Supporting Dialogue Generation for Story-Based Games

Christopher Kerr, Duane Szafron

Department of Computing Science, University of Alberta
Edmonton, AB, Canada
{kerr, duane}@cs.ualberta.ca

Abstract

Providing compelling, realistic, immersive game worlds is one of the major goals in modern game design. The presence of unique and interesting dialogue for all of the characters in a game enhances this sense of immersion. In this paper we present the concept of an "Intentional Dialogue Line" that supports the efficient generation of multiple variations of a dialogue, where these variations are both unique and appropriate to the character who is speaking them. This paper focuses on how machine learning can be used to quickly populate the intentional dialogue lines with existing content.

Introduction

Imagine that you have just stepped into a tavern in a small town. You are here because this is where, according to the locals at least, you are most likely to find a little action. However, instead of a lively establishment full of patrons enjoying themselves, what you have found is more akin to what someone might expect to see in a wax museum. In place of lively patrons you find stiff unmoving people who all seem to repeat the same set of two or three phrases. It is likely that you would find the whole situation quite unnerving and quickly make your way out of the tavern. While this may sound as if it is a scene out of a cheap horror film, it is in fact a scene found all too often in the majority of story-based video games.

The idea of *immersion* has become a hot topic within the games industry in recent years as designers strive to create more believable worlds for their games (Hocking 2008). By populating the game world with characters who behave in a believable manner, the designers enhance the player's ability to suspend disbelief. An increased sense of immersion allows the player to more fully enjoy the gaming experience. In addition, the inclusion of realistic non-player characters (NPCs) in the game world can also provide designers with an increased number of creative options. Designers are able to influence the atmosphere of the game through the ambient behaviors of the NPCs. In our previous example, if prior to entering the tavern the game had featured lively realistic characters, then the behavior of the NPCs within the tavern would seem all the more unsettling. By ensuring that NPC behavior is appropriate and believable throughout the game, the designers could use lack of behavior as a plot device.

Unfortunately, the creation of realistic NPC behavior is not without its costs. The creation of unique and appropriate behavior for the multitude of NPCs, especially secondary NPCs, found in a game is often considered to be prohibitively time-consuming and expensive. What is needed is a means of efficiently and effectively creating realistic character behavior for the NPCs who populate the game world, reducing the cost of making the game more immersive. A serious content bottleneck with realistic behaviors exists that is analagous to the graphics content bottleneck of ten years ago. To solve the behavior content bottleneck, we need abstractions that parallel the *triangle-model-texture map* abstraction that solved the graphics bottleneck by representing graphics as data-driven content rather than procedural algorithms (Hecker 2008) (Rabin et al. 2009). There are several aspects of the realistic NPC behavior problem, each requiring a set of abstractions. In this paper, we tackle one very important component of immersive behavior – varied and contextually-sensitive dialogue. We introduce an abstraction, the *intentional dialogue line*, to facilitate the efficient creation of appropriate, interesting dialogue for an arbitrary number of NPCs.

Following a brief discussion of some related work, the remainder of this paper details the specifics of an intentional dialogue line, as well as some tools to facilitate this abstraction. We discuss how intentional dialogue lines are translated into actual dialogue within the game. Following that, we present a machine learning method for quickly populating intentional dialogue lines with user created content and then discuss the results of our machine learning technique.

Related Work

The existing work on facilitating efficient dialogue generation falls into two categories. One category focuses on the development of tools to better represent game dialogue and tries to simplify the dialogue creation process. The results of this work can be seen throughout the games industry with tools like BioWare Corp's *Aurora Toolset* for the game *Neverwinter Nights* (NWN 2009), or Bethesda Softworks' *Elder Scrolls Construction Set* for *The Elder Scrolls IV: Oblivion* (Oblivion 2009). This category also includes ongoing research to improve these kinds of tools (Siegel and Szafron 2009). While such tools ease the

process of creating dialogue, they do not reduce the amount of dialogue that must be authored by the designers. It is analogous to having excellent tools for creating graphical game objects while lacking the abstractions that allow the same model to be re-used with different textures.

