how to use graphical user interface (GUI) design software, which allows developers to click and drag components such as menus and buttons and lay them out in a conceptual window.

I then tackled the next issue: getting my program to communicate with Google Translate in order to translate Chinese to English. After a few searches, I learned that the best way for me to do this was to use software called cURL which could be implemented into my program. cURL essentially

allows software to easily send and receive data over the Internet, and most importantly, contact websites to receive information automatically that a live human being usually does manually. I'm referring to the functionality of Google Translate. When you visit the site in person, you can type in the text you want to translate into a box and then select the output language, which results in a translation appearing. Well, instead of visiting the website themselves to do this, which can be tedious when one must translate many pieces of information, cURL allowed the developer to automate this functionality by utilizing a feature called representational state transfer, or RESTful, which allows a program to fetch information from a website via the web page's URL address. This method worked great, but it had one drawback: I had to install cURL as extra software in order for it to work, which meant my users would have to install it as well. My goal was to create an executable file with all the resources it needed so the user would not have to install anything else to make it run. This also brought into question using Qt – would my users have to install that as an extra component too?

With this in mind, I determined I could have a problem. If my users must be required to install extra software, that would dramatically lower the chances that they will actually use my own. In order to create a proof-of-concept application, I need to show that other people can use it, and therefore I must make it as easy to use as possible. Adding extra steps in order to run the software could only hinder this process.

I also came to another daunting realization. Because I was working with C++, which is tailored to work on the operating system the developer is using, I would have the additional work of "porting", or translating (in a programming language sense), the software to work on other operating systems such as Windows. Not only did I have uncertainties related to Qt and cURL I would have to deal with in the future, but now I would also need to reserve time to port my applications for use on my user's computers. I did some quick research on what it would take to port the software, and it looked like it would be too much for me handle. The primary purpose of this senior project was to focus on creating software that does a specific task, not to learn about how to get it to work on other people's computers.

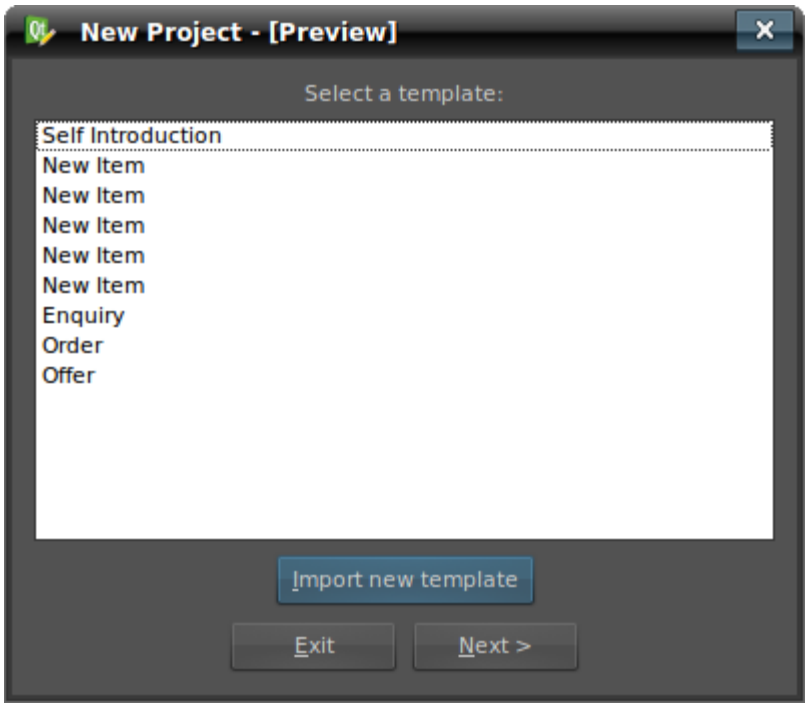

Figure 1: Conceptual window using the Qt GUI system for selecting an English passage to generate.

Enter Java

With the extra challenges discussed above, I concluded my current path of software development would not work for the amount of time I had to complete it. I asked some classmates for their advice, and they reminded me of the Java programming language which is cross-platform compatible. I was familiar with Java too, but it had been a much longer time since I had last seriously worked with it. However, I immediately saw the language's advantages. The most obvious advantage was that I would not have to worry about porting the software from Linux to Windows or Macintosh. What also surprised me was how easy it was to use Google Translate's RESTful interface by simply using the URL programming class that is included with Java. This feature was more convenient to use compared to cURL, and it got rid of the issue of installing cURL as an extra software package. Speaking of installing additional software, I also realized how it would be much better if I used a GUI system I was familiar with: Swing. I had mastered this system in high school years ago, and determined that it would be much faster if I used a system I was already familiar with compared to learning a new one, in this case Qt. The only downside to using Java was that the user would need to have the basic Java Runtime Environment installed, but most people already have that on their machines by default, so I speculated that this should not be too big of an issue.

With that, already one week into the Spring quarter and two weeks into the project, I made the critical decision to switch over to using Java and the Swing GUI system. I spent the next week primarily getting up to speed in Java where I was with my C++ version of the project.

