7
User Interaction

As introduced in the previous chapters, the proposed video-based interactive storytelling system supports two types of user interactions: global and local. Global interactions enable users to suggest events to next story chapters and local interactions allow users to interfere on the execution of the nondeterministic automata. The Suggestion Manager is the main module of the user interaction system and is responsible for managing both forms of interaction, including the process of interpreting and extracting meaning from the suggestions given by users in natural language. While global suggestions are continuously collected by the system, local user interventions occur only at specific points of the narrative in parallel with the global user interaction.

Figure 7.1 illustrates a flowchart of the global and local interaction processes executed by the Suggestion Manager. The global interaction process starts when the module receives from the story generator a set of valid global suggestions represented as simple first-order logic sentences. Once the suggestions have been received, they are then sent to all connected interaction mechanisms to be handled by their respective user interfaces. Then, the Suggestion Manager starts to collect suggestions from the interaction mechanisms. When a new user suggestion is received, it is processed and interpreted through a natural language processing algorithm. The extracted content is then used to update the votes for the global suggestions. This process is executed continuously and, every time the story generator requests a suggestion to be incorporated into the story, the most voted suggestion is sent and removed from the current list of suggestions. The list is maintained and future votes are calculated together with the previous suggestions.

The local interaction process is executed in parallel with the global interaction. It starts when a new set of local interaction options are received from the story dramatization module. Then, similarly to the global interaction process, the interaction options are sent to all interaction mechanisms, suggestions are

collected, interpreted and votes are calculated. However, different from the global interaction, the local interaction process can be interrupted by the dramatization module at any moment. When it happens, the most voted option is selected and the others are discarded. Usually, users have around 15 seconds to decide and vote on desired options.

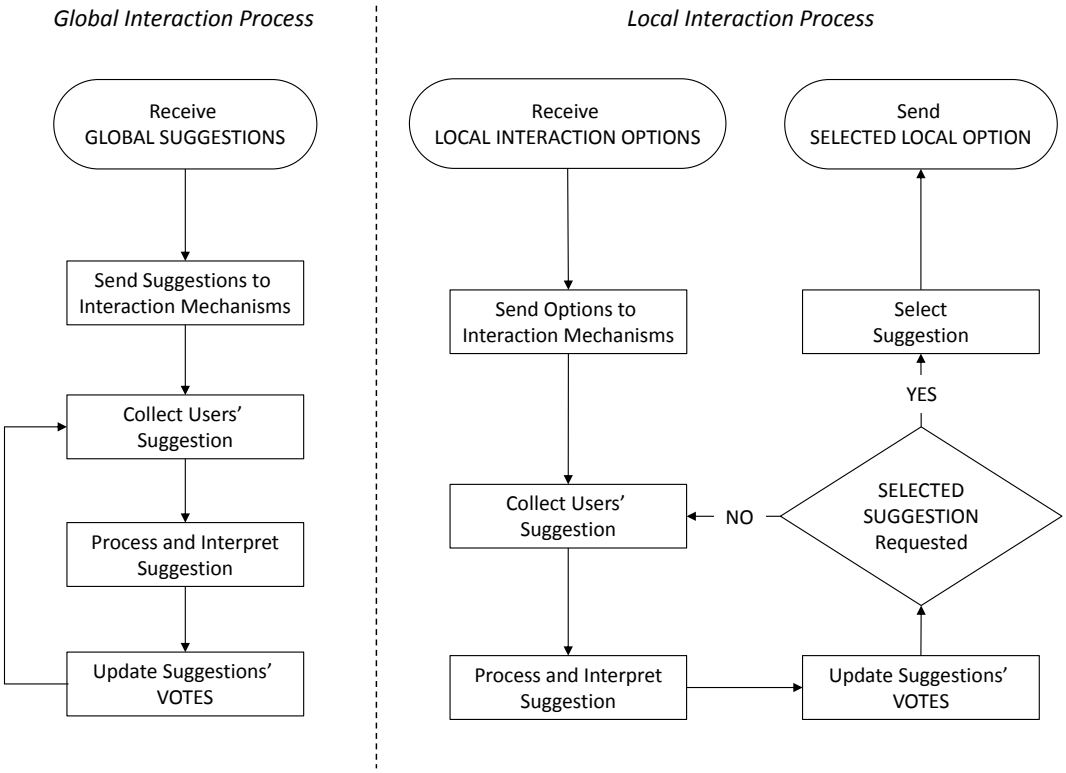


Figure 7.1: Flowchart of the global and local interaction processes executed by the Suggestion Manager module.

