Spoken Language Generation - Part II

Aitor Azcarate
0017949

Joeri Honnef
9992359

Jelle Kastelein
0026549

Paul Koppen
9936696

Abdullah Özsoy
0108901

Liang Wang
0010782

Klara Weiand
0529478

Abstract

In this paper, we explore two sides to expressive speech synthesis. The first is the identification and simulation of different speakers, the second is the synthesis of emotional speech. The two are then combined, generating expressive characters that provide emotional spoken language. All of this is done within the context of fairy tales, as this is expected to be a particularly suited domain.

Contents

1 Introduction

This report will investigate methods to combine different voices and emotions in a synthesized spoken text. This combination should increase the realism of the text, as well as prolong its listenability. Of course, the ability of the Text-To-Speech (TTS) system to generate listenable text is an important one, particularly in, but not limited to, situations where the task is directed at entertaining the audience. With this in mind, we will demonstrate these techniques by applying them to the spoken language synthesis of several fairy tales. Fairy tales are very well suited for this task because they use the full range of emotions, as well as direct speech. Since children are often easily distracted, the listenability of the text is particularly important.

In an effort to make our approach as general as possible, we made use of an existing, and widely accepted markup standard, known as SABLE[1]. This language defines a variety of XML tags that modify vocal features, such as speed, pitch and volume. Using a markup language, rather than, say, Festival[2] scripts, means that we have a way of generating synthesized speech under any TTS system that supports this markup. An advantage of using SABLE is that it is supported by many different TTS systems, even though it is no longer being maintained. The new standard, SSML[3], is very promising, but lacks easy incorporation in Festival. Since SSML and SABLE are similar in many ways, our approach should, in principle, work for both languages.

The project consisted of two parts.

1 To annotate and define character speakers

2 To synthesize expressive and emotional speech

Each of these tasks is carried out by a single group. Finally the two will be combined into a single framework. Two different approaches to accomplishing this symbiosis of speaker and emotion were tested, one in which emotion was added separately to a SABLE file that was already labelled with different speakers, another in which emotions were added on the fly before generating the SABLE file. Each has it's own advantages and disadvantages, which will be discussed later.

[1] www.bell-labs.com/project/tts/sable.html

[2] Festival is the speech synthesis system that was used. See http://www.cstr.ed.ac.uk/projects/festival/

[3] http://www.w3.org/TR/speech-synthesis/

Basically, all the differences in character voices and emotional style can be made by adjusting only a few specific parameters. In SABLE, the most important characteristics that can be adjusted are:

- Basic speaker voice, as denoted by the XML tags:
  <SPEAKER NAME="..">
  </SPEAKER>
  which selects a specific speaker voice.

- Baseline Pitch, as denoted by the tags:
  <PITCH BASE="..">
  </PITCH>
  which defines the baseline pitch offset for the character voice.

- Pitch Range, as denoted by the tags:
  <PITCH RANGE="..">
  </PITCH>
  which can be combined into one tag with the baseline pitch, and defines the vocal range offset.

- Rate of speech, as denoted by the tags:
  <RATE SPEED="..">
  </RATE>
  which controls the speed at which the text is spoken.

- Volume of speech, as denoted by the tags:
  <VOLUME LEVEL="..">
  </VOLUME>
  which controls the volume of a character's voice.

All offsets are measured in percentages. Similar tags can be nested, and the resulting offset of a nested tag will be relative to the corresponding outside tag. So, if an initial offset of 100% is given for the pitch base, and we nest another pitch base offset tag within it, the result will be cumulative.

There are some more tags to consider, which can help to increase the textual flow, such as adding emphasis statements (with <EMPH>".."</EMPH>) or breaks (with <BREAK/>).

Keeping in mind the two tasks mentioned above, we will start by describing the annotation and definition of different character speakers, followed by the approach to synthesizing the four basic emotions, the attribute values of which were based on the emotional values as proposed in [2]. Finally,

each of the two approaches to the combination of the two projects will be discussed.

2 Group 1

2.1 Annotating and defining character speakers

The fairy tales were selected from a range of compilations of fairy tales retrieved from the Gutenberg catalogue of free ebooks[4]. A subselection of fairy tales was then made according to the following criteria:

- the length of the fairy tales – some fairy tales span several chapters and were thus considered to be too long
- the ratio of direct speech
- the occurrence of emotions and their range

The latter two points are important because this work is concerned with synthesizing text, using different speakers and emotions, and the text chosen should of course be suited for that task, in that it allows for the use of different voices and the full range of emotions. From the resulting fairy tales, four were selected to be used for the remainder of the project: "*A Mad Tea Party*", an excerpt from "Alice's Adventures in Wonderland" by Lewis Carroll, "*The Hillman and the Housewife*", from "Old-Fashioned Fairy Tales" by Juliana Horatia Gatty Ewing, and "*Tom Tit Tot*" and "*Mouse and Mouser*", from "English Fairy Tales" by Joseph Jacobs.

