Computational Humour: Utilizing Cross-Reference Ambiguity for Conversational Jokes

Hans Wim Tinholt and Anton Nijholt

University of Twente
{tinholt,anijholt}@cs.utwente.nl

Abstract. This paper presents a computer implementation that utilizes cross-reference ambiguity in utterances for simple conversational jokes. The approach is based on the SSTH. Using a simple script representation, it is shown that cross-reference ambiguities always satisfy the SSTH requirement for script overlap. To determine whether script opposition is present, we introduce a method that compares the concepts involved based on their semantic properties. When a given cross-reference ambiguity results in script opposition it is possible to generate a punchline based on this ambiguity. As a result of the low performance of the anaphora resolution algorithm and the data sparseness in ConceptNet the application performs moderately, but it does provide future prospects in generating conversational humour.

1 Introduction

Many jokes are based on ambiguity. A joke often originates from a set-up, which can be interpreted in two ways. There is an obvious interpretation, but a less obvious interpretation exists as well. The realization that this hidden meaning exists is often perceived as humorous. Natural language provides a lot of possibilities for constructing ambiguous sentences. One type of ambiguity in natural language is cross-reference ambiguity, also referred to as anaphora ambiguity. Some examples of ambiguous anaphoric references are 'they' in example 1 and 'she' in example 2.

Example 1. The cops arrested *the demonstrators* because *they* were violent.

Example 2. Mary asked *Susan* a question, and *she* gave the answer.

Sometimes a joke can be made by emphasizing the hidden interpretation of an ambiguity. In example 1 for instance, one could make a joke by asking whether the cops were violent. Whether someone considers such a joke based on cross-reference ambiguity to be humorous depends on many factors, but jokes based on some ambiguous sentences are more likely to be perceived as humorous than jokes based on others. A joke based on example 1 for instance, is likely to be

F. Masulli, S. Mitra, and G. Pasi (Eds.): WILF 2007, LNAI 4578, pp. 477–483, 2007.

considered humorous, but asking "Did Mary give the answer?" in example 2 is not particularly funny. This paper presents a system that distinguishes non-humorous cross-reference ambiguities in a conversation from ambiguities that are likely to be humorous. Whenever a humorous ambiguity is found the system generates a simple conversational joke.

2 Distinguishing Between Humorous and Non-humorous Misunderstandings

Our method to distinguish humorous anaphora misunderstandings from non-humorous ones is based on the SSTH[1]. The main hypothesis of this theory is:

> A text can be characterized as a single-joke-carrying text if both of the [following] conditions are satisfied:
> i) The text is compatible, fully or in part, with two different scripts
> ii) The two scripts with which the text is compatible are opposite (...).
> [1, p. 99]

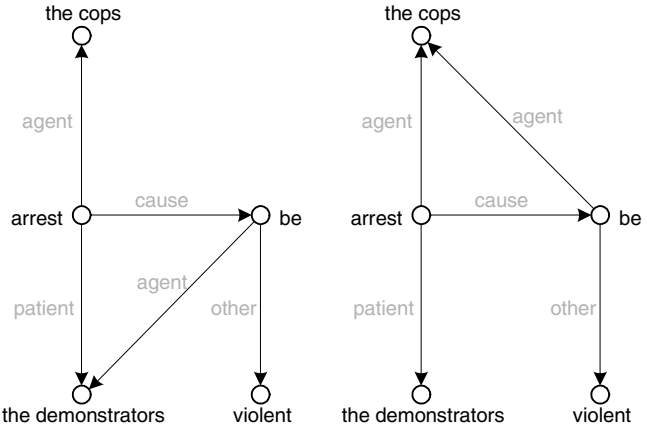
To implement this theory a script representation is needed. Using this script representation it should be possible to generate two scripts that are compatible with a certain anaphora ambiguity. When both script overlap (condition i) and script opposition (condition ii) are present the ambiguity is humorous according to this theory and a joke can be generated. The script representation and the methods to determine whether overlap and opposition are present are discussed below.

3 Script Representation

The script representation used by us is a graph based representation, which is basically a simplified version of the representations introduced by Attardo [2]. It does not require a comprehensive interpretation of the text or the competence to understand the intentions of the persons involved. However, this representation suffices for the analysis of most anaphora jokes and it can be automatically generated based on the output of a semantic role labeller.