The most complex task performed by the Suggestion Manager consists in the process of interpreting and extracting meaning from the users' suggestions, which involves natural language processing. The next section will describe this process and Section 7.2 will present more details about the interaction mechanisms.

7.1. Natural Language Processing

A traditional natural language processing task consists of two main phases (Jurafsky and Martin 2000): (1) syntax parsing, where the syntax tree and the

grammatical relations between the parts of the sentence are extracted; and (2) semantic analysis, which is the extraction of the meaning of words or phrases.

In order to interpret the users' suggestions, we adopted the Stanford Parser to perform the syntax parsing of the sentences. The Stanford Parser (Stanford 2014) is a probabilistic parser that represents all sentence relationships as typed dependency relations instead of using phrase structure representations. However, it also produces phrase structure trees.

The Stanford Parser is capable of producing 55 different typed dependencies (Marneffe and Manning 2008). These dependencies reflect the grammatical relationships between the words. Such grammatical relations provide an abstraction layer to the pure syntax tree and provide information about the syntactic role of all elements. Figure 7.2 (*a*) shows a phrase structure tree generated by the Stanford Parser for the sentence "*The wolf should eat the grandmother!*". The corresponding typed dependencies are listed in Figure 7.2 (*b*). Typed dependencies facilitate the analysis of semantic relationships between words based on both their grammatical relationships and overall sentence syntactical structure.

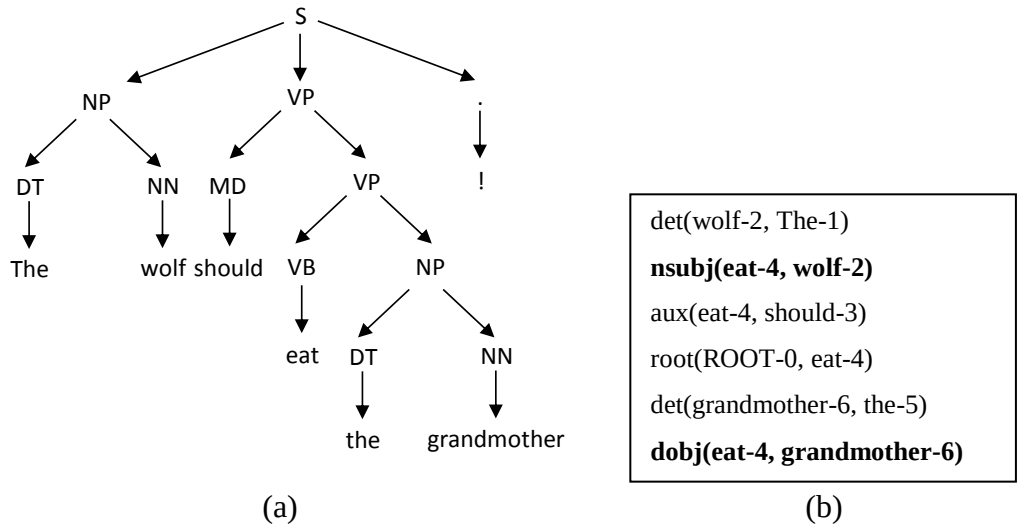


Figure 7.2: Phrase structure tree (a) and the typed dependencies (b) of "*The wolf should eat the grandmother!*".

The typed dependencies are all binary relations, where a grammatical relation holds between a "*governor*" and a "*dependent*". In the above example, the

relation nsubj (nominal subject) relates the noun "*wolf*" with the corresponding verb "*eat*", whereas the relation dobj (direct object) relates this verb with the object "*grandmother*". In this way, the sentence elements are extracted and the sentence structure can be translated into simple first-order logic sentences. In the above example, the following sentence is extracted:

```
eat(wolf, grandmother)
```

which means that the "w*olf*" must perform the action "*eat*" and the victim is the "*grandmother*".