2.1.1 Speaker definition

As was pointed out earlier, fairy tales are very well suited for experimentation with emotional, multiple-character speech, because they usually contain a fair amount of direct speech from different characters, as well as emotional passages. The first thing needed was to identify these separate speakers, and create voices that sound characteristic, given the definition of their character. Each speaker was manually assigned a voice, gender, initial SABLE settings and a number of different attributes, which were later converted to useable SABLE markup. Since the voice to be used has to be specified explicitly in SABLE either by its name or the gender and a number (e.g. male2), the "gender" tag serves no direct purpose in this implementation. However, it is supported as a valid flag within SABLE itself, and other markup languages like SSML, which we also experimented with, make use of "gender" tags, which might thus prove useful in further adaptations.

While there are seven different male voices available for Festival and MBROLA[5], a speech synthesizer that can be integrated with Festival, only one female voice was provided. This meant that all variety and characteristics in the female voices had to be achieved through modification of the one female voice in the initial SABLE settings, while for male characters the range of different voices allowed for the selection of the voice that was considered to be best suited for each character.

Aside from the SABLE features describing pitch, rate of speech and volume, we assigned each speaker an "age" and "species" tag, which were used to automatically adjust the voices, depending on the properties of the different characters.

There is an "age" feature already defined in SABLE, but for some speakers, its usage did not seem to do anything in Festival, and so we used the age tag to modify the speech parameters directly when defining a character's voice instead. The pitch base is the primary parameter to be modified. We lower the pitch by a percentage equal to the age of the speaker (so, a speaker 30 years of age will have a pitch modification of -30% with respect to the baseline we defined).

The species parameter assigns features to non-human characters that we found to be characteristic for their creature type. A mouse, for instance, will get a +150% pitch base modifier, resulting in a very high-pitched voice, whereas a cat will receive a +200% pitch range modification, yielding a voice that sounds very animated and expressive. It will also be quite slow, making it sound like the cat might be purring, or, in some cases, even slightly predatorial and threatening.

Of course, if we want to control all SABLE pa-

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

[5] http://tcts.fpms.ac.be/synthesis/mbrola.html

rameters directly, we can just set the age to 0 and the species to "human", which will result in a 0% offset for all parameters.

The final voices are thus a combination of the manual selection of voice, the initial parameter values, and the automatic adjustment based on the "age" and "species" tags. For example, the voice of the Dormouse, a sleepy mouse in "*A Mad Tea Party*", is initially defined as

```
DORMOUSE = speaker(name="DORMOUSE",
sex="male", age=15, base_speaker="male3",
pitch_base=45, pitch_middle=15,
pitch_range=-44, rate_speed=-18,
volume_level=-62, species="mouse")
```

where the base speaker is the MBROLA voice used for this character. The low values for pitch range, speed and volume convey the "sleepiness"[6] of the character, yielding a quiet and monotonous voice. The conversion script then increases the pitch base by 150% because "species" is set to "mouse" and adjusts pitch values according to the age of 15. For a small analysis of each of the characters, see appendix B.

2.1.2 Tagging

In order to be able to let Festival read the fairy tales out loud with the appropriate voice for each different speaker, an indication must be given of where a change of speaker takes place. The same is true for a change in emotional state. To accommodate for this, the files were hand-tagged by the members of group 1. Because it is important to be able to make on-the-fly adjustments in the character definition, the tags used at this stage are not yet in SABLE markup. Instead, we tagged each file using tags of the form [[EMOTION,SPEAKER]] at each speaker change, where EMOTION is an element of the set of emotions and SPEAKER an element of the set of speakers. These tags were then converted into proper nested SABLE markup by the same script that automatically adjusts the voices (see above). All text in between two tags is assumed to be spoken by the same speaker with the same emotional state. Each tag is then replaced by closing tags for the preceding speaker, followed by opening tags for the next speaker[7]. For instance, if we have the tag for the angry character of the Cat in the tale "*Mouse and Mouser*", [[A, CAT]], we replace this tag by

```
  </VOLUME>
  </RATE>
  </PITCH>
</EMOTION>
</SPEAKER>

<!-- A -->
<!-- CAT -->
<SPEAKER
 NAME="female1"
 GENDER="female"
 AGE="teen">
<EMOTION NAME="angry">
  <PITCH BASE="40%"
   MIDDLE="-135%"
   RANGE="301%">
  <RATE SPEED="-36%">
  <VOLUME LEVEL="-24%">
```

This way, by assuming the same order of tags for all speakers, we can do a simple regular-expression-based find-and-replace search for the [[EMOTION,SPEAKER]] tags. But this does mean that each file must start with a speaker tag. If no speaker tag is found, we insert the Narrator as a default speaker. The <EMOTION> and </EMOTION> tags were added to be able to use the XML transform filter of group two for emotion addition[8].

Finally, where it was deemed appropriate, the hand annotated files were tagged with additional <EMPH> and <BREAK/> tags in order to improve the flow and naturalness of the speech.

[6] Which could of course also be automated by adding "sleepy" and the corresponding features to a new set of "state" tags, or perhaps to the set of emotions used.

[7] This works because the closing tags are always the same, and because we used the same tag set in the same order for each speaker. Because the first tag in a file is also replaced in the same way, the file will start with closing tags, which would have no opening tags. We simply remove the characters that comprise the first closing tags.

[8] This means that the file in question is not yet a fully useable SABLE file, and thus we output it as a ".emsable" file, to indicate that further modification is still required. The script also allows us to add emotions directly, inside the script, omitting the intermediate step of using emotion tags, and thus producing a directly useable SABLE file.