The other category of work on game dialogue uses a variety of natural language processing techniques to wholly generate game dialogue. Examples of this approach can be found in Kacmarcik's NPC dialogue generation (Kacmarcik 2006) and the free form dialogue found in Façade (Mateas and Stern 2003). While this approach is perhaps the most exciting, it also faces the greatest challenges. Much of the dialogue these approaches generate is simplistic and rigidly structured. Pure generation of game dialogue wrests too much dialogue control away from the game designers, who are tasked with ensuring the game's entertainment value and integrity. Additionally, there is a great deal of difficulty in assigning ratings, such as those of the ESRB, to games in which the dialogue that is presented to the player is not fully known prior to game time. Our approach is a hybrid of these two categories. We propose tools that not only support dialogue created by game designers, but also help to generate dialogue based on a designer's own style and present this dialogue to the designer for approval before the game ships.

The Intentional Dialogue Line

Dialogue in a story-based game generally consists of a set of player character (PC) and NPC lines grouped into *exchanges*. At any given point in the dialogue, an NPC will speak a line, and then the player will choose one of potentially many responses. This will often be followed by a series of other exchanges of NPC and PC lines until the end of the conversation is reached. Such a structure is clearly visible in the dialogue trees of games like *Neverwinter Nights* (NWN 2009) and *Mass Effect* (Mass Effect 2009). However, even games such as *Oblivion* (Oblivion 2009) that lack tree-based dialogue tend to resort to exchanges.

The lowest level element of a dialogue is the dialogue line. A *dialogue line* may consist of any statement, question, or other similar element that the designer wishes a character to say. Consequently, the set of all potential dialogue lines is massive. However, not all lines occur with equal frequency in an average dialogue. If one considers the purpose or intent of a line, instead of the exact text, some patterns arise. Perhaps the most obvious examples are the *greeting line* and *farewell line* of most dialogues. While the exact text will differ, all greeting lines share the same intent. We propose a system in which any dialogue can be composed of intentional dialogue lines. These intentional lines can be converted to actual lines at game time based on some characteristic, such as the mood of the character speaking the lines. We are not proposing game time generation of novel lines, but instead the dynamic binding of the intentional line to an actual line that shares its intent and meets the situation's requirements. This approach allows NPCs to relate similar information using different phrasing, helping to eliminate the drone effect described previously. The PC lines in *Oblivion*'s (Oblivion 2009) wiki-style dialogue system can be thought of as "intentional" dialogue lines, though they are not replaced with an actual dialogue line at game time. Similarly, the short synopsis lines that the player is presented with in *Mass Effect* (Mass Effect 2009) can also be thought of as the "intentional" lines, though in this case each "intentional" line has only a single actual dialogue line associated with it.

The question is how to create these intentional dialogue lines, and how to dynamically map the intent to an actual line during the game. One idea is to have the designer simply write a dialogue line with the given intent, and then perform some automatic paraphrasing to morph the line into a variant with the desired traits. Unfortunately, there are some major difficulties with this approach. In order for the NPCs in a game to be believable, they must speak in a manner that is both fitting to their character and varied from other characters. Modern paraphrasing techniques are simply not sophisticated enough to accomplish this task (Lin and Pantel 2001). While it may be possible to achieve more desirable results through the use of domain-specific, rule-based morphing algorithms, it is not clear that doing so will provide the degree of variation and suitability that is required.

How then should one approach the task of creating intentional dialogue lines? Instead of a system based on automatic generation of morphed lines, we have created a system focused on both flexibility and control. Designers can manually generate the variations to be associated with an intentional dialogue line or they can use automatic methods to generate the variations. Regardless of the method of generating the dialogue lines themselves, we provide a number of tools to support the automatic, dynamic replacement of intentional lines with appropriate actual dialogue lines based on the context in which they are being displayed (i.e. game state). We feel that this approach provides the greatest deal of control over the final result while still greatly reducing the effort required to create similar, but varied, dialogues.