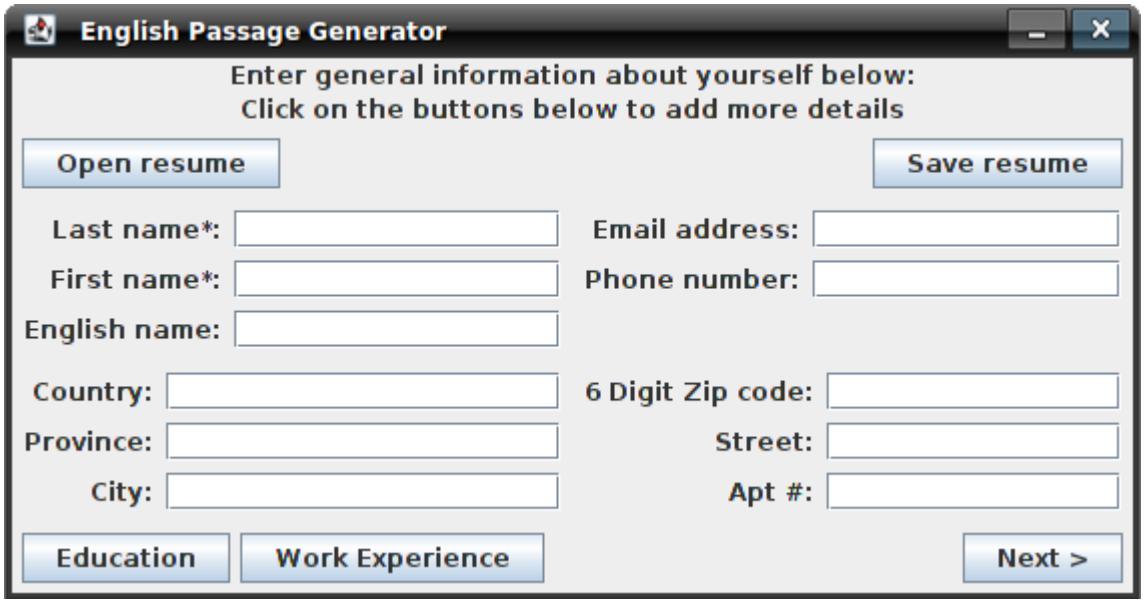

Figure 2: Main form for entering information. This is from the original English version.

Pushing Forward With Development

The next few weeks saw me working on the basic structure of the program. This included setting up the various windows, dialogs, input fields where the user would enter their information, and getting mechanisms prepared which would store this data. Much of this work also included putting parsers and checks in place to make sure the program could handle whatever the user entered or "threw" at it. Parsing is a technical term used to describe making sure software can work with information entered by a human. For example, if the computer can only handle a certain date format, in

my case in Chinese yyyy 年 mm 月 dd 日 (year, month, day), and not anything else (such as “month, day, year” - the standard Western date format), the programmer must make sure the user enters the date using the correct format. I would also say “parsing” encompasses which fields must be filled out in order for the user to continue using the program. I found it tricky to put checks in place making sure certain fields were filled in while at the same time allowing the program to continue if other fields/information was not filled in. I would say the challenge of parsing the user's information was perhaps one of the more difficult technical problems I faced, and personally I think this part of the code in my project is the least stable/reliable. I even implemented a feature that allows the user to save their information in a “.resume” file to their computer. This would allow them (and me), to open the file if the program is used again later to generate another cover letter and not force them to fill out all of their information a second time. I could use these files after testers tried out the software to see what exactly they originally typed in before creating their letters.

Translating the Software to Chinese

The idea was to first write the software in English, and then have my friends who could understand Chinese (other Cal Poly students including Aaron Gao and Justin Kuo) help me translate it to that language. I started asking them for help around the 4th/5th week into the quarter. Aaron helped me with the most translation work, and he would work with me about once a week up until the end of the term to get the text converted. I thank both Aaron and Justin for all the free time they devoted to helping me make this software possible. While they are responsible for most all of the Chinese text seen in the application, I can understand/read about 75% of it with my two years or more of Chinese language experience.

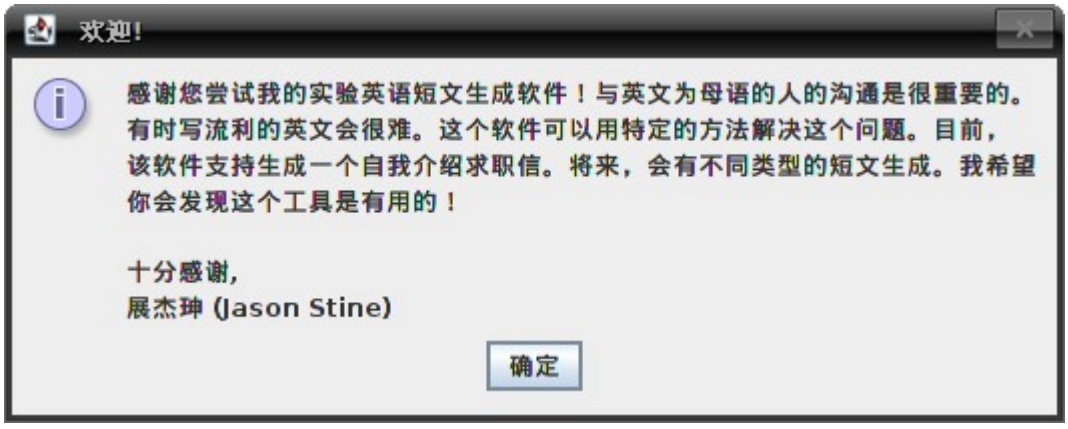

Figure 3: Welcome screen in Chinese shown first when the program runs.

Writing the Core English Generation Code

By the time the middle of the quarter arrived, I still had not started to implement by far the most important part of the software: the cover letter generator itself, or rather, the code that produced sentences and assembled paragraphs. This was one concept I had been struggling with in the back of

my mind for a while: exactly how was this going to work? To find out, I consulted one of my best friends and classmates within computer science, Eriq Augustine. We spent a good few meetings bouncing design concepts and ideas off each other, but the final decision of course was up for me to make. Eventually I settled for a sentence template system with tags representing the user's data. For example, one sentence near the beginning of the generated passage might be, “Hello, my name is <NameOfPerson>.” In this case, “<NameOfPerson>” would be substituted with the user's actual name, thus creating a complete sentence that works. The tags or less-than and greater-than symbols tell the program it is a keyword that must be replaced.

My idea was to create a simple database (in this case a text file containing all these templates to be read in by the software for use) that could be easily altered and added on to. I was thinking that what I could do was categorize sentences and store them in an organized way. For example, I would use a map data structure in the form of a “tree” data structure. A map contains two elements: one is the data, and the other is a “key” or reference to that data. One can retrieve the data by requesting it using the key. At the top of this “tree” I speak of, there would be three maps: one for an introduction, body, and conclusion paragraph. Each paragraph would then have a list of maps itself pointing to sections within them. Finally, each section would contain a list of node maps that connect to individual sentence templates as their values. While this would be very efficient in terms of retrieving a specific sentence (if the sentence was in the conclusion paragraph the software would not have to check sentences in the introduction paragraph), I decided to go with a basic list data structure instead as I realized I would not have that many sentence templates.