	ALICE	HARE	HATTER	DORMOUSE
ALICE	4	0	0	0
HARE	0	2	2	0
HATTER	0	2	2	0
DORMOUSE	0	0	0	4

Table 1: Identification matrix for "A mad tea party".

	MOTHER	DAUGHTER	TOM	KING
MOTHER	4	0	0	0
DAUGHTER	0	4	0	0
TOM	0	0	4	0
KING	0	0	0	4

Table 2: Identification matrix for "Tom Tit Tot".

	MOUSE	CAT
MOUSE	4	0
CAT	0	4

Table 3: Identification matrix for "Mouse and mouser".

2.2 Character evaluation

To give a (very basic) evaluation of the voices created, the members of group 2 were asked to listen to a sample file in which all the characters spoke one line. The narrator was explicitly given in advance, since his voice was the same in every fairy tale, and it was felt that, given the narrating style the narrator has in the fairy tales, it would be impossible to misidentify him. This meant that each member of group 2 had to identify three groups of four, and one group of two characters, making for a total of 14 characters to be evaluated. The complete test took 3 minutes and 21 seconds.

The voices in the test were grouped by fairy tale. To make sure that the identification could be made without hearing the complete tale, a short, one line summary of the fairy tale to be judged was given by the narrator before each identification group, in order to provide some context. The summary text for the Tom Tit Tot story, for instance, was

> "The second story is Tom Tit Tot. It's a story about a girl who must guess the name of a leprechaun, or be his forever."

Each character then spoke a sentence based on the following template:

> I am one of the voices of [*fairy tale name*]. There are [N] characters: [*a comma delimited list of characters*], and the narrator. Guess who I am?

where the items between brackets differed for each fairy tale, and N is the number of characters in the story, plus the narrator. Since we only have four evaluations for each voice, the numbers provide only a rough estimate of the ease with which each voice can be identified.

As is seen in table 1, most of the identifications for "*A Mad Tea Party*" were made correctly. The only misidentification is a switch between the mad hatter and the march hare, both of whom are male, and supposedly mad, and thus on first view do not differ very much in terms of their characteristics.

All the identifications for "*Tom Tit Tot*", displayed in table 2, were correct. This is no surprise, since each of the characters has quite distinct characteristics.

All the identifications for "*Mouse and Mouser*", displayed in table 3, were also correct. Again, the differences between the two characters are quite large, and so there was little room for confusion. Finally, table 4 shows the identifications made for each speaker of the fairy tale "The hillman and the housewife". This fairy tale has the most confusing set of speakers, since the chimney and the hillman actually turn out to be the same speaker (but we did create two separate speakers with a very small pitch and volume difference). Since the two are practically the same, we suspect that the listener can only guess.

The results speak for themselves. Even with only little information about the characteristics of the speakers, 8 out of 14 speakers were identified correctly unanimously and two by majority, and in no case was a speaker clearly misidentified by the majority of votes. The problems that arise seem to stem mostly from swaps between similar characters. One must also keep in mind that the evaluation synthesis used has placed each character outside of the context provided in their respective fairy tale, and it will be much easier to distinguish them within the context of a fairy tale, since explicit cues are often given by the narrator, who, through the

	SERVANT	CHIMNEY	WIFE	HILLMAN
SERVANT	3	0	1	0
CHIMNEY	0	2	0	2
WIFE	1	0	3	0
HILLMAN	0	2	0	2

Table 4: Identification matrix for "The hillman and the housewife".

role he takes in the narration of the fairy tales, will be almost impossible to misidentify. Overall, we consider the construction of voices successful, as the rate of correctly guessed voices is high and, as explained above, the contextual cues will ease the identification of the few characters for whom the identification was ambiguous in the test.

3 Group 2

3.1 Synthesizing expressive speech

Our goal is to express four basic emotions in natural speech; neutral, happiness, anger and sadness. The relevant and promising synthesis parameters that will realize these emotions have already been mentioned in the introductory section (e.g.: Baseline Pitch, Pitch Range, Rate of Speech, and Volume).

Currently, a lot of research is being done on emotions in information technology in general. We claim that adding emotions to speech in daily life user systems makes people feel more understood by these computers. Although we have not researched any scientific evidence of this claim, it does seem very intuitive.

For example: When we synthesize a fairy tale for children, they will immediately notice the difference between a monotonous synthesized piece of text, or one with added emotions. But, of course, not only children will notice this difference. Almost all of us know the 'silly' sounding applications of public transportation systems, which offer realtime feedback by telephone. Either they do not adequately interpret your question, or they simply do not properly identify the right words you are pronouncing, but in the end, the most upsetting thing about these applications for many people, is that there is almost no intonation and emotion underlying the voices with which they reply. In all likelihood, these systems will be much more listenable to the average human user, and will therefore be used a lot more, if they sound a bit more natural. Adding more emotion to a synthesizer will thus make it sound more empathic, even if the content that it pronounces is complete rubbish.

So, what core principles do actually define emotions? We all know what emotions are, and we know how they are expressed, but they are so common that we do not really consciously process them anymore. When we are asked to define these speech core principles we all directly name things like volume and pitch. When we further elaborated on these two we finally found that there are, of course, more than just these two. For this experimental setup, we chose to fine tune the parameter values mentioned in the main introduction of this report. Our task, basically, was to find the right values for these parameters. In finding these, we used a more fine grained table of parameter values from the paper by Pirker et al., [3]. We compressed this table (see table: 5) and translated the values afterwards into parameter values for the SABLE domain.