In the present work we generate simple logic sentences composed by a conjunction of predicates, e.g. from "*The wolf should eat Anne and his grandmother!*" is generated the sentence:

```
eat(wolf, anne) ∧ eat(wolf, grandmother)
```

With this dependency chain, the system is able to extract "subject – direct object" relationships from sentences. However, for this pattern to be valid, four conditions must be met: (1) a nominal subject (nsubj) dependency must exist; (2) the dependent of the nsubj dependency must be a family member (in the phrase structure tree); (3) the governor of this dependency must be a verb, which means that a family member is the head noun of the subject of a clause which is predicated by the verb; and (4) a direct object (dobj) dependency must exist and the governor of this dependency must match the index of the governor of the nsubj dependency – then we assume that the dependency of the dobj relation is paired with the family member found initially.

In the example above, the extracted logical sentence already contains the semantic meaning needed by our interaction system to infer a valid suggestion to the story. However, there are some cases where the subjects are not directly referenced. For example, in the sentence "John saves the grandmother and marries her.", the pronoun "her" refers to "grandmother". However, when we compute the typed dependencies for this sentence (Figure 7.3), we see in the relation "dobj(marries-5, her-6)" that the pronoun "her" was not resolved and, in some cases, it's not possible to solve it using only the phrase structure tree. The process

of resolving what pronoun or a noun phrase refers to is called anaphora resolution. To solve this problem, we used another tool from the Stanford Natural Language Processing Group, the Stanford Deterministic Coreference Resolution System (Raghunathan et al. 2010), which is able to indicate precisely the correct reference of any unknown pronoun.

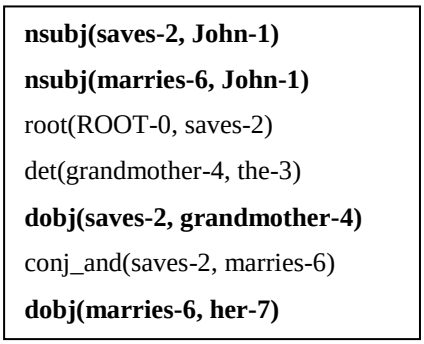


Figure 7.3: Example of anaphora problem in the sentence "*John saves the grandmother and marries her*".

The parser also verifies the occurrence of negations. For example, in the sentence "*The wolf should not eat Anne!*", the adverb "*not*" completely changes the meaning of the sentence. To identify negations, the parser analyses the occurrence of negation modifiers ("*neg*") in the typed dependency list. Figure 7.4 illustrates the typed dependency for the example above and the occurrence of the negation modifier.

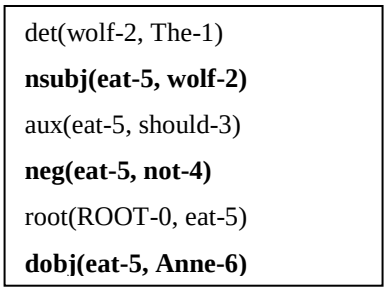


Figure 7.4: Example of negation in the sentence "*The wolf should not eat Anne!*".

After translating the "subject – direct object" relations into first-order logic sentences, the parser also needs to validate the sentences. For example, the predicate "$eat(CH_1, CH_2)$" requires a nominal subject CH_1 that is a valid character

and a direct object CH_2 that also is a valid character in the story context. Moreover, the verb "*eat*" also must be a valid action. To perform this validation, the parser has access to a list of valid actions, characters and places. In this way, the parser is able to identify the elements that the words represent. However, almost all words have synonyms and to deal with this, the parser also incorporates a dictionary of synonyms associated with each action, character and place. So, it is able to parse sentences such as "*John should annihilate the villain!*", where the verb "*annihilate*" is a synonym of the action "*kill*", and the object "villain" the role of the character "Wolf".

Ideally, the parser expects sentences that contain at least one verb, one nominal subject and a direct object. However, this does not always happen; in some cases the subject, the direct object, or both are omitted. For example, the sentence "*Kill the wolf!*" does not express directly who should perform the action "*kill*", but indicates the direct object "*wolf*" (Figure 7.5). In this case, the parser is still able to generate a partial logic sentence to represent it:

```
kill(*, wolf)
```

which means that someone "*" must perform the action "*kill*" and the victim is "wolf". The operator "*" can be replaced by any valid character to complete the logical sentence.

root(ROOT-0, Kill-1)
det(wolf-3, the-2)
dobj(Kill-1, wolf-3)

Figure 7.5: Example of omitted subject in the sentence "*Kill the wolf!*".