Database Options

Earlier, when I was seeking advice from Dr. Michael Haungs of computer science, he suggested I use a full-fledged database system and learn some basic SQL programming language code to deal with it. Poor Eriq spent an hour afterwards with me setting up a “portable” SQL database system that could be called by Java code and be included with my standalone executable. We were able to get a routine working with a “MySQL embedded library for Java” that we found online called “mysql-je”, but in the end I decided against using it. In reality I would only have maybe a hundred or less sentence templates to store as I only had so much time to create/write them. What was also an issue was the mysql-je's database's size: a massive 27 megabytes compared to what my entire project was at that time of just 150 kilobytes. For a Chinese user with a slow Internet connection in China, I could not expect them to wait for all that to download on their end. I studied abroad in China and lived there for four months myself, so even I know what the Internet is like for my users. Due to these constraints, I went with the easiest but perhaps not necessarily most efficient way and instead put the templates in a text file which was to be read in by the software and stored in a list data structure.

The following few weeks had me working on the system that produces sentences and the creation of job duties. Please see the sections “How a Sentence is Made” and “Verb Phrase Creation” for more information on how these processes work.

Finishing It Up

At this point (at about week 8), with the fundamentals in place, there were two main tasks I had to complete: getting the window which would display the passage set up, debugging, and finalizing the translated Chinese version of the software in addition to creating a traditional character version.

I laid out the passage display window to have controls corresponding to their paragraph type at the top. One of the more useful tools was the salutation/greeting creator. Users could type in the last name in English of the letter's receiver and then select a title for addressing that person (like Mr., Dr., etc.). The rest of the controls consisted of options to include certain information such as whether or not they wanted their high school to be mentioned or if their physical address was to be shown at the bottom of the passage in a Western format.

I also created a crash report dialog that would pop up if anything in my software went wrong. If I were to miss something in the program's code and it crashed while the user was using it, they would receive a report that such was the case and shut the program down at least somewhat gracefully. Like other mainstream applications out there, I requested that they send to me a copy of the report by email so I could troubleshoot it.

Finally, the last major task was to thoroughly debug the program. One must hand it to Murphy's Law as I had a couple somewhat “scary” issues develop during the final week of the software's development which could have jeopardized the entire project if I could not fix them (week 10). Both were related to running the program on Windows:

- For the list of verbs I provided for creating job duties, all of them were read in from a separate file. This worked fine on Linux, but the characters failed to appear correctly on Windows. My solution, while not the most elegant, was to hard-code them into the software itself so the compiler, which generates the end-result executable the user can run, could transform the words into a form Windows could properly display.
- For some reason the phonetic pronunciation I was retrieving from a website called pin1yin1.com, which included tone marks over certain letters, failed to display correctly in Windows. I eventually settled for a much better, offline alternative: software called “pinyin4j”. This also helped in that it made the software less dependent on resources online.

Last minute additions/suggestions I implemented were

- Versions in Chinese with a larger font. Some of my users had to squint uncomfortably to see the characters as Swing's default font size was optimized for viewing Latin-based languages like English.
- A pop-up encouraging users to fill out their work and education information if they hadn't already done so. Sometimes users would miss those two buttons on the main form's window, and the more information they could provide, the better the generated passage.

In the end, I released three versions of the software by language: simplified and traditional character editions and an English version.

Figure 4: An example of the window displaying the generated passage based on my own information, some of which I entered in English.

How it Works

How a Sentence is Made

What follows is a step-by-step process of how the program creates the sentences:

1. Determine what data exactly the user provided. If the user did not enter information for something, make a note of it. The program will not generate sentences for data that the user did not provide.
2. When a sentence is requested to be created, the software sends to the database tagged keywords which help determine what sentence exactly is needed next. Below are the types of sentences classified by their tag type:
   - Sentences with substitution tags. Again, the example, “Hello, my name is <NameOfPerson>” comes to mind.
   - “Filler” sentences. That is, a sentence that does not contain any of the user's information but is used to link sentences together which do contain this information. Sometimes one filler sentence can come after another. These help to keep the English passage's “flow” as smooth as possible, in addition to adding smooth transitional sentences. An example of this type of sentence is, “<FillerHeld> In addition, I used to hold another position.”
   - This type of sentence requested is a combination of the first two, except that instead of the previous type's leading tag classifying the sentence as a “filler”, this tag stands for a sentence type that would be appropriate to use given what the user's information is that is not to simply be substituted with following tags. For instance: “<Attended> At <CollegeName> I studied in <Major>.” In this case the first tagged keyword “<Attended>” specifies the type of sentence. It means that this sentence should be chosen if the user had at some point in time attended a college or university but not necessarily graduated from it. The primary purpose for that specific leading tag was to distinguish the verb “studied”'s tense (in this case the past tense).
   - There are other types of tags for special situations like newlines or telling the program it needs to do something about the grammar/wording around that tag. For example: “<Holding> I began working <FromDate>.” Regarding the grammar/words that immediately precede the keyword tag <FromDate>, which the program will identify as being a “date” tag, this will trigger code which will determine, based on what date information the user provided, whether the preposition “in” or “on” should be inserted. Fortunately for cases like this a standard rule applies: If the date mentions a specific day, “on” should be used. On the other hand, if the date contains just the year or the month and the year, “in” should be used.
3. A particularly challenging phase of implementing this tagging system was how I was able to differentiate between the multiple jobs and schools the user specified. As is seen above, one of the keyword tags is “<CollegeName>”. This is a general-purpose tag for any school or institution that the user provides. Before the substitution process occurs, the software takes note of what user data is available and stores this information in a data structure. This data structure is a “map”, of which I explained earlier in the paragraph discussing my sentence template database options. This map can only store values with uniquely identifying keys, and in this case the keys are simply keyword tags from the templates of what data the user provided. In order to make these keys unique, the software ads a number to it representing the college or job

according the order that they were added by the user. For instance, “<CollegeName>” becomes “<CollegeName2>” for the second school that the user entered. Later, when sentence templates are being chosen and the correct user data is ready, this number is stripped off so the substitution process can commence.