The next paragraph will describe what approach we used in order to transform an input XML file of group one into a proper formatted XML output file, which is the actual SABLE file which is fed into the speech synthesizer. And why we chose for this XML/XSL approach.

The Extensible Stylesheet Language, XSL hereafter, is the standard for transforming one XML structure into another. This is very useful if different applications require different formats, but they want to share the same information. Exactly this is the case in the connection between the character generation and emotion definition. Because SABLE is conform XML standards, the output of group 2 (first structure of information) can easily be transformed so that emotional vocal parameters are injected (second structure of information). The XSL file first matches the root SABLE tag. Then it starts a search for <EMOTION/> tags where the emotion values will be placed. All information outside these tags is, without modification, copied to the output, so <SPEAKER/> and other definitions remain intact. For each recognized <EMOTION/> tag, the name is searched for in a separate XML file that holds all particular settings for the different emotion types (sad, happy, angry and neutral). The result from that XML file is

	Angry	Disgusted	Happy	Sad	Scared	Surprised
avg pitch	-5	0	-3	0	10	0
pitch range	10	3	10	-5	3	8
volume()0-1	10	3	0	-5	10	5
speech rate	8	-3	2	-10	10	4

Table 5: Table extracted from paper by [3]

a collection of tags (<PITCH/>, <RATE/> and <VOLUME/>) with attributes and their appropriate values. These tags are then recursively placed into the output. The innermost tag is then filled with the original content of the <EMOTION/> tag, which is the text that has to be synthesized with this emotion. The application of this XSL transformation is carried out by python, writing the result to a .sable file. Then python makes a system call to run Festival on this file so that the emotional fairy tale is spoken wisely by our beloved narrator.

After these transforms, we just listened to the results and used our common sense to adjust the parameters here and there.

In the end, we realized that speech with a higher degree of emotion is probably possible if we could also adjust pitch contour and do the annotation at word-level. Doing this at word level requires some understanding of the text. For example, proper nouns are good indicators of text content. WordNet[9] could be used to get some more understanding of what kind of emotions are expressed in the text. After the classification of the emotion in a sentence or even at a more detailed level at positions within a sentence we can annotate the speech synthesis parameters, and get better emotional and expressive speech synthesizers.

3.2 Emotion evaluation

Evaluating the emotions

While defining the emotions we changed the values of the parameters to reflect our own recognition. Sad was very recognizable. It sounded too slow though, thus we increase the speed. The emotion 'happy' was intensified because it sounded better, we did this by increasing the pitch base and volume. Neutral was changed to make it fit better in comparison to the other emotions. It should not have as narrow a range as sad. It should have a normal speed, and since we changed the speed of the other emotions a small speed increase was needed. We also increased the speed of 'angry', this felt more threatening.

We evaluated our emotions on different people, members of the other group and friends. We used sentences that did not have an emotional meaning (e.g. 'Would you close the door please?') but still could be interpreted as emotional statements because we did not want to prejudice our test results. We gave the test subjects a list of the emotions to choose from and a sound file with the different emotions. The test subjects listened to the sound file and gave feedback on the emotions heard in every sentence.

The result matrix can be found in table 6.

	neutral	angry	sad	happy
neutral	5	0	0	
angry	0	5	0	0
sad	0	0	5	0
happy	0	0	0	5

Table 6: Evaluation scores of the emotions.

As can be seen in the result matrix all subjects correctly recognized the emotions. For our current needs this is sufficient and we did not further adjust the emotions any more.

4 Combining characters and emotions

In this section, two approaches to the construction of the symbiosis between characters and emotions.

[9] http://wordnet.princeton.edu/

4.1 On-the-fly summation

One downside to using new <EMOTION> tags is that they are not part of the SABLE markup definitions. This means that the output files are not directly useable, since they contain undefined operations. By simply summing the emotion-specific pitch, volume, and rate of speech values found by group 2 with the values specific to the speaker inside the tag-replace script, the values for emotional speech can be synthesized on-the-fly, rather than using <EMOTION> tags. This means that the output files are directly useable for TTS systems with SABLE support. We can also combine[10] emotional values very easily this way, simply by taking the average values of the emotions in question. This gives a somewhat more nuanced range, which can be useful, for instance, when gradually switching between emotional states for a single speaker, or when toning down an emotion by uniting it with a Neutral tag.

A downside of this approach is that we cannot easily modify existing SABLE files by adding emotions. This is where the second approach to combining speakers and emotions comes in, which was implemented by group 2.

4.2 Stepwise annotation of emotions

This process can be divided up into two steps. Step one, performed by group 1 outputs an XML file. This file contains the emotion tags specific to each speaker.In this way the XSL tranform knows what pieces of XML to transform into what specific SABLE parameter values. As group 1 decided to use the emotion definitions of group 2 and to build in a nice facility to convert their own handmade tags on the fly to the desired output SABLE file, group 2 decided to adhere to the XML standard. In this way we can interpret any third-party SABLE file which contains characters and add the emotions using the XSL transform discussed before in section 3. A small drawback of this technique is that we absolutely need the emotion tag, which is currently not a standard tag in the SABLE definition and off course will never be there since SABLE is not maintained anymore. On the other hand, people can define their own DTD specifications for their output character XML files. This enables group 2 to easily rewrite their script, which extensively makes use of XML libraries which support reading DTD files and so interpreting new XML input files of third-parties.