An *intentional dialogue line* is an abstract line with a specific intent, such as a greeting line or a farewell line. We then associate a number of actual dialogue lines that share the intent of this abstract line. For example, three such lines with the greeting intent are: "Hello", "What reason could you possibly have for disturbing me?" and "Aye, and a fine day to you too laddie." The benefit of such an approach is that while there are potentially a very large number of characters that need to deliver a greeting line, it is very unlikely that each and every character would use a different greeting. As such, while initial uses of a specific intentional dialogue line may require the designer to create a new variation most times, later uses of the line should be possible without the need to create any new content.

However, simply having a collection of similar dialogue lines is not enough to allow us to use an intentional dialogue line in a game. We also need a means of selecting which of the actual lines should be selected at game time. For this we need two things: one is a means of partitioning the actual lines based on some characteristics, and the other is a set of rules that allow us to map game state to one of these partitions and therefore support the selection of one of the appropriate actual lines in that partition.

Mapping from Game State

How do we decide which actual dialogue line to use in place of the abstract intentional line at game time? Our goal here is to ensure that lines spoken by a character at game time are appropriate, while also providing variation between similar characters. We must first have a way of differentiating the actual lines from one another based on some characteristics of the lines. To accomplish this, we use a multidimensional grid with one dimension for each characteristic and three values for that characteristic: low, medium and high. For example, if there are two characteristics of interest, sophistication and disposition of the speaker, we would use a 3x3 grid. Such a grid is shown in Figure 1.

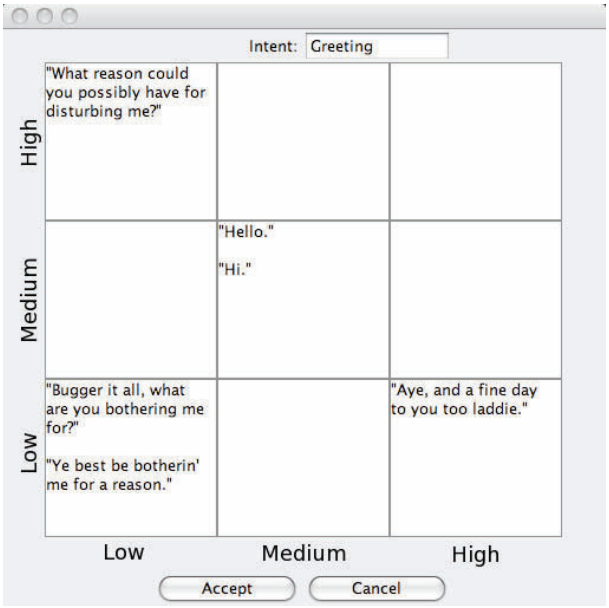

Figure 1. An intentional Greeting line and a partitioning of its actual lines on two conceptual axes – sophistication (vertical) and disposition (horizontal).

With this set of bins to hold the actual dialogue lines, there remain a few problems to solve. First, we must consider how to place the lines within the bins. For now we will consider a straight-forward, user-driven approach in which the author of the intentional dialogue line manually chooses the bin in which an actual line should be placed. This provides the maximum level of user control, while not precluding the future addition of automated methods that are described later in this paper.

With the bins of our intentional dialogue line populated, the problem becomes how to select, at game time, the specific actual line that is used in a given situation. In order to provide believable characters it is important that the specific line that appears in their dialogue is appropriate. Returning to the example of Figure 1, we should consider various elements of game state that would tell us about the sophistication and disposition of the character speaking the line. However, there are many ways in which to go about deciding on how to map game state to these characteristics. In the case of disposition one might consider character attributes of both the speaker and listener. Such attributes could include charisma, various speech-related skills, racial differences between characters, and countless other possibilities limited only by the game engine and the imagination of the designer. A fixed set of rules for mapping game state to our bins will not provide the necessary expressive power.