4. To make each generated English passage unique, I introduced some randomization. This may not be the best approach for choosing sentences, but as long as the different sentences considered convey the same meaning, in most cases randomly choosing one does not degrade the passage's overall flow or quality. For example, when the software requests a sentence containing the tags “<LastName>” and “<FirstName>”, the following choices are found:

    - My name is <LastName> <FirstName>.
    - I'm <LastName> <FirstName>.

    As you can see, they both have the same meaning, and randomly choosing one or the other won't negatively affect the quality of the passage being constructed. In Chinese the last name always comes before the first name, and this is generally how we present such names in English when we use the name's phonetic pronunciation such as “Zhan Jieshen”.
5. After a random sentence is chosen, the actual sentence template manipulation such as noun substitution and grammar changes are made to produce a finalized version that can be placed in the passage.

For this section, let me make some concluding points. If the user does not provide enough information to create a sentence and a filler sentence would generally precede it, both of these sentences will not be added to the passage. Also, it perhaps should be noted that at first I didn't know “filler” sentences would be needed until a bit later on.

Verb Phrase Creation

While sentence template system I detail above is great, by the time the 6th week came around, I could not help but feel that using that technique alone would not make a satisfactorily complex enough English passage. For the job experience portion of the form the user fills out in the application, I originally intended to allow users to specify responsibilities or duties associated with each job they add. At first all I could do was allow them to enter in anything into a standard text field. This information would then be sent to Google Translate and inserted into the passage without any manipulation. I quickly realized this method was inadequate because the English phrase returned by Translate might not work with the types of sentence templates I was using to “list” these job responsibilities. For instance, I use the sentence template “<Held> Certain duties related to this position included <Duties>,” which require that a verb of the present participle form follow the word “included”. For those who forgot, a present participle verb is an “ing” verb like “managing”. It should be apparent now that if any other verb form were to be used, the sentence would be obviously erroneous and stick out like a sore thumb to its English readers.

To fix this issue, I knew I had to figure out a way to limit just what information the user could enter and how it was entered so I could guarantee that whatever was inserted into the duties sentence template would work. At first I thought I could somehow “outsmart” Google Translate and insert a Chinese word before whatever verb was being translated in an attempt to force Translate to return the present participle form of that verb. This word, 正在 (zheng zai), and a couple other variations of it, is

used before verbs in Chinese to make them of the present participle form, but this method only worked for some of them, and I needed all verbs to be of this form.

At this point I again consulted my good friend Eriq. We came up with the idea of having predetermined verbs already stored within the software and already translated to be in the correct English present participle form. These verbs would have to be very general so as to work with any job duty being described. I then consulted my academic advisor, Dr. David Gillette. He agreed that this was probably my best option. He suggested I come up with a list of perhaps 20 or 25 very general verbs and just go with that. He also suggested various resources I could use to find such verbs. However, I was uncomfortable with this solution. I felt that having only 20 or 25 different verbs to describe any kind of job would generalize the generated English passage too much. But Eriq was right about one thing: I might as well go with an extremely general solution that worked rather than with one that had the chance of not working.

Before I get to solution of this verb problem, let me now bring into the picture the noun that this verb would be acting upon. All I wanted to do was create very basic phrases such as “managing teams”. Unfortunately, the noun itself came with many problems of its own. Notice that the previous example uses “teams”, which is plural. If Google Translate returned the singular form of this noun instead, I would have trouble. “Managing team” does not work. It would have to be “managing a team” or “managing the team”. Before I knew it, articles and other grammatical concepts began to really make creating these phrases way too complicated. Then Eriq came up with another idea. What if I instead flipped the verb and noun to where the noun was followed by the verb. At that moment I felt I we were on to something. As long as the noun was singular, this could work. “Team managing” represents this new ordering. Strangely, for some reason I was incorrectly convinced for a couple of days that Google Translate would always return the singular form of a noun as long as I gave it the most basic form of that noun in Chinese. Unfortunately, Chinese does not have the concept of singular or plural noun forms. For instance, 狗 (gou), can mean “dog” or “dogs”. The surrounding sentence context which this noun is placed determines whether or not it is to be interpreted as singular or plural. If, say, the sentence in Chinese states a particular number greater than one of the noun, the noun's form is taken as plural. Therefore, Google Translate could either return the plural or singular form of the noun and which one it would produce was could not be predicted. Perhaps the singular noun followed by a present participle verb would not work after all.

Obviously I needed something else other than just Google Translate to aid me. Whatever order the verb phrase took, I needed code that could determine if the noun was plural or singular so my software could appropriately handle it. Then I found it online: natural language processing. I settled for a basic processor that would test whether a noun was plural or not by trying to “stem” it, or change it to become singular. If the word is unchanged after an an attempt to make it singular, that means the noun is already singular and is not plural.

Another friend of mine brought up another good point. He said whenever he spoke about job descriptions, he would always use the plural form the verb. As long as I could make the verb plural I would be good to go. I found yet another solution: a project called ModeShape, which helps to manage data in various forms, had open source code with a utility for transforming verbs to be in their plural form. This concluded solving the problems related to whatever noun the user entered in Chinese.

Soon afterwards I also decided to go with a list of translated present participle verbs, but found a resource which allowed me to expand this list to be more than just 20 of them. I found a “glossary” of precompiled job description verbs maintained by Simon Fraser University of Canada. I manually had to go through and translate all the verbs to Chinese, but it did allow me to expand the list to be of 124 words, and the work of finding general-purpose verbs to describe a job duty was conveniently taken care of for me already.

In the end, this allowed my software to construct basic phrases that are intuitive to English readers in the order of a present participle verb followed by a plural noun. As long as the user entered a noun in its most basic state without any modifiers (just “dog” by itself versus “big dog”), Google Translate should be able to return the proper word in English and then hand it off to the natural language processing code, which in turn can create a grammatically correct verb phrase for use in the generated passage.