The entire process of extracting valid first-order logic sentences from text phrases is illustrated in Figure 7.6. In the Syntax Parsing step, the Stanford Parser receives a text phrase S_x as input and generates a Dependency Tree and the Typed Dependencies for the sentence. Using this information, in the Semantic Analysis phase, the parser performs the Anaphora Resolution process to resolve the pronouns of the sentence and find valid synonyms using the Synonym Dictionary.

Finally, the parser checks the integrity of the sentences using some Logic Rules and returns a list of valid first-order logic sentences (P_x^n).

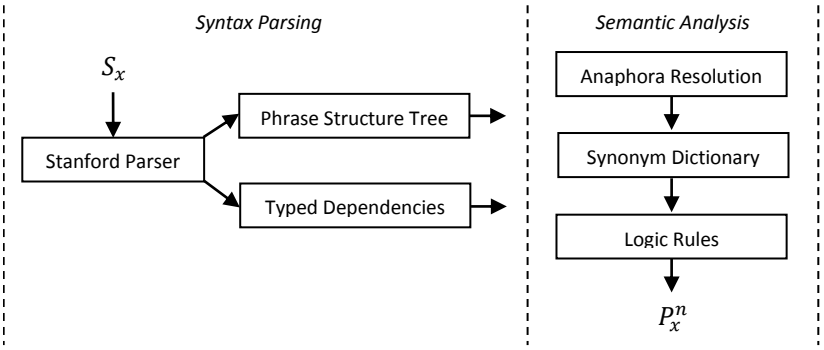


Figure 7.6: The process of extracting valid first-order logic sentences. S_x is the input text phrase and P_x^n is the output list of predicates.

Another way of identifying and extracting meaning from natural language sentences is through an analysis of the user satisfaction with some target concept. The process of extracting user's satisfaction also involves natural language processing, more specifically the area of Sentiment Analysis (Liu 2010). However, instead of using complex sentiment analysis techniques, we adopted a more simplistic approach to solve this problem. We describe the target concept as simple questions based on possible suggestions to the story (e.g. "*Would you like to see the Big Bad Wolf attacking Little Red Riding Hood in the next chapter?*"). Users usually respond to the questions positively or negatively, i.e. agreeing or disagreeing. In this way, the parser only needs to identify positive and negative answers in the users' comments.

The approach adopted by our parser to identify positive and negative answers uses a list of words, where each word W_i is associated with a numerical score $W_i^s \in [-1.0, +1.0]$. High negative scores represent very negative words and high positive scores represent very positive words. Considering C_x a user commentary, the sentiment $St(C_x)$ is given by:

$$St(C_x) = \frac{1}{n}\sum_{i=1}^{n} W_i^s \quad if \quad (W_i \in C_x)$$

where $St(C_x) \in [-1.0, +1.0]$ indicates if C_x is a positive commentary ($C_x > \beta$) or a negative commentary ($C_x < \alpha$), in which α and β defines a precision threshold where uncertain commentaries are ignored (classified as neutral).

To illustrate this process, let's consider $= -0.3$ $\beta = +0.3$, and the following user commentaries for the suggestion $S_1 =$ "*Would you like to see the princess Marian dying in the next chapter?*":

1. "*Yes!!* :)"
2. "*I would love to see it happening!!!* ;)"
3. "*No!! I love her…* :("
4. "*This story is boring…* :("

For case (1), the word "*Yes*" and the emoticon ":)" have both the score +1.0; giving the sentiment $St(C_1) = +1.0$ and classifying it as a positive commentary. In case (2), the word "*love*", "*see*" and the emoticon ";)" have the scores +0.8, +0.5 and +0.9 respectively; giving the sentiment $St(C_2) = +0.73$ and classifying the sentence as a positive commentary. In case (3), the word "*No*", "*love*" and the emoticon ":(" have the scores -1.0, +0.8 and -1.0 respectively; giving the sentiment $St(C_3) = -0.4$ and classifying the sentence as a negative commentary. Finally, in case (4) the word "*boring*" and the emoticon ":(" have the scores -0.7 and -1.0 respectively; giving the sentiment $St(C_4) = -0.85$ and classifying the sentence as a negative commentary.

7.2. Interaction Mechanisms

The user interaction module of the video-based interactive storytelling system works as a multimodal and multi-user interaction server that supports the integration of several interaction mechanisms based on suggestions. Each interaction mechanism acts as a multi-user server that has its own client interface, allowing several users to be connected in the same interaction network.