In order to provide the necessary flexibility in defining the relationship between game state and the bins of the intentional dialogue line, we need a set of dynamic mappings, where a different dynamic mapping can be assigned to a different character or type of character. Such dynamic mappings are able to capture the differences between characters. This allows us to use a common set of lines, in the bins, from which each character will select only the lines that are appropriate. For example, a designer may assign a “racial difference” mapping to the disposition axis of a group of townspeople. If the race of the NPC speaker and PC are the same, the disposition of the NPC speaker is high. If the race of one is human and the other is non-human, the disposition of the NPC speaker is medium. Finally, if the races of both are non-human and different, the disposition of the NPC speaker is low. This is a dynamic mapping as it varies between NPC speakers depending on their race (and varies across PCs in a multiplayer game or in replays with different PC races). However, it is only one of a set of dynamic mappings, since there may be another “occupation” mapping where those NPCs who receive this mapping have dispositions based solely on their occupations. The designer selects different dynamic mappings for different groups of NPCs based on the story.

One implementation of these sets of dynamic mappings is based on dialogue filters (Siegel and Szafron 2009). A filter is essentially a script attached to a dialogue line that determines whether the line will be displayed to the player. However, filters are presented to the designer in a manner that allows them to create filters without specific programming knowledge. These dynamic mappings can be reused in other contexts to support the creation of complex mappings within seconds. To implement intentional dialogue lines in *Neverwinter Nights* (NWN 2009), our system includes all associated actual dialogue lines in place of each intentional dialogue line. Each actual dialogue line

has a filter attached. At game time the filters then determine which of the lines is presented to the player. Changes in PC or NPC characteristics are reflected in the lines chosen by the filters. However, from the designer's perspective we are able to hide this complexity, presenting them with only the set of bins and the filters for determining how to map game state to each axis. In this way, we are able to provide the necessary power and control while maintaining ease of use.

Using Learning to Populate Bins

While filters are a convenient and efficient method to dynamically map intentional lines to actual lines, we need an efficient method of populating the intentional line with its variants. While the manual approach mentioned before does provide a great deal of control, it also requires a great deal of effort. When transitioning to our system one would like to be able to take advantage of existing content. However, going through dialogue files to select lines with the desired intent and sort the selected lines into the bins is time-consuming. While we do not provide an automated method for identifying the intent of a line, we do present a machine learning approach to automate the sorting of the selected lines into bins.

Consider the 3x3 sophistication/disposition grid in Figure 1. In order to determine which bin to place a dialogue line in, we must categorize it based on these two properties. One approach would be to build a single classifier that assigns one of nine labels to each line. However, such an approach lacks scalability, requiring a unique classifier for each combination of potential axes. Additionally, the number of outputs of such a classifier would increase exponentially with the inclusion of additional axes (3^n outputs for n axes). We instead propose a more modular approach in which each axis is classified independently. Doing so provides greater extensibility with regards to the addition of new axes, as well as the ability, if necessary, to pursue different classification technologies for different properties. In this paper we describe our experience with building a classifier to determine the sophistication of a dialogue line. A similar classifier could be designed for each characteristic of interest to designers.

While there are a large number of approaches one could use when developing a natural language classifier, we used a support vector machine due to the effectiveness it has shown in identifying sentiment elsewhere (Annett and Kondrak 2008). More specifically, we used the support vector machine algorithm (C-SVM) implemented in the LIBSVM library (Chang and Lin 2001). We used a linear basis function, set the cost of constraint violation to 1.0, and set the stopping threshold to 0.001. Prior to selecting the parameters, we tried many different settings, but found that these provided us with the best results.

In addition to a classifier algorithm, we need to select a good set of features. We used a feature vector with one slot for each word in a selected corpus – the bag-of-words approach. The grammar and word order of a line are ignored. This approach is desirable because it greatly reduces the size of the feature space. We used two different bag-of-words implementations.

One approach is to use binary values in our feature vectors. The value of a specific feature is 1 if that word is present in the line and 0 otherwise. This approach has the benefit of not requiring any knowledge of the corpus as a whole, but lacks any information on multiple occurrences or the importance of a word.

The second approach is to use TF-IDF (term frequency – inverse document frequency) (Salton 1991). TF-IDF is a measure of how frequently a word occurs within a given line, compared to how many lines within the corpus contain the word. An increase in the number of occurrences of a word within a line will cause an increase in the value of the feature. In contrast, an increase in the number of lines throughout the corpus that contain a word will cause a decrease in the value of the feature. In this way, we are able to represent information about both the number of occurrences of a word within the line as well as the overall importance of the word.