[10] For instance, if the MOUSE speaker is both Angry and Sad, we use the tag [[AS,MOUSE]], with the emotion tags in alphabetical order.

5 A note on Festival

A final note that should be made before presenting our conclusions, is that getting Festival to run correctly with the MBROLA voices on the University computers proved to be practically impossible. Even after the required voice files were installed, and after having changed parts in Festival itself, the program still did not generate speech from SABLE or SSML files. Some other TTS systems were also considered, but these systems either provided only limited support for automatically generating speech from annotated text, or a limited subset of the functionality of the annotation.

6 Conclusion

Although a few errors were made in trying to identify the different character speakers, the larger part was successfully identified. We have only little evaluation data at this point, but we foresee no problems with the current approach. In fairy tales in particular, the characters are often quite polarized, which means that they can be easily distinguished.

The definitions of the emotions were taken from a table defining these emotions (as seen in figure 2). These emotional parameters were then slightly exaggerated, according to the ability of the members of group 2 to recognize them. At this point, the ability to identify these emotions is a more important goal than making them sound exactly humanlike. It was found that this strategy was justified by the evaluation performed by group 2, the guesses of whom were all correct. Of course, the fact that there were only four emotions to be considered in this experiment, the set of which was given in advance, may well have played a role. It may also be the case that the emotions are easier to identify *relative* to one another than they are individually. In spite of this, we feel confident that the results

found with the current definitions of the emotions satisfied the project requirements.

A question that does arise, is whether SABLE is the best possible option for generating emotional speech. A particular shortcoming of this markup is that it is severely limited in it's ability to control the prosodic contour of a sentence. Although emphasis can be added, this has only limited effect, and often the side effects of stressing one word on the rest of the sentence are quite hard to predict.

In spite of this, the end result sounds quite nice. The synthesis is relatively lively, and inside the context of each fairy tale, the characters are easy to keep track of. Therefore, we consider the project to be a success.

References

[1] J. Cahn. Generating expression in synthesized speech, 1989.

[2] Silvia Quazza Stefano Sandri Enrico Zovato, Alberto Pacchiotti. A rule based approach towards emotional speech synthesis:. In *Proceedings of the Fifth ISCA ITRW on Speech Synthesis (SSW5)*, pages 219–220, 2004.

[3] H. Pirker and B. Krenn. Assessment of markup languages for avatars, multimedia and multimodal systems. In *NECA-project, Deliverable D9c, May 2002*, 2002.

A Individual work

A.1 Group 1

Below are the individual achievements of each group member of group 1, in alphabetical order. It should be noted that, although it is not extensively dealt with in this report, getting Festival to work the way it should with the extra voices turned out to be no small task, and many hours were spent getting this program to run on a number of different systems, as well as searching for viable alternatives.

A.1.1 Jelle Kastelein

First, after we had divided the hand annotation workload between the members of our group, I annotated two of the fairy tales that were picked. The first was "*Tom Tit Tot*", and the second was "*Mouse and Mouser*". In order to make the annotation both easy and modular, and because, at this point, we were still unsure of what markup we were going to use (SSML or SABLE?), I wrote a first version of a script that searches for [[EMOTION,SPEAKER]] tags, and replaces them by some very basic, fictional, XML-like markup. The hand annotated tags are very simple, and independent of the specific markup or parameter settings, and made the annotated text more transparent. The first version of the script used a markup that we defined, and we expected to convert this to the final markup (e.g.: in SABLE) when we combined them with the emotion parameters. Once it was clear, however, that we were going to use SABLE, I completely rewrote the script in a more principled way, to output SABLE markup directly. Because it seemed like the sensible thing to do, and because it followed naturally from the current find-and-replace implementation, I simply summed the emotion settings of group 2 with the speaker settings (or the average of those settings for an emotion combination) in this script. In order to accommodate for the option of testing the XSL transform of group 2, I also added an option to output ".emsable" markup, which contained explicit <EMOTION> tags, rather than summing the emotions on-the-fly.

After many prior attempts at doing so, and with *much* help from Klara Weiand, who had previously managed to get Festival to work under Ubuntu[11] and had made valiant attempts at doing so under OS X, I finally managed to get Festival to work with the extra MBROLA voices under Cygwin[12]. Getting Festival up and running took quite a bit of time, but once we had it working, the project made rapid progress.

With the script in place, all we really had to do was create, and experiment with the different speaker settings for each of the speakers that the script searched for, and so it was time to start working on our characters. Klara had made a head start on that part, and was of great help, so that the initial settings for each of the characters in the two stories (Tom Tit Tot, the King, the daughter and the mother, and the mouse and the cat, for "*Tom Tit Tot*" and "*Mouse and Mouser*" respectively) were defined quite fast. After that, I did a lot of fine tuning for each character, and I added some statements for word emphasis and some breaks to increase the flow of the text.

To evaluate our system, I created a test file (annotated in the same way as we did the fairy tales) for the other group to hear, in which each of the characters presented themselves in turn. I did this on the basis of a preliminary test made by Klara that was meant for our group alone.