Two interaction mechanisms were integrated with the user interaction module: social networks and mobile devices. The first method is based on the idea of using social networks (such as Facebook, Twitter and Google+) as a user

interface, allowing users to collaborate with the development of the stories in a social environment. The second interaction mechanism combines the use of mobile devices (such as smartphones and tablets) with natural language to allow users to freely interact with virtual characters by text or speech. The next sub-sections will present more details about these interactions mechanisms.

7.2.1. Social Interaction

The social interaction interface is implemented through the application programming interface (API) provided by the social networks. The module is constantly sending messages through the social networks to induce facts to users or to provoke them, which we denominate "induction messages". When the system starts or when a new chapter is beginning, the induction message is an introduction to the story or chapter. Users receive this message as an update in the social network and are able to comment on the message (Facebook and Google+) or use hashtags (Twitter) to indicate suggestions. The introduction message describes the story characters, places, gives some tips about what could happen in the story and incentive the users to comment what they would like to see happening in the story. Figure 7.7 shows an example of introduction message used for fairy tale interactive story.

"*Once upon a time there was a little girl, named Anne, but mostly known as Little Red Riding Hood. She lived in a certain village with her mother, who was excessively fond of her; but always concerned about the health of Anne's grandmother, who lived in distant village. Not far away in a sinister forest, lurched the Big Bad Wolf, the evil wolf ready to eat anything that fits in his mouth. But there was also a brave woodcutter John, ready to save everyone. Uncountable stories can be told in this world of fantasy. Will Little Red Riding Hood be attacked by the Big Bad Wolf? Or killed by the monster? Will the grandmother be eaten by the wolf? Will the woodcutter save everyone?*

A new interactive story is about to begin. Comment here what you would like to see happen with the characters of this story."

Figure 7.7: Example of an introduction message.

During the dramatization of a chapter, the system keeps sending induction messages to the social networks using the global suggestions for the next chapters received from the Suggestions Manager. In this case, the induction messages have the form of simple questions (e.g. "*Would you like to see Little Red Riding Hood hitting the Big Bad Wolf in the next chapter?*"). When the social interaction module receives local interaction options, a special induction message is created in the form of an invitation for a poll (e.g. "*Little Red Riding Hood should trust the strange wolf? Yes or No?*"). In this case, a poll is created in the social networks and users are able to select and vote on the desired outcome by clicking on the poll option. Figure 7.8 illustrates the dynamic behavior of the social interaction system through an activity diagram.