Additionally, we also use a set of features measuring the average number of words per sentence in a line (a dialogue line may contain multiple sentences). In order to increase effectiveness, we use a set of binary features in place of a single feature with a floating point value. The average number of words per sentence within a line is calculated, and then the result is rounded to the nearest integer value (all values over 10 are rounded down to 10). The features correspond to these 10 integer values, with a value of 1 being assigned if a line has that average number of words, and a 0 assigned otherwise.

Constructing a Training Set

A classifier technique requires data to train and a method of evaluation. However, as one might expect, there is no available set of game dialogue lines that are pre-labeled for sophistication. One solution would be to write a collection of lines ourselves. However, it would be best if the lines used are representative of the lines found in real games and this requires the use of professionally written dialogue. While it is possible to collect a set of unlabeled dialogue lines directly from games, the tendency for game dialogue to be stored in proprietary file formats makes automatic collection of the lines difficult. The time required to collect the dialogue lines manually through the game, as well as the time required to have the set of lines labeled by a jury makes this approach fairly expensive. However, this approach should also provide the highest quality labels. As such, this is the best approach if one is only concerned with the quality of the trained classifier, and not the time required to produce it.

Due to the cost of the jury approach, we also propose a more automated solution. While game dialogue is not easy to come by, movie dialogue is. The linear nature of a film, in contrast to a game, means that the dialogue is written in scripts that can be easily parsed. In addition to being

readily available, film dialogue is professionally written and story-based so it shares many characteristics of game dialogue. However, even with the task of collecting the dialogue lines now automated, the time required to label the collection is still a problem. Therefore, we propose a semi-automated solution. Dialogue lines from a movie script come with an additional piece of information, beyond the text of the dialogue line itself: the name of the character speaking the line. While the number of individual dialogue lines within a script can be quite large, the number of characters in a film is generally relatively small. By assigning a fixed characteristic label (e.g. low, medium or high sophistication) to each character, we can propagate those labels to the lines the characters speak. This allows us to obtain a labeled set of dialogue lines very quickly.

Unfortunately this semi-automated approach is not without potential flaws. While a character may generally speak at a certain level of a characteristic (e.g. sophistication or disposition), it is unlikely that the character will always speak at that level of that characteristic. Therefore, there is a certain level of unreliability to the labels that are generated. In addition, even without the issue of label reliability, there is no guarantee that movie dialogue is directly transferable to game dialogue. These concerns must be addressed when evaluating a classifier based on semi-automatic labeling of film dialogue.

Evaluation

We evaluated the results obtained from a classifier trained on a manually labeled (juried) set of training data from games, as well as a set of training data from movie scripts labeled using our semi-automated method. For the game data, we used 5-fold cross-validation, aggregated over 10 trials to account for the randomness in fold selection. However, for the classifier trained on movie data, simple 5-fold cross-validation does not address the concerns noted previously. As such, we trained the classifier on the set of movie lines and then used the manually labeled game data to evaluate the classifier. Here the aggregating of multiple trials was unnecessary as there is no random fold assignment.

The best results would be obtained by an approach in which both the precision and recall of the predicted labels are high. Such a requirement is captured in the F-measure, which is the harmonic mean of the precision and recall. If the F-measure for each of the three labels is then combined to produce a single aggregate F-measure for all of the data, this quantity then becomes equivalent to the accuracy (due to the lack of a "no prediction" option). As such, the results presented here will show the error in the predictions for each label as well as the overall accuracy of the predictions.

Manually Labeled Game Dialogue

For our set of manually labeled (juried) game data, we used a collection of 225 dialogue lines collected from the first two chapters of the game *Neverwinter Nights* (NWN 2009). These lines were given to seven individuals with both game-playing experience and content creation experience, who labeled the lines as being of low, medium, or high sophistication. Each individual performed their labeling independently. We then took the set of lines in which five or more individuals agreed on the label and used this as our data set. The size of our labeled set was 181 lines with 50 low, 76 medium, and 55 high sophistication lines.