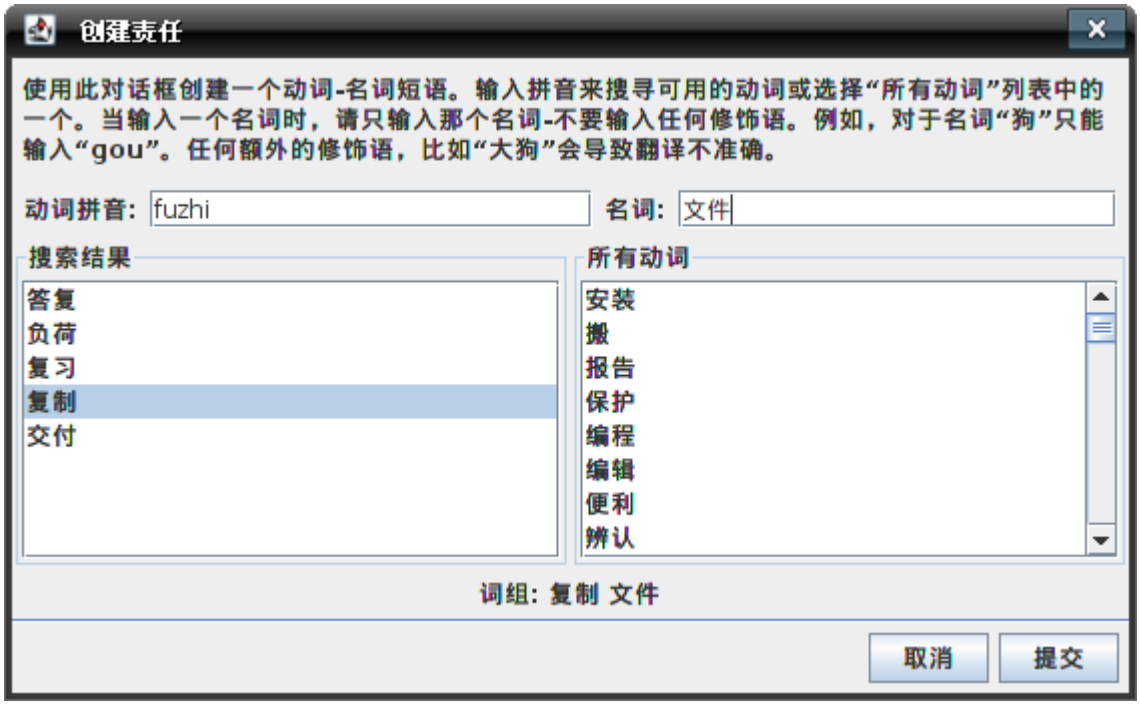

Figure 5: Job Duty Creation Window - the user can select a predefined verb and then type in their own noun to create a verb phrase. In this case the phrase “复制文件” (fuzhi wenjian) means “copying files”.

Feedback and Suggestions From Users

Please also see the general email message I sent out to my testers and the English cover letters they generated in the Appendix section.

- Side-by-side translations of generated passage not only in English but also in Chinese so users do not have to take my translation for granted. This may consist of two panes sitting next to each other so the user could compare them.
- A more advanced design that would give the users more control and allow them to assemble the paragraphs themselves. The idea would be to allow users to click and drag completed sentence templates with their information already inserted in Chinese and drop it into the passage they are creating, allowing them to build and form their own paragraphs.
  - However, such a tool would be more complex to engineer, especially when the goal is to keep the resulting passage flowing “smoothly” according to its readers. Because in my project I had control over what sentence templates were available and how they were used, I could keep transitions and other “flow” elements of the passage in check.

- Create a Cantonese (or other Chinese dialect) version of the software. Some of my testers' families had roots in southern China, and aspects of my project such as the pinyin phonetic pronunciation of characters and grammar of text for the GUI did not work very well for them. Pinyin is a system for pronouncing the characters in Mandarin, but in Cantonese there are other systems for such pronunciation. As for the Chinese text in the software, it may have been more suited for those who originate from northeastern China and may be subject to regional/cultural differences. An example of such a difference became apparent when one of the students who helped with the translation of the project, whose family comes from Taiwan, told me the word "software" was 软体 (ruan ti). However, earlier my other translator, who's family comes from northeast Mainland China, insisted that "software" was really 软件 (ruan jian). In the future this software would take into account such regional/cultural differences.
- Title creation of proper nouns would be upgraded. My project assumed that whoever used the program was located in China, and in this way would add "Province" after the territory they specified. One user entered "California" as where she lived, and the software incorrectly produced "California Province" instead of "the State of California".
- Fix the software to not literally attempt to translate proper nouns like company or employer names. Instead, have the name itself converted to its phonetic pronunciation and have, say, a menu that the user could choose the type of organization from, like "company" or "nonprofit organization".

One mistake I made when sending out the email was that I never mentioned that the software required the standard Java Runtime Environment package installed. I believe this may have created confusion for some users who could not understand why my program would not run on their computers.

Future Work

If I want to create a more usable and powerful tool out of this software, I would not hesitate to put it online in the form of a website. I would follow the lead of other successful Internet companies that have earned revenue by providing free services in return for advertising. Also, putting the program online would free the user from having to store the software locally on their own computers. A truly powerful and effective English passage generator would have a vast sentence template database and a much larger word database for entities such as the already translated verbs I have for the duty creator. As can be seen from the feedback received from my testers and my own experience, the natural language processing code I found was inadequate in handling all cases of nouns and their plurals. If I really wanted a system that could be effective for such a process, I would have to use a much larger and complete software package such as Stanford University's natural language processing applications. These applications are many hundreds of megabytes in size, and are thus not practical to require that each user download a copy. However, if the generator took the form of a website, there would only have to be one occurrence of each of these large packages stored on the website's servers, eliminating the problems described above. A website would also remove the prerequisite of a user needing software such as Java already on their machine which is needed to run the program, including other technical problems related to this.

Other generated passage export options would be supported. The only way one can save his or her generated letter would be to copy and save it using other means such as in an email message or

using a word processing program to save it locally to their machine. Future expansion could see the ability to save the passage including, but not limited to, the following file types: .pdf (Adobe's Portable Document Format), .doc or .docx (Microsoft Office Word document format), .odt (Oracle's OpenOffice Writer open document text format), etc.

Because I changed out the phonetic pronunciation resource at the last minute (I ended up using a program called “pinyin4j”), there is one aspect of this feature that I never had the chance to make clear to my users. Some characters have multiple meanings and thus multiple ways of pronouncing them. Consider the part of my software that takes a Chinese name in characters and converts it to its equivalent phonetic pronunciation. For instance, if a user were to type in the character 和 as part of his or her name, the program would print out "he", when in fact there are other ways to pronounce it such as "huo". The software does know about these alternatives, but I never got around to implementing some sort of menu system which would allow the user to select the correct pronunciation.