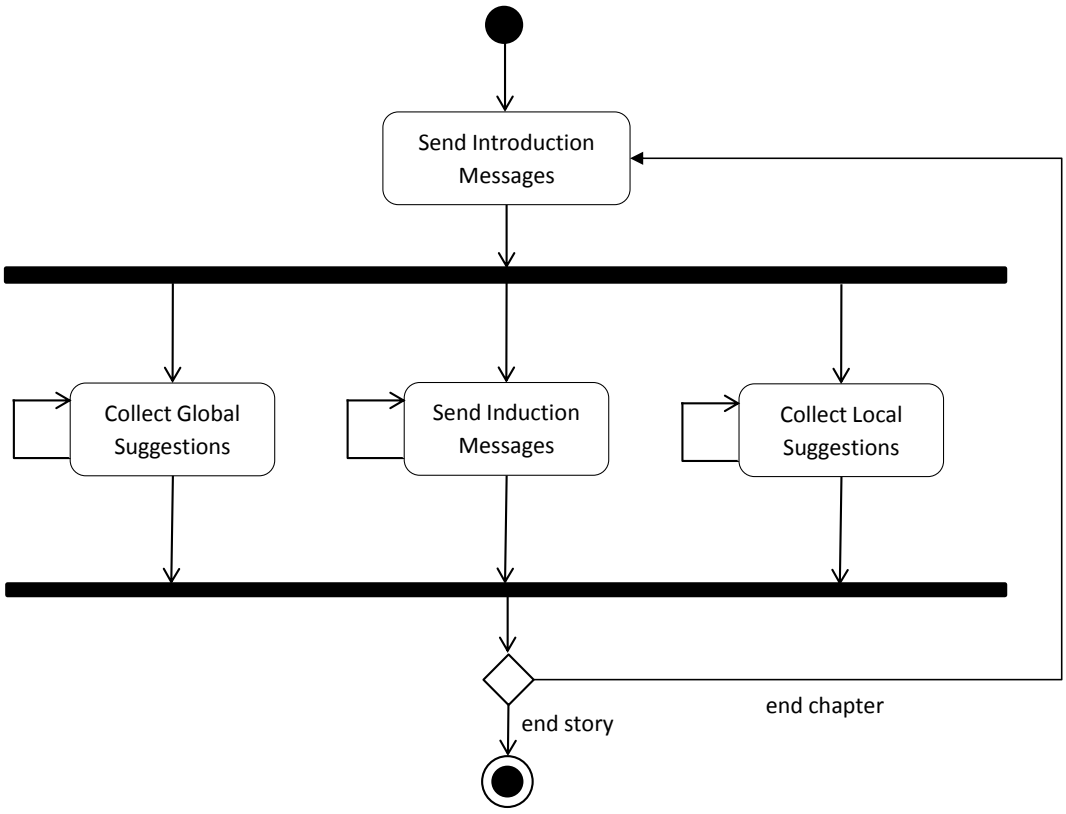


Figure 7.8: Activity diagram of the social interaction module.

There are three basic ways users can interact with the stories through social networks: (1) interaction by comments – where they explicitly express their desires through comments in natural language; (2) interaction by preferences – where they express satisfaction or state preferences; and (3) interaction by poll –

where a poll is created and users vote in what they want. The next sub-sections present more details about these interaction methods.

7.2.1.1. Interaction by Comments

The interaction by comments allows users to explicitly express their desires through comments on the social networks (Figure 7.9). Every time the system detects a new user comment in an active introduction or induction message, the content of the comment is extracted and sent to the Suggestion Manager to be interpreted and counted as a vote to the expressed suggestion.

Figure 7.9: Example of user comment expressing a suggestion on Facebook.

In the case of Facebook and Google+, besides writing comments, users are also able to “like” or “+1” a comment of another user, which indicates that they liked what the comment says. In this way, the interaction system considers the number of users that directly wrote that something should happen and the number of users that liked the respective comments.

7.2.1.2. Interaction by Preferences

The interaction by preferences allows users to express their satisfaction with the story suggestions through social networks. Instead of directly writing a

comment expressing a desire, users are able to “like” (Facebook) or “+1” (Google+) a suggestion generated by the interaction system (Figure 7.10). Users can also write comments on the generated suggestions expressing their satisfaction with the proposed events.

Figure 7.10: Example of user “liking” a system generated suggestion on Facebook.

Every time the system detects a new user comment in an active induction message that expresses a global suggestion, the content of the comment is extracted and sent to the Suggestion Manager to be interpreted by the sentiment analysis algorithm, which will classify the comment as positive or negative. Positive comments count as positive votes to the suggestion described in the post and the negative comments count as negative votes. The number of users that “like” (Facebook) or “+1” (Google+) the suggestion also count as positive votes.

7.2.1.3. Interaction by Poll

The interaction by poll allows users to choose what they want through polls in the social network. Instead of directly writing a comment or waiting for the desired suggestion to appear (posted by the interaction by preferences), they are able to see all available options and vote on the suggestion of their choice (Figure 7.11).

The interaction system is constantly checking the results of the active polls, and informing the Suggestion Manager about the most voted suggestions. The process of extracting users’ choices from a poll does not require any complex algorithm. However, the importance of this method should not be underestimated, because it provides an easy way of interaction where users who do not like to

write or do not know exactly what they want are able to interact just by clicking on a poll option.

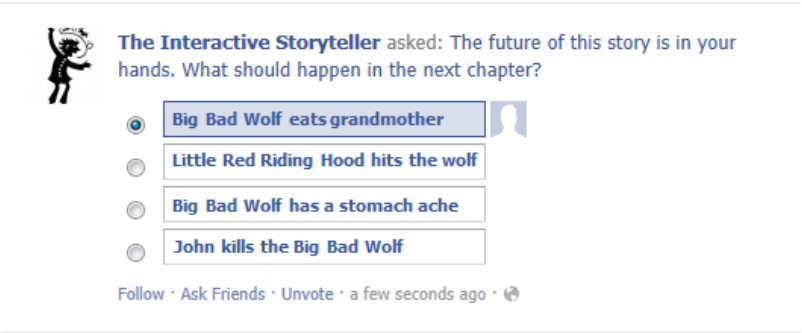


Figure 7.11: Example of poll with story suggestions generated by the system on Facebook.