The best results were obtained through the use of the binary valued bag-of-words classifier with the average words per sentence feature included. With this classifier we were able to obtain a mean accuracy of 63.37% with a standard deviation of 0.75%. The use of TF-IDF in place of a binary value caused overall results to drop by approximately 14%. Figure 2 shows the results. The bars represent the percentage of the lines with the given actual label to be predicted as low, medium, or high sophistication (as ordered left to right in each of the groups). We can see that the majority of mistakes made by the classifier are by a single category, whereas the number of misses by two categories is quite low.

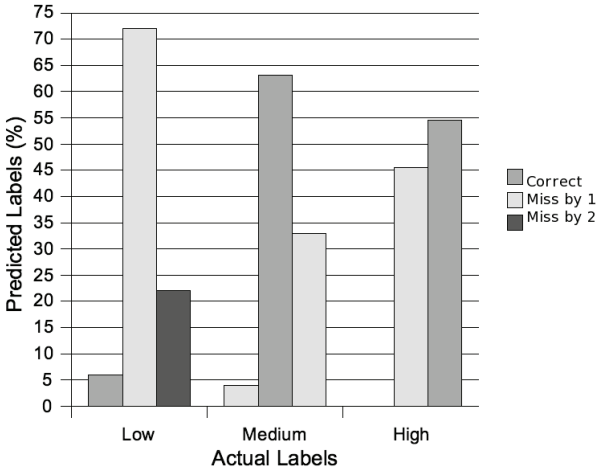

Figure 2. Sophistication classifier accuracy of juried game dialogue lines.

Semi-automated Labeling of Movie Dialogue

For our movie dialogue lines, we chose to use *The Lord of the Rings: The Fellowship of the Ring* (LOTR 2002) and *The Princess Bride* (TPB 2001). Character labels were selected by viewing the films and judging the perceived level of sophistication in each character's dialogue. Characters who lacked a consistent manner of speaking were excluded. The character labels were then propagated

to the lines spoken by each character, providing us with a labeled collection of 1,410 dialogue lines. The evaluation set used was the same set of 181 juried lines from *Neverwinter Nights* (NWN 2009) described previously.

The best results were obtained through the use of the TF-IDF valued bag-of-words classifier with the average words per sentence feature included. These features provided an overall accuracy of 45%. Use of the binary-valued classifier caused a drop of 11%. Results for this classifier are shown in Figure 3 in the same format as those from the previous experiment. Here we can see that while the results for medium and high sophistication lines are similar to those obtained when training on juried data, the performance of the classifier on low sophistication lines is quite poor. An inspection of some low sophistication dialogue lines, from both games and movies, suggests that low sophistication movie dialogue is of higher sophistication than the low sophistication dialogue in games. Many of the low sophistication game dialogue lines are rather comically low, whereas the film lines are not so extreme. The film lines are more self-consistent than these results indicate. The 5-fold cross validation on the film data produced accuracies of 41% for low, 59% for medium and 45% for high. However, these values remain too low and further study is required for this method to be of practical use.

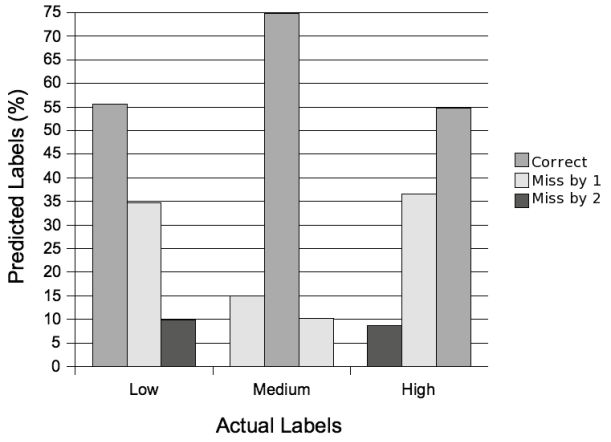

Figure 3. Sophistication classifier accuracy of film dialogue lines.

Additional Results

In addition to the results presented, we also investigated some other features and techniques. An average-syllables-per-word feature was tried; the feature was much like the average-words-per-sentence feature described previously. There was little to no increase in accuracy using the feature, however, and it required a far greater number of bins than did the average words per sentence feature.

