Automatic Curriculum Vitae Creator Using a Virtual Human Speech Interviewer

Joseph Clint J. Molon⁺, Sandra Mae W. Famador^{+,*}

Department of Computer Science,
University of the Philippines Cebu, Gorordo Ave., Lahug, Cebu City 6000, Philippines

*Corresponding Author: swfamador@up.edu.ph

Abstract

An application that takes in speech, analyzes it, and turns it into a Curriculum Vitae is an innovative approach. This study aims to create such application using text-to-speech, speech recognition, and natural language processing. The user is asked about his/her personal and educational background in an orderly manner using a text-to-speech module. The order of the questions depends on the format of the Curriculum Vitae. With the SpeechRecognition python library, the user's speech input is captured and converted into text which will be further processed and analyzed. Word tokenization is used to divide a sentence into list of words. Each element in this list of words is labeled with their corresponding part-of-speech tag. These tagged words will then make it easier for the program to identify the actual user input, which will be written in a Word or LibreOffice document (.docx, .doc, .odt) using the python-docx library. Using this approach, the study was successful in creating such application. Speech-to-text, speech recognition, and natural language processing can be integrated to create a virtual human interviewer capable of conversing with the user to create a Curriculum Vitae. Although the tools used proved to be very effective and efficient, the python-docx library does not provide a tool for editing written text in the event of the user giving a wrong input, creating an area for future research.

Keywords: speech recognition, natural language processing, Curriculum Vitae, virtual human agent.

1. Introduction

Handsfree technology is popular nowadays for its convenience. People are becoming more comfortable in talking to their devices in order for it to do certain tasks. Speech recognition and natural language processing can be integrated to create an application capable of understanding and interacting with its users. The Virtual Human Toolkit⁽¹⁾ made it possible for an easy production and development of virtual human agents that interacts with users. This study integrated different aspects of human-to-human interaction, which includes: speech recognition, natural language understanding, nonverbal behavior understanding, natural language generation, and nonverbal behavior generation. The addition of nonverbal behavior understanding and generation enhances the face-to-face interaction between human and computer.

This study is significant because it aims to shorten the time of filling up a Curriculum Vitae. Speaking, instead of writing, would make the process faster. Also, due to the limitation of the Speech Recognition software, words must be properly uttered for the System to rightly detect the input speech. As a result, non-native English speakers will have a chance to practice how to pronounce the words correctly. Another significance of this study is to create a framework for an automated speech to text form filler. This framework promotes more possibilities for future research.

An example of an application that interacts with humans is SimSensei⁽²⁾ which engages in a face-to-face interaction with a user, usually for assessing his/her mental health. This interaction involves verbal and nonverbal behaviors which

⁺ Both authors have equal contribution.

enhances the user's experience when talking to the virtual interviewer. The study used 4 trained Natural Language Understanding (NLU) classifiers, each of which: identifies generic dialogue act types, assigns positive or negative valence to the user input, supports domain-specific small talk, and identifies domain-specific dialogue acts. The dialogue policy the study implemented used the FloReS⁽³⁾ dialogue manager. The researchers conducted an experiment and the results showed that the users were comfortable in sharing information with SimSensei, giving the impression that virtual human agent are effective in extracting information from its users.

Virtual humans are proven to be huge advancements and are used to look into the health and welfare of patients^(2, 4, 5). Bickmore et al.⁽⁴⁾ introduced Hospital Buddy. A virtual human agent that accompanies patients during their hospital stay—including medical topics and entertainment functions. Results from this study showed that it truly provides companionship to patients who has nothing to do unless there is some medical intervention. SimCoach, a study by Rizzo et al.⁽⁵⁾, provides healthcare processes and support specifically to retired service members from the military. The study uses the clinical framework, PLISSIT (Permission, Limited Information, Specific Suggestions, and Intensive Therapy), which is a model that encourages help-seeking behaviors in a person who may feel stigma and insecurity regarding a clinical condition.

2. Pipeline of Work

The aim of this study is to make a virtual human agent capable of information extraction from the user's speech input and must be as accurate as possible. The pipeline of the study, as shown in Figure 1, depicts the steps taken and the structure of the automatic CV filler process. Using text-to-speech, the system starts with a question on the first statement in the CV model. It is then followed by speech-to-text. Tokenization is then performed by the system for it to understand the input text. To better understand the input, and to separate the words, speech tagging is performed. For each input spoken, the system will store it in the memory. Once the end button is pressed or all pre-defined questions are traversed, the system will reformat the strings so that the final output will follow the predefined CV format.