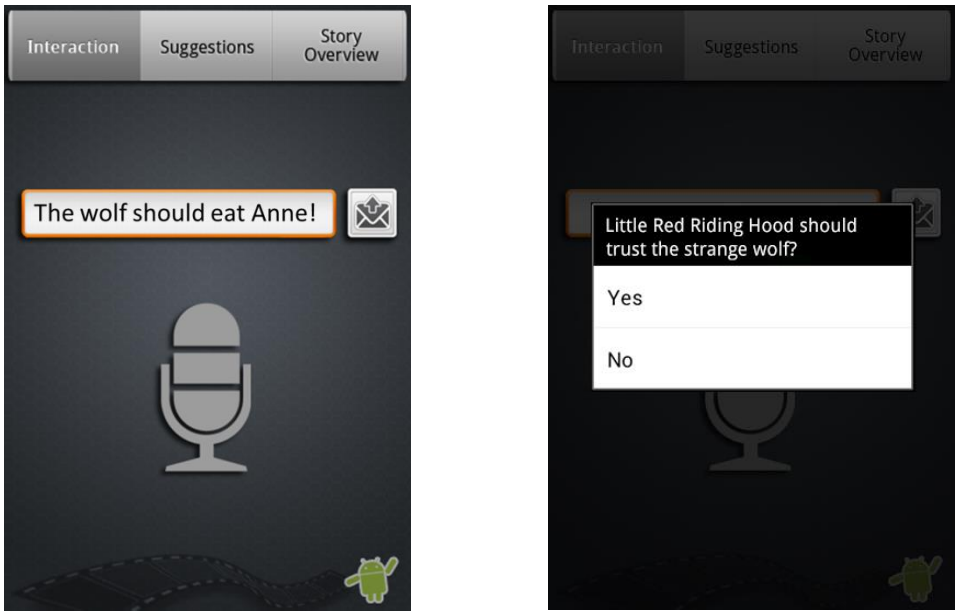
7.2.2. Mobile Interaction

The multi-user mobile interaction interface was designed to support both global and local user interactions. It incorporates spoken and written natural language in a simple mobile application, where users can write or speak what they want to happen in the story or easily select the desired outcome for local decision points. With this interaction method it is easy to imagine the possibility of watching a movie while advices are continuously being sent to the characters.

The mobile interface consists of a small application developed for Android mobile devices (such as smartphones and tablets), where users can interact with the ongoing stories by writing or speaking a suggestion/advice to the virtual characters using natural language. The user interface of this application is shown in Figure 7.12.

Users can interact with the story by typing the suggestions and advices in the text box shown on the mobile interface or by pressing the microphone icon and then speaking out the intended suggestion. The Android Speech Recognition API is used to recognize the user speech and to convert it into text. In this way, the system only needs to handle and understand written text. After reading the user input (text or speech), the mobile application sends the user suggestions through a TCP/IP connection to the Suggestion Manager, which is responsible for interpreting and managing all user suggestions.

The mobile application is also in charge of handling local user interactions. When the application receives local interaction options it displays the list of choices on the screen and alerts the user by vibrating the mobile device. Users are able to touch the desired option on the screen or speak out the intended choice (Figure 7.12 - b). The selected option is then sent to the Suggestion Manager to be counted as a vote to its respective outcome of the local decision point.



(*a*) (*b*)

Figure 7.12: Mobile user interface. Image (*a*) shows the main screen of the mobile application; and image (*b*) shows the interface during a local interaction.

7.3. Conclusion

This chapter described the technical details about the implementation of the user interaction module of the video-based interactive storytelling system and the multi-user interaction mechanisms using mobile devices and social networks.

The mobile user interface allows users to freely interfere in the narratives using their personal mobile devices, while the social interaction allows users to do the same in the multi-user environment of social networks. It is important to notice that the social interaction interface is dependent of the social network services, which are constantly changing the way external applications can interact

with their services. Consequently, the social interaction module of our system requires constant updates to adapt to these changes. In addition, it faces the risk of being unable to perform its tasks if an update in the social network API blocks its access to the required resources.

The next chapter describes the interactive narratives produced to validate our system and the technical tests performed to evaluate the algorithms used in the dramatization and in the user interaction modules of the proposed system.