Some attempts were made at culling the movie data to remove inaccurate labels. In the first case, 5-fold cross-validation was performed on the movie data and incorrectly predicted lines were removed. The remaining lines were used to train a new, more self-consistent, classifier. In the second case, a similar approach was taken, except that when excluding a fold, the remaining four folds were combined into four classifiers (excluding one fold in each) and a voting scheme was used to decide whether to remove a line. Neither method provided a large enough increase in results to justify being used.

We also tried the standard practice of excluding a list of stop words from the word features used. Several stop word lists of varying size were used, and while some did provide minimal gains to the accuracy of the movie-dialogue-based classifier, all of them caused large decreases to the accuracy of the game-dialogue-based classifier. Again, the minimal gains were not large enough to warrant the inclusion of the technique.

Finally, the use of word pairs as features (as well as more general n-grams) was also attempted. This approach did show some promise when used with the game dialogue. However it drastically increased the size of the feature space. Due to this, it was not possible to use the features in the movie-dialogue-based classifier. It is likely that the witnessed increase was due to the word pairs implicitly including some grammatical information. However, due to the inefficiency of the feature, it was not included in the final classifiers.

Conclusion

We defined an abstraction based on the intentional dialogue line that allows game designers to manage multiple variations of a game dialogue. This approach allows the designer to map an actual dialogue line at game time, based on desired characteristics of the NPC speaker, PC and other game context. The designer can use a different mapping for different NPCs or groups of NPCs or at different times during the game. We also created a classifier to label the sophistication of a dialogue line to allow designers to more rapidly populate intentional dialogue lines by making use of existing content. While the juried game dialogue classifier certainly has room for improvement, the quality of its predictions is high enough for it to be of practical use. With nearly two-thirds of the dialogue lines properly sorted within the intentional dialogue line, the time required to re-sort the lines that were erroneously labeled will be less than the time it would take to sort all the lines from scratch. Using film dialogue to train a classifier was problematic. However, its relatively low cost indicates that more work should be done before dismissing it.

References

Hecker, C. 2008. Structure vs Style. Invited talk at *2008 Game Developers Conference*. http://chrishecker.com/Structure_vs_Style (accessed 2009).

Rabin, S., Pfeifer, B., Hecker, C., Reynolds, S., Isla, D. 2009. The Photoshop of AI: Debating the Structure vs Style Decomposition of Game AI at *2009 Game Developers Conference – AI Summit*. https://www.cmpevents.com/GD09/a.asp?option=C&V=11&SessID=8279 (accessed 2009).

Hocking, C. 2008. I-fi: Immersive Fidelity in Game Design. Invited talk at *Game Developers Conference*.

NWN. 2009. http://nwn.bioware.com

Oblivion. 2009. http://www.elderscrolls.com

Siegel, J. and Szafron, D. 2009. Dialogue Patterns - A Visual Language For Dynamic Dialogue. *Journal of Visual Languages and Computing*. Forthcoming.

Kacmarcik, G.. 2006. Using Natural Language to Manage NPC Dialog. In *Artificial Intelligence and Interactive Digital Entertainment*.

Mateas, M. and Stern, A. 2003. Facade: An Experiment in Building a Fully-Realized Interactive Drama. In *Game Developers Conference*.

Mass Effect. 2009. http://masseffect.bioware.com

Lin, D. and Pantel, P. 2001. Discovery of Inference Rules for Question Answering. *Natural Language Engineering* 7(4):343-360.

Annett, M. and Kondrak, G. 2008. A Comparison of Sentiment Analysis Techniques: Polarizing Movie Blogs. *Proceedings of the Twenty-First Canadian Conference on Artificial Intelligence,* 25-35.

Chang, C. and Lin, C. 2001. LIBSVM : a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Salton, G. 1991. Developments in automatic text retrieval. *Science* 253:974-979

LOTR. 2002. The Lord of the Rings: The Fellowship of the Ring. Dir. Peter Jackson. [DVD] New Line Home Entertainment, Inc.

TPB. 2001. The Princess Bride. Dir. Rob Reiner. [DVD] MGM Home Entertainment.