I would also make some changes to the way I store sentence templates. Now that I have successfully completed the project, I can see that a more efficient way of storing them would be to use an actual SQL database. In the future there will be many hundreds or even thousands of these templates, and simply storing them in a basic Java list data structure would greatly hinder template retrieval performance. Another such database could also be used for the storing of job duty verbs.

As I mentioned earlier when touching on the topic of parsing the user's data, I stressed that I felt this part of the code in the project was the weakest and least stable. Much effort in the future would be used towards developing a very solid, stable parsing infrastructure. Not only would this make the code more easily readable, but also make sure certain information is correctly filtered.

Finally, not just passages made for a purpose such as applying for a job at a company are what this software is limited to. An obvious addition could be to add a tool for creating basic resumes. My project already asks the user to enter resume-specific information, hence the file type extension “.resume”, so it would only be a matter of creating a formatted document with the translated text. Also, my job duty creator feature would be perfect for listing various responsibilities and short, to the point, verb phrases commonly found on resumes.

Concluding Remarks

This project has been an absolutely amazing experience for me. Unlike most students, I was very fortunate in that I had the entire spring quarter to devote to the development of this application because I did not have any classes as I had completed all of those the quarter before. At the beginning, I did not understand fully what I was getting myself into, but all this hard work really paid off as I was able to establish something that truly had potential for being useful.

From a technical point of view, this was probably one of the best computer science “classes” I have ever taken at Cal Poly, despite the lack of a daily instructor. After I changed majors two years earlier from Computer Science to Liberal Arts and Engineering Studies to focus more on learning Chinese as a second language, my academic career was shifted towards less programming courses and more liberal arts classes. Until the spring quarter of 2011, I had not seriously utilized my programming skills since a year before, so this exercise really brought me back up to speed in this area, especially with the Java programming language.

On the other hand, this project was perfectly applicable to my major in that it allowed me to focus on subjects other than computer science and programming. I learned many new words in

Chinese, and whenever I worked with the two students who helped me translate the software's GUI, Aaron Gao and Justin Kuo, the hours I spent with them taught me many aspects of the language as I was the one typing in all of the Chinese characters.

I also received a taste of what it is like for professional software developers to take a software package in one language and translate it to another, including what technical issues come up when this is done. A major factor related to this is language support on the user's computer and if their computer can properly display a certain language and its characters or letters. I was fully capable years ago in high school of creating a full-fledged application such as this, but what was really new to me was the Chinese language component.

Ironically, what surprised me was how this project turned more into the study of English and how this language works rather than that of Chinese. Initially, I thought all I would be doing was to write up sentence templates in English, have the software translate whatever the Chinese user entered, plug it into those sentences, and then add them to certain paragraphs of the passage. From my section above explaining how this process works and how the tagging system I came up with works, it is apparent that it was not that simple. I had to take into account grammar concepts of English, as well as making sure the passage was put together in a manor that was not awkward to the reader. In this sense most of my critical thinking actually went towards figuring out how the English part of it was going to work. Creating the GUI labels and instructions displayed by my program in Chinese was much easier as I had Aaron and Justin to help me.

On a final but somewhat amusing note, at the end of week 11, when I had by that time completed this project, I finally learned about what I had really created. After some searches online related to natural language processing, I discovered the concept of “example based machine translation”, or EBMT. While this concept has been discussed more in terms of simply translating one language to another, my software does take an “example based” approach when it uses the sentence templates to form sentences. EBMT takes a similar approach in using such templates, but as it is for general-purpose language translation, it relies on a massive database to handle any and every kind of general saying or phrase that is common between two languages. My project is different from that of a full-fledged translator in that it focuses on very specific subjects to translate, or rather, generate.

While the English letters generated were definitely not “perfect”, I consider this project to be a fantastic success as it is a solid proof-of-concept software package demonstrating that unique English passages that make sense to English readers can be created based on information completely entered in Chinese. Whenever I describe this project to others, I like to say that I am trying to “find middle ground” between two major forces in the language translation market: human translators and full-fledged machine translation software. The “human” part of the project includes aspects such as the English sentence templates being declared good translations beforehand, whereas the machine translating portion refers to times when the software sends a word to Google Translate. With this, I am proud to have come up with yet another tool to help break down the “language barriers” of our time.

Appendix

Development Timeline

Starting on March 21st, 2011, the beginning of Spring break week:

Week 0 (spring break) – Week 1:

- C++ version development, using Qt, and using cURL to contact Google Translate

Week 3:

- Created a Java version of the software
- Continued with UI (user interface) design
- Continued the data storage back-end development

Week 4:

- Started translating the English version of the software to Chinese

Weeks 5 through 7:

- Began work on the core sentence creation part of the software
- Created a sentence template database
- Continued developing the UI
- Began looking into how I would deploy the program for testing

Week 8:

- Developed the job duty verb phrase creator and added natural language processing software
- Refined the sentence creation/passage generation routines
- Began serious debugging procedures

Week 9:

- Created the window that displays the generated passage including corresponding controls
- Created a crash report dialog which would help debug the program

Week 10:

- Finalized software, finished debugging to the point of having a stable version for testing
- Created the traditional character version of the software
- Implemented better phonetic pronunciation processing software for Chinese characters (third party software called “pinyin4j”)
- Released completed versions on Sourceforge.net to users for testing

Week 11:

- Continued to share the software and receive user feedback

General Email Message to Potential Testers

- Please note that I also contacted testers in China using Chinese via instant messaging software.

Hi (...person's name...),

My senior project is complete – a software application that allows Chinese people who know little to no English introduce themselves in good English using a cover letter format.

It would be much appreciated if you (and even your friends) could help me by trying the program out. All I'm asking is for you to

1. Run the program (download below)
- Fill out your information (the more info you provide the better the cover letter)
- Save a ".resume" file with this information to your computer

2. Generate an English cover letter (keep clicking "Next >")
- Use the options at the top to customize this letter

3. Send me (jstine@calpoly.edu) your generated letter and .resume file
- Use the "copy" button to copy your letter
  - Paste it in an email to me
- Attach your .resume file you saved earlier

That's it! Thanks for trying it out!

