Synthesising Chat Sessions

Caroline Foster

June 4, 2004

Abstract

Festival's text-to-speech output was compared with the that of the computer game Unreal Tournament 2004. Some simple techniques for alleviating immediate tokenisation and pronounciation deficiencies are outlined, along with an introduction to other problems that were encountered during the investigation.

1 Introduction

Unreal Tournament 2004 is a networked multi-player first person shooter. It has different game formats (e.g. Capture The Flag, DeathMatch) that encourage different levels and types of player interaction. In the Unreal Tournament 2004 environment, players can communicate with each other via a chat client which synthesises user-entered text to speech. This is particularly popular at the end of the game when players no longer need their hands for killing each other and can type friendly messages instead.

1.1 Requirements

Many abbreviations are used, since speed of output is of prime importance. Emoticons are commonly used :-). This results in a heavier burden on the tokenising rules than for other types of text. The Unreal software had built in support for common gaming/chat abbreviations e.g.

gg → good game
lol → laughing out loud
:-) → smile.

however there is no functionality for adding extra abbreviations or redefining existing ones, you are stuck with the ones the game already knows. Gamers are not particularly eloquent and will often say the same things to each other e.g. many will say "good game" at the end of a game. Since these are all spoken with the exactly the same intonation by the synthesiser this is funny at first but soon sounds monotonous.

There are multiple speakers, often speaking in different languages with foreign or unusual names. Gamers will often spell their nickname with special characters or digits substituted for letters e.g. '@' for 'a', '3' for 'e', etc. to help make it unique. The Unreal software uses only an English voice. This can have some comical effects on European servers where conversations regularly include a mixture of English, French, German and Dutch.

The text is also reproduced on-screen, so it is not essential that every utterance be synthesised. This is in contrast to the usual approach where it is imperative that the message is spoken as it is assumed that it is not available in any other form. In particular; long sentences, urls, obscenities and garbage text are probably best left unspoken. This is something that the user may want to customise.

2 Customising Festival

The main Festival functionality is set up by the Scheme file init.scm. This script by default looks for a siteinit.scm local initialisation file containing additional scripts to run at startup, and all customisations were added to this file.

2.1 Abbreviations

To add abbreviations to Festival, they were defined in a Scheme module and loaded at startup. This module uses pattern matching to identify target abbreviations and expands them to the full list of words.

Many languages and domains have their own abbreviations e.g. in Dutch:
aub → alstublieft
ff → even.
so in practical terms is is impossible to cater for every one. This raises the requirement for users to have the ability to add their own abbreviations to the system at runtime. For research purposes this can be achieved by reloading the initialisation script after making ay changes. For a real application the abbreviations would preferably be stored in a data object accessible at runtime by both the user interface and the Tokeniser. A design of this type would make it much easier to add domain-specific abbreviation plugins.

2.2 Emoticons

It is difficult to know what to do with emoticons- should they be spoken or should they affect the intonation? A sophisticated system would try to apply the emotion to the prosody, but the Unreal chat client settles for replacing the emoticon with a suitable word e.g.
:-) → smile.
Even taking this simplistic approach with Festival, emoticons proved to be trickier to implement than normal abbreviations. They are typically constructed only from punctuation marks and so pronounciation rules dictate they are not required to be spoken at all. This means that they are not passed on to the tokenisation rules and so adding extra rules for emoticons initially had no effect. To identify the emoticon it it necessary to redefine the set of punctuation characters to exclude those used to build the emoticon.

2.3 Unknown Words

Foreign or unusual words can be added to the lexicon to correct mispronounciation. This does have the advantage that they can be added at runtime, but a disadvantage is that it only applies to the lexicon in use - a switch to another voice may require adding the entry again. Also, the word must be converted into the phonetic components and this requires some knowledge of the voice, phoneset and lexicon currently in use.

For this investigation the oald lexicon and the mrpa phoneset were used. Entries were tested with the don_diphone voice, but they should also work with any voice using the oald lexicon, e.g. the rab_diphone voice.

2.4 Intonation

Chat messages are usually short and to the point, so the Simple intonation module was selected. This method lives up to its name by simply stressing all content words. Since chat messages tend to be short and high in content words this makes them sound like exclamations which seems to give the right effect. By contrast the ToBI intonation method sounded more serious.

3 Further Problems

3.1 Multiple Languages

In dealing with multi-user chat sessions it is not valid to assume that everyone is speaking the same language, let alone that is is English. Ideally the chat system will detect the language of the text and choose an appropriate voice before attempting to synthesise it. Festival does not offer support for language selection, although this could be added as an extension.

Switching voices is relatively time-consuming, and in multi user sessions the language may change frequently. It is also likely that a speaker will use different languages depending on the language of the other users, and may switch between languages during a session or even a single post.

Taking this a step further, it should be possible to identify the speaker and choose a custom voice and/or speaking style to suit. SABLE markup supports the selection of different speaking voices and a text mode for SABLE is provided with Festival. The chat logs provided by e.g. Trillian[1] provide the username followed by a colon at the start of every line which makes it easy to identify the speaker. A new text mode could be defined to synthesise the chat logs by switching voices to identify the different speakers.

[1]a popular instant messaging client

3.2 Bad Input

Often in chat sessions people type fast and make lots of errors; it is irritating to the listener if all of these errors are spelled out. Many of these errors could be rectified by running the text through a spell-checking module before passing it on to the synthesiser.

4 Evaluation

Festival deals well with some repetition e.e hahahahaha which is said very fast and actually does sound a bit like laughing. Adding one or more ? marks to phrases that would otherwise not be a question (e.g. it is where) does change the intonation, but adding any to an existing question (e.g. where is it) does not.

The Unreal software did not deal well with long urls or techniques used for emphasis e.g. surrounding text by dashes. In the best cases the symbol was simply ignored, in the worst cases the next word was not recognised and was spelled out instead of being said as a word. Festival correctly tokenised the "gg" as "good game" in both –gg– and *gg*. The dash is defined as non-spoken punctuation in the lexicon so was not spoken. The asterisk was said as "asterisk", also as defined in the lexicon.

All changes to Festival were tested using the don_diphone voice. The quality of the speech generated was deemed to be better than the Unreal software, particularly regarding the intonation.

5 Conclusion

Unreal also offers support for VoIP and this is increasingly being used for tactical in-game communication between team members.[2] The limitations of real-time TTS are too great for this demanding user environment and as a practical tool it is likely to be eclipsed by the growing availability of VoIP systems.

[2] regular players form clans and play in leagues so team tactics are important