Figure 1 shows the two scripts that can be generated based on example 1. The script representation on the left represents the interpretation that 'they' refers to 'the demonstrators', the script representation on the right represents the interpretation that 'they' refers to 'the cops'. This type of script representations will be used to check whether both overlap and opposition exist between the two scripts involved in an anaphora ambiguity.

[1] Note that the SSTH has been extended to a more general theory: the GTVH. This theory allows jokes to be analyzed at different levels and it can account for joke similarity. However, for the simple computational analysis of jokes used in our system these additions do not provide much added value.



(a) Script 1 (b) Script 2

Fig. 1. Representations of the two interpretations of the anaphora ambiguity

4 Script Overlap

In the semantic network representation of scripts, two semantic network graphs are considered to be overlapping if there exists a subgraph G, such that $G \subset S_1 \& S_2$ or in other words, if they share at least one edge and two adjacent vertices [2, p. 34–35]. In the script representation of example 1 it is clear that both graphs are overlapping. Both scripts are identical, except for one edge.

The fact that the scripts of both interpretations of the anaphor are overlapping in the given example is not a coincidence. This is always the case with anaphora misunderstandings because (1) the vertex with the verb to which the anaphor is connected (in the example the vertex 'be') is always present in both graphs and (2) this verb is always preceded by a piece of text that mentions both possible antecedents (in the example 'the cops' and 'the demonstrators'). The script representation of this piece of text is exactly the same for both interpretations of the anaphora ambiguity, so it will contain at least one additional overlapping edge and an extra overlapping node.

5 Script Opposition

Now that we have established that script overlap is always present in anaphora ambiguities, the only requirement left for an ambiguity to be humorous is script opposition. It was already stated that in the case of an anaphora ambiguity, the graph representations of both possible interpretations are identical, except for one edge. Therefore this edge must play a key role in the opposition. In the example the edge that differs is the edge between 'the demonstrators' and 'be' in figure 1a and the edge between 'the cops' and 'be' in figure 1b.

'The demonstrators' and 'the cops' are both noun phrases (NPs). Intuitively one might say that these NPs are more or less opposed, but NPs do not have distinct antonyms. The approach we use to compare NPs is inspired by the statement of Fromkin et al. [3, p. 258] that two antonyms have all characteristics in common, except for one, which is present in one word, but not in the other. We use this observation to find antonyms of NPs based on the properties of both NPs. 'The cops' for example are 'human', 'alive', 'grouped', 'orderly' etc. This means that an opposite should have all properties in common, except for one, which should be opposed. This opposed property will be called the *comparison axis*. If we use the property 'alive' as comparison axis for example, then a group of dead cops can be considered to be opposed to 'the cops'.

In our computer implementation we use ConceptNet [4] to retrieve all properties of the two NPs that are involved in the scripts. If they have a sufficient number of properties in common[2] and the non-overlapping properties contain at least one antonymous pair, then the NPs are considered to be opposed. In the example 'the cops' and 'the demonstrators' actually have a lot of properties in common. There are only a few properties that differ (good, brave, orderly, rowdy and grouped) and these properties do contain a pair of antonyms: According to WordNet [5] the property 'rowdy' of 'the demonstrators' is an antonym of the property 'orderly' of 'the cops'. Based on this observation it can be concluded that both NPs are more or less opposed.

Technically speaking this method only proves that the two NPs in the ambiguous sentence are opposed. It does not prove that the complete scripts are opposed. However, the NPs are essential for the difference between the two scripts. Therefore it seems to be safe to assume that if the vertices of the NPs are opposed, the scripts are opposed as well.

6 Implementation

The presented method to distinguish humorous anaphora ambiguities from non-humorous ones was implemented in a Java application. Figure 2 shows an overview of the four modules that form this system.

The *conversation agent* is responsible for receiving input from the user and keeping the conversation with the user going. It forwards all input from the user to the rest of the system. If the system indicates that an anaphora joke can be made, then the conversation agent will make this joke. Otherwise, the conversation agent uses an AIML based chatbot [6] to formulate a response to the user. Figure 3 shows an example of a conversation with the conversation agent. Based on the first two utterances of the user the system cannot make an anaphora joke, so it uses the AIML chatbot to formulate reactions. The last utterance does contain a humorous anaphora ambiguity, so this time the conversation agent returns the joke "The cops were violent?".