I will be presenting this project at 11 or 11:30am on Thursday the 9th of finals week, and need some examples I can present to the panel who will judge me. You're welcome to attend if you haven't yet left SLO by then – email me for more information.

(See below for more information.)

----------DOWNLOAD THE SOFTWARE----------
If you can read Chinese, please use a Chinese language version.

--Windows--
Simplified Chinese:
http://sourceforge.net/projects/generateenglish/files/GeneratorSimplified.exe/download
Simplified Chinese (larger font):
http://sourceforge.net/projects/generateenglish/files/GeneratorSimplifiedLargeFont.exe/download
Traditional Chinese:
http://sourceforge.net/projects/generateenglish/files/GeneratorTraditional.exe/download
Traditional Chinese (larger font):
http://sourceforge.net/projects/generateenglish/files/GeneratorTraditionalLargeFont.exe/download
English: http://sourceforge.net/projects/generateenglish/files/GeneratorEnglish.exe/download

--Mac/Linux--
Simplified Chinese:

http://sourceforge.net/projects/generateenglish/files/GeneratorSimplified.jar/download
Simplified Chinese (larger font):
http://sourceforge.net/projects/generateenglish/files/GeneratorSimplifiedLargeFont.jar/download
Traditional Chinese:
http://sourceforge.net/projects/generateenglish/files/GeneratorTraditional.jar/download
Traditional Chinese (larger font):
http://sourceforge.net/projects/generateenglish/files/GeneratorTraditionalLargeFont.jar/download
English: http://sourceforge.net/projects/generateenglish/files/GeneratorEnglish.jar/download

ENGLISH VERSION NOTES:
- This software was made for Chinese users entering their information in Chinese, so don't worry if you get strange output due to entering information in English!

- For the job duty creator feature (when filling out your work experience), I've provided a list of all the verbs with their English translations so you can search for them using their phonetic pronunciation:
https://sourceforge.net/projects/generateenglish/files/GeneratorDuties.pdf/download

- Again, this software was made for people who use Chinese on their computer – if you don't have the language software installed on your computer, you may not be able to see the characters properly.

KNOWN ISSUES:
- On Windows operating systems (at least with XP), the font seems really grainy, so I've provided larger Chinese font versions.

- My program isn't perfect. For example, if my program outputs, say, a translated noun that's incorrectly plural (like “mails” instead of just “mail”), don't worry about it. Please send me these results so I can improve the software.

ERRORS:
- I'm only human, so I may have missed something in my code. If this is the case and if my program crashes for whatever reason, an error report should pop up allowing you to copy it and send the technical details to me. Please be as specific as you can in what you did that resulted in this error so I can later reproduce it on my end and fix the problem.

- If you notice any poorly written Chinese anywhere in my program, please don't hesitate to let me know with a better alternative. There may be regional differences as the person who helped me translate the software has roots in northeast Mainland China.

This project has great potential to become something even better, and I'm considering starting a business based on these ideas and concepts. Please let me know if you might be interested in working on something like this!

Please send me any comments or questions you may have after trying my software out.

Looking forward to your reply, and have a good summer,

Jason Stine

Cover Letters Generated By Testers

Letters Generated By Users Who Understand Chinese

- Please note that much of the following information provided by the testers is “fake” . Rather, what is important to focus on rather is the quality of the English translations and overall “flow” and grammar of the generated passages.

Sieg's letter:

Dear Mr. Lu Shengkai:

Hi, I'm interested in potentially seeking employment at your company. I'd like to introduce myself in order to hear your thoughts on what available opportunities might be appropriate for me. I would appreciate it if you could spend a moment of your time reading about who I am and what backgrounds I'm coming from. My name is Lu Yufan. In English, I go by Sieg. I reside in Yongan, a city that lies in the province of Fujian within China.

To begin, I'd like to mention a few details regarding my academic background. I previously graduated from Wing One High School in June 2008, located Wing of Fujian Province. However, I've gone further in my studies since high school. As a current student at Shanghai University, I am studying Computer Science.

I'll appreciate any advice you may have regarding job opportunities at your company that may suit me after considering what I've said above. It thrills me to think about the prospects of joining your team and being a part of developing your company's products! Thank you so much for your time, and I look forward to hearing from you soon!

Cordially yours,
Sieg

Address:
Fuyoulu
Apt. #C-605
Yongan, Fujian Province 366000
China

- NOTE: His resume file also included job information. I produced this letter using the .resume file he sent me:

Dear Mr. Lu Shengkai:

I am writing to request some information regarding your university. I am considering applying to be a student there, but would first like to make sure the programs your institution offers would appropriately satisfy my academic needs. I believe the best way to initiate this process would be if I first gave you a general introduction of myself. My English name is Sieg. My name is Lu Yufan. I'm currently living in Fujian Province, China, in a city called Yongan.

I'd first like to discuss my academic and educational background. For high school, I used to go to a

school called Wing One High School in Wing, Fujian Province. I graduated in June 2008. However, I've gone further in my studies since high school. As a current student at Shanghai University, I am studying Computer Science.

In the past I held a position at Xu Weimin as the Professor. For that job I began working in January 2011, which then ended in June 2011.

I believe the above more or less explains in general where I'm coming from. Please let me know what your thoughts are regarding this. Your university looks like it may have what I'm looking for, and depending on what your response is I may apply there! Thank you so much for your time, and I look forward to hearing from you soon!

With anticipation,
Lu Yufan

Raymond's letter:

Hello, thanks for allowing me this fantastic opportunity to introduce myself! I'd like to take the time to summarize aspects about me that you may find helpful in knowing. My name is Lin Feng, but in English I go by Raymond Lam. The city I live in is Shanghuan. This is in Hong Kong.

Please now allow me to explain my academic background. Earlier I attended The Chinese High School, an institution in Kowloon, No Province, in June 1999. This now brings me to the subject of college education. When I was at Canadian University, I had the opportunity to study Electrical Engineering. I might also mention that I minored in Chinese Culture. Afterwards I graduated in June 2004.