At this point, Liang was facing a personal crisis for which he had to leave for China, and so I did some final refinements on the voices of the characters of the story he annotated, "*The Hillman and the Housewife*", as well as adding some emphasis and break statements.

Finally, I wrote part of the report presented here, in large part in collaboration with Klara, I did a lot of the work on the layout of the document, and I participated in creating the presentation.

A.1.2 Liang Wang

NOTE: Because Liang could not be here for the latter part of the project due to unforseen circumstances, he was not able to provide us with his individual work paragraph. Therefore, we can only provide a summary of what we know he has done. This list may not be exhaustive. We feel that, for the time he was working with us, he showed equal

[11] http://www.ubuntu.com/

[12] Cygwin is a Linux-like environment for Windows (http://www.cygwin.com/).

dedication to this project to both the other members of the group.

- Searched for possible alternatives to the Festival TTS system and SABLE markup language
- Hand annotated *"The Hillman and The Housewife"*
- Created voices for *"The Hillman and The Housewife"*:
  - Hillman
  - Housewife
  - Chimney
  - Servant Girl

A.1.3 Klara Weiand

I started by finding and preselecting the fairy tales that were going to be used in the project. That is, I downloaded a broad selection of fairy tales from Project Gutenberg, decided what properties a fairy tale should have in order to be suited for further use (see section 2.1) and compiled a selection of fairy tales that seemed appropriate. I also added "*A Mad Tea Party*" to the selection, since, while not as traditional a fairy tale as the others, it seemed particularly well-suited for this project. The selection of the final four fairy tales was then done by the whole group. I then used Jelle Kastelein's script and tag system to annotate "*A Mad Tea Party*" with speaker information. I also wrote a script that formatted the annotated text into a clearer format whose functionality was later incorporated by Jelle into the Python script responsible for converting the tags. A big part of my work was concerned with getting Festival and the voices it uses to run. Morphix NLP is a live CD that provides a big number of language and speech processing tools including Festival. However, since Morphix NLP only provides Festival with two voices – none of them female –, I turned to an installed version of Ubuntu Linux and worked out how to install further voices for Festival and MBROLA. The latter was important because none of the voices provided directly for Festival are female which could have posed a problem for creating good voices for the female characters. Through the use of MBROLA, the number of voices we could use increased by several male and one female voice. Once Festival, including the voices was working, I familiarized myself with the software and the SABLE markup language and created the first versions of the voices for the characters in "*A Mad Tea Party*", part of whose definition (i.e. age and species) was later facilitated and improved by the script that Jelle wrote. I then created a first test file that I sent to the members of my group as a very first test of the voices. Since the other group initially used SSML as a markup language, I made the changes necessary to enable Festival to process SSML files and tried to find a way to use SSML to convey the same values for the speakers that were already present in the SABLE files generated by the Python script. This failed however, and both groups later decided to use SABLE rather than SSML. After the Python script was improved to being able to automatically add emotions, and in part generate the basic voice definitions (using age and species information), I fine-tuned my voices and used the wave files produced for "*A Mad Tea Party*" to improve text flow and emphasis in the reading of the story by adding SABLE tags like $< BREAK/ >$. Finally, I wrote the section about the work of group 2 together with Jelle and took part in creating the presentation. Since I found the procedure installing Festival complete with all voices, additional packages and MBROLA to be inconvenient, overly complicated and badly documented, I modified the Ubuntu live CD to include everything needed. The CD allows to run Festival, including all components we used in an Ubuntu system that runs from a CD and thus requires no installation.

A.2 Group 2

A.2.1 Joeri Honnef

Before we actually generate the wave files, which have to be examined at the end, we first have to preprocess the input file. The input file is an XML formatted file. For each sentence which has to be processed we want to add different parameter sets for the different predefined emotions. These predefined emotions reside in another XML file which is used by the XSL transform and acts like a sort of external knowledge source. In this way the XSL transformer knows what parameters have to be applied to sentences with different emotions. This XSL transform, written by Paul, together with an input file and the emotion definitions together are all input for the final processor script which I wrote("process.py"). The script, written in python makes use of a very fast XML binding specially developed for Python (libxml2 and libxslt). The XML library itself is part of the GNOME project and written in the C language, this makes it especially suitable for larger projects when file sizes will increase and speed has to be maintained. Imagine case where whole parts of books have to be synthesized automatically. In these kinds of cases a lot of text has to be annotated with the emotional SABLE parameters and we of course want this to be fast. If it can be done off line, speed is not a really big issue but again think of cases where all has to be done real time. In these sort of situations the annotation process should be fast and that's why I chose for this low level library which is accessable for a high level scripting/programming language such as python. Another argument for the use XML is that it is generally used in the industry as a data container. The main characteristics and properties of XML make it re-useable and at the same time we do adhere to the standards of the W3C. Not only in a industry setting but even more in the scientific field the use of XML enables effective and efficient collaboration. Now more on what I did in this group assignment. In the beginning I, just as all others, was contributing to get Festival to work. Unfortunately a lot of time went into this part of the assignment. But still I learned a lot of all this and realized that also this kind of 'stupid work'has to be done and should not be underestimated. After all these installation procedures, MAKE, environment variables, tarballing, etc. we finally got Festival working. I set up a meeting at my place to do parameter tweaking and listening session to get the right parameter values for expressive speech. That was a whole lot of fun. Finally, at the end of that day, we all were very content with the results.Even the examinations of the other group, regarding the sentences we synthesized, were at a precision level of 100%. It should be noted that this was a very small test group and more reliable results must be obtained by a larger test group. The process script I wrote to perform the XSL transform actually does the following steps in general:

- First it reads in the file onto which the xsl tranform is applied.
- Second, it reads in the XSL transform file.
- Third, we apply the XSL style file to the source XML file
- Finally, the output is written to a command line specified file and all file handlers are closed.