Fig. 1. Pipeline architecture for the automatic CV filler system

3. Text-to-speech and Speech-to-Text

Prompting the user for the speech input is done sequentially so that a defined template could be followed. This query specifies the Curriculum Vitae field that the user is asked to answer or fill. To be able to do so, this application utilized eSpeak⁽⁶⁾, a speech synthesizer for most operating system and platforms. The next thing the application would do is to listen for the user's speech input.

To make the user's input comprehensible to the application, speech-to-text procedures are needed. The following equation separates the speech from the acoustic:

$$\hat{W} = \underset{W}{\operatorname{argmax}} P(W|A) = \underset{W}{\operatorname{argmax}} \frac{P(W)P(A|W)}{P(A)} \quad (1)$$

where for the given acoustic observation $X=X_1X_2...X_n$, the goal is to find the resulting word sequence $\hat{W}$ that has the maximum posterior probability $P(W/X)$.

To implement speech recognition, we used Python¹'s Speech Recognition library.⁽⁷⁾ The speech recognition module completes the exchange between the user and the computer. The library enables the program to capture the user's speech and converts it to text, preparing it for further processes.

To follow the pipeline of work, the code was modified. Additional features were added such as word tokenization, and part-of-speech tagging. The application also provides guidelines on what to say to the speech recognizer when using the application.

As of the writing of this paper, background noise may confuse the recognizer on what the user is trying to say, leading to the input possibly being misinterpreted. To produce a more accurate output, background noise must be minimized.

4. Information Extraction

Information extraction is the part wherein natural language processing takes place. This part receives the text converted from the speech-to-text module and is applied with the procedures necessary for the application to meet its objectives. Included here are the processes of word tokenization, part-of-speech tagging, and the identification of the actual user input. The NLTK⁽⁸⁾ python library is the tool used in this study to implement information extraction.

4.1 Tokenization

Word tokenization is the process wherein a sentence is divided into a list of words. This prepares these words for further processes—usually lexical analysis. There are different word tokenizers that the NLTK library provides: the

TreebankWordTokenizer, PunktWordTokenizer, WordPunktTokenizer, and RegexpTokenizer. The function used by this study for word tokenization, `nltk.word_tokenize(value)`, is a wrapper function that calls NLTK's recommended word tokenizer, TreebankWordTokenizer. The snippet below shows how to tokenize words:

```
1 words = nltk.word_tokenize(value)
```

where `value` is the user's speech input. This line produces a list of every word from the user input, which is necessary when applying part-of-speech tagging.

4.2 Part-of-speech Tagging

Jurafsky and Martin⁽⁹⁾ discussed the Hidden Markov Model (HMM) for part-of-speech tagging. The goal of HMM decoding is to choose the tag sequence that is the most probable given the observation sequence of n words w_1^n :

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \quad (2)$$

using Bayes' rule to instead compute:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)} \quad (3)$$

and dropping the denominator:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n | t_1^n) P(t_1^n) \quad (4)$$

The HMM taggers make two simplifying assumptions. The first is that the probability of a word appearing depends only on its own tag and is independent of its neighboring words:

$$P(w_1^n | t_1^n) \approx \prod_{i=1}^n P(w_i | t_i) \quad (5)$$

the bigram assumption is that the probability of a tag is dependent only on the previous tag, rather than the entire tag sequence:

$$P(t_1^n) \approx \prod_{i=1}^n P(t_i | t_{i-1}) \quad (6)$$

¹ We used Python 2.7 in the implementation of this research. Python is a

scripting language created by Guido van Rossum

Table 1. POS Tags (source: Penn Treebank Project)