Working for China Travel Service, I'm currently working as the Manager, and I began on July 09, 2000. Certain duties related to this position include writing reports and training news.

Thank you so much for your time, and I look forward to hearing from you soon!

Kind regards,
Lin Feng

Aaron's letter:

Dear Mr. Stine:

I am writing to request some information regarding your university. I am considering applying to be a student there, but would first like to make sure the programs your institution offers would appropriately satisfy my academic needs. I believe the best way to initiate this process would be if I first gave you a general introduction of myself. My name is Gao Yibo. In English, I go by aaron. I'm living in China in a city called Tianjin.

To begin, I'd like to mention a few details regarding my academic background. I previously graduated from Righetti High School on June 13, 2008, located sm of california Province. This now brings me to the subject of college education. At cal poly slo I am majoring in environmental engineering.

I believe the above more or less explains in general where I'm coming from. Please let me know what your thoughts are regarding this. Your university looks like it may have what I'm looking for, and

depending on what your response is I may apply there! Thank you so much for your time, and I look forward to hearing from you soon!

Many thanks,
Gao Yibo

Address:
Nankai
Apt. #394
Tianjin, 389004
China

Scott's letter:

Dear Mr. Jason:

Hello, thanks for allowing me this fantastic opportunity to introduce myself! I'd like to take the time to summarize aspects about me that you may find helpful in knowing. My name is Wen Zhang, but that's my English name! My Chinese name is Zhang Wen. In China, I live in the city of Shijiazhuang, located in the province of Hebei.

Please now allow me to explain my academic background. Earlier I attended Hebei Normal University High School, an institution in Shijiazhuang, Hebei Province, in 2001. However, I've gone further in my studies since high school. As a student at Hebei Normal University, I studied Economy and History. Later I graduated in 2001.

At the moment I'm the Huai An East Road working for Great Economics. I started work in 2003.

Thank you so much again for spending some of your time to read up on a little about myself. It was a pleasure in getting this chance to be in communication with you, and I look forward to getting your response soon!

Cordially yours,
Zhang Wen

Address:
Hongqidajie
Apt. #123456
Shijiazhuang, Hebei Province 555412
China

Email: zhongwenxi@yahoo.com
Phone: 1234567890

Maggie's letter:

Hello, thanks for allowing me this fantastic opportunity to introduce myself! I'd like to take the time to summarize aspects about me that you may find helpful in knowing. My English name is Maggie. Mname is Liu Minqi. The city I live in is Xianggang. This is in China.

To begin, I'd like to mention a few details regarding my academic background. For high school, I used to go to a school called Prospect High School in Campbell, California Province. I graduated in June 2007. However, I've gone further in my studies since high school. At California Polytechnic State University, San Luis Obispo I majored in Biochemistry. I graduated in June 2011.

Thank you so much for your time, and I look forward to hearing from you soon!

Kind regards,
Liu Minqi

Ryosuke's letter (Ryosuke is a Japanese university student studying Chinese):

Dear Mr. :

Hello, thanks for allowing me this fantastic opportunity to introduce myself! I'd like to take the time to summarize aspects about me that you may find helpful in knowing. I live in Mingguwu. This is a city within Aichi Province, Japan.

To begin, I'd like to mention a few details regarding my academic background. As a major studying Chinese, I'm currently taking classes at Chukyo University.

Thank you so much again for spending some of your time to read up on a little about myself. It was a pleasure in getting this chance to be in communication with you, and I look forward to getting your response soon!

Sincerely,
Zhongdao Liangjie

Address:
Mingguwu, Aichi Province
Japan

Letter Produced Using the English Version

I had one user who cannot read Chinese try out the English version of the software. As a comparison to those letters produced above using the Chinese versions, here is Cris' letter:

Hi, I'm interested in potentially seeking employment at your company. I'd like to introduce myself in order to hear your thoughts on what available opportunities might be appropriate for me. I would appreciate it if you could spend a moment of your time reading about who I am and what backgrounds I'm coming from. In The United States, I live in the city of Santa Maria, located in the province of California.

To begin, I'd like to mention a few details regarding my academic background. Earlier I attended

Righetti High School, an institution in Orcutt, california Province, in 2007. This now brings me to the subject of college education. As a student at Cal Poly, I studied Liberal Arts and Engineering Studies. It's also true that I minored in Mechanical Engineering.

In the past I held a position at Holiday Inn as the Waiter. For that job I began working in 2005, which then ended in 2005.

I'll appreciate any advice you may have regarding job opportunities at your company that may suit me after considering what I've said above. It thrills me to think about the prospects of joining your team and being a part of developing your company's products! Thank you so much for your time, and I look forward to hearing from you soon!

Many thanks,
KIm Young

Address:
5000 Cherry trees
Apt. #12
Santa Maria, California Province
The United States

Email: xkimyoungx@yahoo.com
Phone: 8055555555

Bibliography and Resources

Honorable Mentions:

Professor Clinton Staley, Ph.D. of Computer Science:

- Came up with the original idea for this software

Dr. David Gillette, Ph.D. of English and Director of the Liberal Arts and Engineering Studies program:

- Helped with design concepts, gave advice on how to generate English sentences/phrases

Dr. Michael Haungs, Ph.D. of Computer Science

- Helped with database ideas and concepts

Eriq Augustine:

- Helped with design concepts and debugging

Aaron Gao (高亦博) and Justin Kuo (郭庭均):

- Helped with translating the user interface from English to Chinese

Resources:

Google Translate: translate.google.com

- For general translation of Chinese to English

pinyin4j: pinyin4j.sourceforge.net

- For retrieving the phonetic pronunciation of Chinese names

Glossary of Job Description Verbs. Simon Fraser University, Human Resources. 18 May 2011. <http://www.sfu.ca/human-resources-old/forms-cabinet/job-descriptions/cupe/glossary.pdf>.

- Provided verbs for job duties

Using code from the ModeShape project: www.modeshape.org

- Inflector.java: Transforms nouns into their plural form

Using code from the YAGO-NAGA project's Javatools:

- www.mpi-inf.mpg.de/yago-naga/javatools/
- PlingStemmer.java, FinalMap.java, and FinalSet.java: determines if a noun is plural