My final task was writing the main introduction of our group work in the report, the section named "Synthesizing expressive speech" and the main introduction of the complete report.

A.2.2 Paul Koppen

Four people needed to work together on a seemingly simple task; adding labels to annotated text that specify parameters to effectuate emotion in synthesized speech. In order to get this task collectively done it was important to divide the approach strategically into sub-parts which could be separately solved by each participant. Also, because everyone was also busy with other projects apart from this course, deadlines were set so progress was ensured. This task of defining sections and managing collaboration was done by me.

During the first weeks it became more and more important to get feedback of our works in the form of sound, but nobody succeeded in getting Festival to run. So a search for alternative speech synthesizers was started. I came up with Microsoft TTS. The advantage of it would have been it's easy integration for scripting. Another good thing was that

it works with XML. The major drawback, though, was that it had not enough parameters to express emotions well. Therefore we rejected it as an alternative. In the end, Rob van Son demonstrated Festival working and soon we had it working on home computers as well.

We requested the other group to deliver their output in a XML format defined by us so that only a single XSL transform would be needed to apply the correct labels for the different emotions. The strategy for the XSL transform was that parameters for emotional variance could be stored in a single XML file so that all the XSL file would only contain transform information whilst the expressiveness of speech was separately adjustable. Because I have much experience in this area I built the XSL transform file together with the structure for the XML settings file. The vocal parameters were adjusted in a meeting at Joeri's place so that we could all agree on the actual values.

A.2.3 Aitor Azcarate Onaindia

Our group had the assignment of defining and adding emotions to sentences. This assignment was split in four sub assignments (the group was of size four). My sub assignment consisted of two parts. The first part was searching and reading publications for the definitions of emotions. The second part was to automatically call Festival with the sable file in python (the programming language we used).

For the first part I searched in Google and found different papers. Most of the papers only defined the emotions in comparison to neutral. By this I mean that the papers defined it as e.g. for sad the speech rate being slower. Thus they did not give numerical data to the different emotions. When continuing with the search I found [1] where such numerical data was given. When testing the different parameters we found that the emotions were correct thus we used this paper for the definitions of the emotions.

The second part was to automatically run Festival with the sable file from python. I tried to install Festival with Cygwin on a windows computer. I was able to run Festival but it didn't do anything (not even the SayText command). Thus I was unable to test if it worked on a windows machine. Although this problem I managed to program that you are able to enter the path to where Festival is installed and then run Festival. For Unix machines I did manage to get Festival to run automatically with a sable file.

A.2.4 Abdullah Zeki Özsoy

First I tried to get Festival to run, like the others in my group did, on Windows and Linux. Including the voices that were requested, by finding and downloading them, or changing the voice in the 'ssm' file did not work. Installing Cygwin on Windows helped get rid of some errors that did not influence the normal text to speech, but getting rid of those errors also did not help with the conversion of annotated text, like Sable or SSML text, to speech generation. If one gets rid of all the errors that are generated we simply get the Festival program to crash when we try to generate a SSML or Sable text to speech. I also found a Graphical user interface for Festival called Carnival, which one can use to annotate speech with Sable or SSML but since Festival was not working this was not very useful.

After that I installed JSML[13] together with its speech engine[14]. This engine had a demo application which would allow you to manipulate four things such as the pitch base, range, volume and the speed of the speech, and generate this speech. During that time I also found a table that explained a bit in non numerical way on how emotions were spoken by humans. I tried to implement this a bit in JSML and tried to run it. This engine also could not generate speech automatically from annotated text.

One of my colleagues installed a Speech Development Kit of Microsoft and recommended it to me. When I wanted to learn more about this engine I found out that this engine could only perform a subset of the functionality of Sable [3]. For instance it could not annotate and so also not generate different pitch ranges. Finally, after the suggestion of the teacher, I tried to install Festival at home in Ubuntu. This worked. After that I tried to implement the emotions with the help of the table Aitor found figure 2 in SSML. After getting an error when I included a pitch range in the SSML

[13] For JSML Specification, see:
http://java.sun.com/products/java-media/speech
/forDevelopers/JSML/Specification.html#14914

[14] FreeTTS 1.2: http://freetts.sourceforge.net/

The effect of emotions on the human voice

	fear	anger	sorrow	joy	disgust	surprise
speech rate	much faster	slightly faster	slightly slower	faster or slower	very much slower	much faster
pitch average	very much higher	very much higher	slightly lower	much higher	very much lower	much higher
pitch range	much wider	much wider	slightly narrower	much wider	slightly wider	
intensity	normal	higher	lower	higher	lower	higher
voice quality	irregular voicing	breathy chest tone	resonant	breathy blaring	grumbled chest tone	
pitch changes	normal	abrupt on stressed syllable	downward inflections	smooth upward inflectio ns	wide downward terminal inflections	rising contour
articulation	precise	tense	slurring	normal	normal	

Figure 1: Effects of emotional speech on the human voice.

file I continued in Sable. In Sable I made a beginning with generating the emotions using these values together with sentences that did not have a meaning from which one could be influenced. Then I tweaked these values a bit according to my recognition of these feelings, and forwarded it to my colleagues for review and demonstration. We then had a meeting as a group were we tweaked these values a bit more. When we were satisfied one of us got a call, we tested our results with this person. We got a positive recognition of all the emotions. We did get a comment that the sad voice was a bit too slow, as if it was going to commit suicide but just not yet, so this caused another small tweak.