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	<i>and, but, or</i>	SYM	Symbol	<i>+, %, &</i>
CD	Cardinal number	<i>one, two, three</i>	TO	"to"	<i>to</i>
DT	Determiner	<i>a, the</i>	UH	Interjection	<i>ah, oops</i>
EX	Existential 'there'	<i>there</i>	VB	Verb, base form	<i>eat</i>
FW	Foreign word	<i>mea culpa</i>	VBD	Verb, past tense	<i>ate</i>
IN	Preposition/sub-conj	<i>of, in, by</i>	VBG	Verb, gerund	<i>eating</i>
JJ	Adjective	<i>yellow</i>	VBN	Verb, past participle	<i>eaten</i>
JJR	Adj., comparative	<i>bigger</i>	VBP	Verb, non-3sg pres	<i>eat</i>
JJS	Adj., superlative	<i>wildest</i>	VBZ	Verb, 3sg pres	<i>eats</i>
LS	List item marker	<i>1, 2, One</i>	WDT	Wh-determiner	<i>which, that</i>
MD	Modal	<i>can, should</i>	WP	Wh-pronoun	<i>what, who</i>
NN	Noun, sing. or mass	<i>llama</i>	WP\$	Possessive wh-	<i>whose</i>
NNS	Noun, plural	<i>llamas</i>	WRB	Wh-adverb	<i>how, where</i>
NNP	Proper noun, singular	<i>IBM</i>	\$	Dollar sign	<i>\$</i>
NNPS	Proper noun, plural	<i>Carolinas</i>	#	Pound sign	<i>#</i>
PDT	Predeterminer	<i>all, both</i>	"	Left quote	<i>(' or ")</i>
POS	Possessive ending	<i>'s</i>	"	Right quote	<i>(' or ")</i>
PRP	Personal pronoun	<i>I, you, he</i>	(	Left parenthesis	<i>[, (, {, <</i>
PRP\$	Possessive pronoun	<i>your, one's</i>	)	Right parenthesis	<i>],), }, ></i>
RB	Adverb	<i>quickly, never</i>	,	Comma	<i>,</i>
RBR	Adverb, comparative	<i>faster</i>	.	Sentence-final punc	<i>(. ! ?)</i>
RBS	Adverb, superlative	<i>fastest</i>	:	Mid-sentence punc	<i>(: ; ... - -)</i>
RP	Particle	<i>up, off</i>			

plugging the previous two assumptions to Eq. 4 results to the most probable tag sequence from a bigram tagger:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^n \overbrace{P(w_i | t_i)}^{\text{emission}} \overbrace{P(t_i | t_{i-1})}^{\text{transition}} \quad (7)$$

The emission probabilities, $P(w_i | t_i)$, are the output while the transition probabilities, $P(t_i | t_{i-1})$, control the way the hidden state at time t is chosen given the hidden state at time $t-1$.

Based on the work of Jurafsky and Martin, we can therefore tag words from the input signal or sentence. Words can be tagged as nouns, pronouns, adverb, verb, adjective, preposition, and the like (see Table 1). The NLTK part-of-speech tagging tool accurately labels the converted speech input with the appropriate part-of-speech tags. The following code snippet shows how to label each of the words from the user speech input with their corresponding part-of-speech tag using the list of strings words from the previous code snippet:

```
2 tagged = nltk.pos_tag(words)
```

this line of code gives us a tuple of every word paired with is corresponding POS tag. If “My name is John Doe” is assigned to `value` in the first line of the code snippet and `tagged` is printed onto the console, the resulting tuples would be:

```
>>> tagged
[('My', 'PRP$'), ('name', 'NN'), ('is', 'VBZ'), ('John', 'NNP'), ('Doe', 'NNP')]
```

As we could see, every word in from the `value` string is paired with their corresponding POS tag.

4.3 Identifying Actual User Input

The actual user input is the word or sequence of words to be placed on the output document along with its corresponding CV field. This is usually introduced after words such as “is”, “are”, “from”, and “in”. This can be done by identifying certain classifiers labeled by the part-of-speech tagging tool.

In distinguishing the actual user input from the speech input, words with the tags “IN” (preposition or conjunction), “VBZ” (verb, present tense, 3rd person singular) and “VBP” (verb, present tense, not 3rd person singular) are identified. These words includes “is”, “are”, “from”, “in”, etc. After identifying these words, the application takes the next substring to be the actual user input and is placed into the Word⁽¹⁰⁾ or LibreOffice⁽¹¹⁾ document.