Affect Editor parameter values

	Angry	*Disgusted*	*Glad*	*Sad*	*Scared*	*Surprised*
Accent shape	10	0	10	6	10	5
Average pitch	-5	0	-3	0	10	0
Contour slope	0	0	5	0	10	10
Final lowering	10	0	-4	-5	-10	0
Pitch range	10	3	10	-5	10	8
Reference line	-3	0	-8	-1	10	-8
Fluent pauses	-5	0	-5	5	-10	-5
Hesitation pauses	-7	-10	-8	10	10	-10
Speech rate	8	-3	2	-10	10	4
Stress frequency	0	0	5	1	10	0
Breathiness	-5	0	-5	10	0	0
Brilliance	10	5	-2	-9	10	-3
Laryngealization	0	0	0	0	-10	0
Loudness	10	0	0	-5	10	5
Pause discontinuity	10	0	-10	-10	10	-10
Pitch discontinuity	3	10	-10	10	10	5
Precision of articulation	5	7	-3	-5	0	0

Affect Editor parameter values for the six emotion stimuli.

The Affect Editor's parameters are measured on a scale of values that range from -10 to 10. A value of -10 designates the minimal effect of a parameter on speech, while 10 designates its maximum effect.

Figure 2: Detailed numerical vocal parameters for emotional speech.

B Character definitions

Where appropriate, we will place some remarks on each character's most distinguishing characteristics. We will do so grouped by fairy tale, in alphabetical order. Note that, since MBROLA contains only one female voice, the distinct characteristics of the female characters in each fairy tale were created purely by alterations in pitch, volume and speech rate settings.

B.1 The Narrator

The narrator is the character with the most lines, and the only character to appear in every fairy tale. His voice is soothing, slow, and easy to comprehend. Because he is simply an observer, his speech is always neutral in tone.

B.2 A Mad Tea Party

- **The Mad Hatter:** This male character has a mad sounding pitch range.
- **Alice:** This female character's voice is that of a young girl.
- **The March Hare:** This character is quite bossy, and so he speaks rather loudly. His tone of voice is quite strict (he has a high rate of speech), and to the point.
- **The Dormouse:** This male character is constantly sleepy, and as a result has a low pitch range, and little expressive power. He is also a mouse, which means that his baseline pitch will be quite high.

B.3 Mouse and Mouser

- **The Mouse:** This male character is a mouse, which means that his baseline pitch will be quite high.
- **The Cat:** In order to create a feline voice, the cat has a very high pitch range, and talks quite slow. This female character's baseline starts relatively low.

B.4 The Hillman and the Housewife

- **The Hillman:** This male character is a dwarf-like creature, which means that his baseline pitch will be quite high. He has a bit of an accent, and generally talks in a cheery tone of voice.
- **The Chimney:** This turns out to be the same character as the Hillman, but the voice is coming from the chimney, meaning that it's a bit softer, and it is emotionally annotated to be a bit whiny.
- **The Housewife:** This woman is in her mid-fourties, and has a lower voice than the other female character, the servant girl.
- **The Servant Girl:** This is a young adult female, with only one line of text.

B.5 Tom Tit Tot

- **Tom Tit Tot:** This character is described as "a little black thing", which is impish, and reminiscent of a leprechaun. His voice is even higher than that of the Hillman, and his speech is quite fast, always holding a devilish middle between angry and cheery.
- **The Daughter:** This is the younger of the two female characters. Most of the difference between her voice, and that of her mother's, is achieved by using a different age parameter.
- **The Mother:** This woman is approximately the same age as the Housewife, and the two sound a lot alike, although there are subtle differences.
- **The King:** The king has a posh-sounding British accent. Because his life has been comfortable, his voice has had to endure very little abuse, so he has quite a high pitched voice.

C Characteristics of emotions

To generate the emotions we used the pitch base, pitch range, volume and speed parameters.

C.1 Neutral

The values of the parameters of this 'emotion' should all be 0. We increased the pitch range for neutral because sad should have a narrower range. Taking the other emotions into consideration we increased the speed a bit because neutral sounded to slow.

C.2 Sad

This is the most recognizable of these emotions. It should have a very slow speed. We speeded it up because it sounded too slow. This emotion has the lowest pitch range. The volume has to be lower than neutral. The pitch base should be higher than happy. We however did not implement it this way, we made it somewhat high but not the highest pitch base.

C.3 Angry

This emotion should have a high volume and speed. It has a low pitch base with a high range. We adjusted the values in comparison to happy.

C.4 Happy

This emotion is similar to angry. It has high volume and speed, but both have to be lower than with the angry emotion. The pitch base and range should be high.