5. Formatting the Results

Formatting the results into a CV form makes the final output readable and comprehensible. The sequence of the information is also necessary to avoid confusion and to follow the sequence of the predefined CV template. The python library `python-docx`⁽¹²⁾ lets you create and update Word and LibreOffice Document (.docx, .doc, .odt) files. This library was used in this research to present the user input being formatted in a form of a CV. The text are properly indented, place, and arranged. After the whole process is done and all the queries are asked,

Table 2. Sample run results

Total words uttered	91
Total words correctly identified	67
Total words not in Google database correctly identified	0
Total words in Google database incorrectly identified	14

the output of this application is written into a Word or LibreOffice Document file which is located in the same folder as the program file or a specified path by the user.

6. Experimental Results

The system was tested using several laptops with built-in microphone under both silent and noisy background conditions. In a noisy environment, majority of the words are not recognized. Noise in this study is considered as other environmental noise not generated by the speaker. Under a good background condition, where only the voice of the speaker is processed by the microphone, a good amount of words are correctly recognized by the system. Table 2 shows a sample run result. Accuracy of the system is 73.6% for total words in Google database correctly identified. Total words not in Google database correctly identified is 0%. For total words in Google database incorrectly identified, the system has an accuracy of 15.4%. Figure 2 shows a sample output.

7. Conclusion

A simple virtual human interviewer capable of filling up a predefined Curriculum Vitae can be done with the combination of speech recognition and natural language processing. These two processes usually go hand-in-hand in today's advanced technologies. Adding more features would further enhance the user's experience.

Figure 2. Sample output

The approach that this study took has shown a favorable outcome. Based on the results, users can make Curricula Vitae through conversing with the application, making the task hands free.

Part of the limitation of this study is that if the user does not pronounce the word properly, the algorithm cannot recognize the words that the user uttered. This is due to Google² training pattern of its speech recognition software. It is recommended that a training algorithm should be implemented so that the software can easily understand the speech pattern of the user.

For future research, the addition of computer vision to monitor or observe the user's gestures could tell the application what the user is feeling or the level of comfort. The virtual human should be capable of interacting in a human-like way which would improve the exchange between the device and the user into a more rapport conversation. Finding an effective speech recognition module that is not dependent on the connectivity of the device to the internet would enhance the level of convenience of the application. The user should also have the ability to return to a previous query and replace the user input with a new one, which is currently unavailable in this application.

Also, future work will include choice of several CV templates.

² Google is founded by Larry Page and Sergey Brin

References

- (1) Hartholt, Arno, et al. "All together now." *Intelligent Virtual Agents*. Springer Berlin Heidelberg, 2013. training." *Intelligent Virtual Agents*. Springer Berlin Heidelberg, 2013.
- (2) DeVault, David, et al. "SimSensei Kiosk: A virtual human interviewer for healthcare decision support." *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014.
- (3) Morbini, Fabrizio, et al. "FLoReS: a forward looking, reward seeking, dialogue manager." *Natural Interaction with Robots, Knowbots and Smartphones*. Springer New York, 2014. 313-325.
- (4) Bickmore, Timothy, et al. "Hospital buddy: A persistent emotional support companion agent for hospital patients." *Intelligent Virtual Agents*. Springer Berlin Heidelberg, 2012.
- (5) Rizzo, Albert, et al. "Simcoach: an online intelligent virtual human agent system for breaking down barriers to care for service members and veterans." *Healing War Trauma A Handbook of Creative Approaches*. Taylor & Francis (2012).
- (6) Duddington, J. (2010). eSpeak text to speech (Version 1.43.12) [Software]. Available from <http://espeak.sourceforge.net/>.
- (7) Zhang, A. (2016). Speech Recognition (Version 3.4) [Software]. Available from https://github.com/Uberi/speech_recognition#readme.
- (8) Bird, Steven, Ewan Klein, and Edward Loper (2009), *Natural Language Processing with Python*, O'Reilly Media.
- (9) <https://web.stanford.edu/~jurafsky/slp3/9.pdf>
- (10) Microsoft Word [Computer software]. (1986). Redmond, WA: Microsoft® software
- (11) LibreOffice 3.4.5 [Computer software]. (n.d.). Linuxpusher.
- (12) Canny, S. (2015). python-docx (Version 0.8) [Software]. Available from <https://github.com/python-openxml/python-docx>.