[2] Note that the requirement that both concepts must have *all* properties in common except for one is relaxed a little. A concept that has *most* properties in common still turned out to be reasonable opposite.

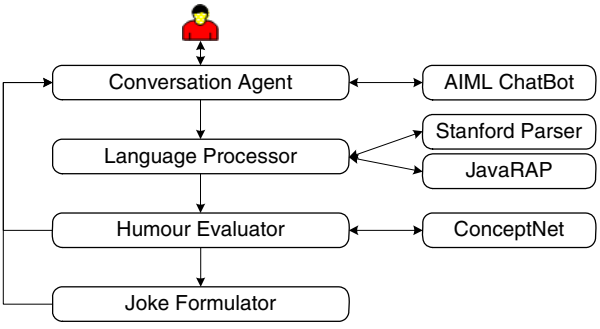


Fig. 2. System architecture

```
User: Hi.
System: What can I call you?
User: My name is John.
System: John, nice to meet you! Are you a man or a woman?
User: I am a man. Did you know that the cops arrested the demonstrators
      because they were violent?
System: The cops were violent? Or the demonstrators? :)
```

Fig. 3. Example of a conversation with the system

The analysis whether a user utterance contains humorous anaphora ambiguities is done by the *language processor* and the *humour evaluator*. The language processor uses JavaRAP [7], a Java implementation of the resolution of anaphora procedure of Lappin and Leass [8]. JavaRAP locates every pronominal anaphor in the text and returns its *real antecedent* only. For the purposes of our system it was adjusted so it returns all possible antecedents from the current and previous sentences, which are not excluded based on eliminating factors like gender and number agreement. If at least two possible antecedents are found, the anaphoric expression is ambiguous and this information is forwarded to the humour evaluator to check whether this ambiguity is humorous.

The *humour evaluator* implements the check for script opposition described earlier in this paper. Roughly this comes down to comparing the *real antecedent* of the anaphor to the possible *other candidate(s)*. The properties of both NPs are retrieved from ConceptNet. Both NPs are compared based on their properties. If they have a sufficient number of properties in common and the non-overlapping properties contain at least one pair of antonymous properties, then the scripts representing both interpretations of the text are considered to be opposite, and a joke can be formulated.

In case of a humorous anaphora ambiguity, the *joke formulator* formulates a joke. The joke is a simple clarification request that indicates that the anaphoric reference was (deliberately) misunderstood. In case of the example with the cops and the demonstrators this results in the clarification request "The cops were violent?".

7 Evaluation

The system was evaluated by having it analyze a chatterbot transcript and a simple story text. In total these texts contained 253 cross-reference ambiguities. These ambiguities were first analyzed by hand to locate all possibilities for anaphora jokes. After this manual analysis the texts were input to the system for automatic analysis. The results of the automatic analysis were then compared to the manually obtained results to evaluate the performance of the system.

In this evaluation the system achieved a precision of 15% and a recall of 15%. This shows that the system is still quite susceptible to errors. The low precision can mainly be attributed to errors in the anaphora resolution. JavaRAP has an accuracy of less than 60% when it needs to rely on imperfect parser output. This caused the system to make a significant amount of non-humorous jokes. The low recall can be attributed to three factors. Errors in the anaphora resolution caused 60% of these errors, 13% was caused by data sparseness of ConceptNet and 27% can be ascribed to limitations of our humour evaluation method. This moderate performance makes it unfeasible to use the system in existing chat applications. However, the system was able to make a number of jokes and the humour evaluation method does provide future prospects in generating conversational humour.

8 Conclusion

This paper described a first attempt at automatically generating jokes based on cross-reference ambiguity. Cross-reference ambiguity occurs frequently in texts and conversations, but it turned out that humorous cross-reference ambiguities are very rare. In fact, it was not feasible to chat with the system until it made one or several jokes. Therefore the system was evaluated by having it analyze several chat transcripts and a simple story text.

The evaluation showed that the system is susceptible to errors. Errors in the anaphora resolution and the data sparseness of ConceptNet turned out to be significant limitations. Nevertheless, the system was able to make a number of jokes from the chat transcripts and simple texts. Since many of the errors cannot be prevented without improving the anaphora resolution and the amount of information in ConceptNet, it is unfeasible to further develop the implemented system and use it in existing chat systems. However, the methods that were introduced may very well prove to be useful in future applications and the idea to utilize ambiguities in a text to construct jokes is useful as well.

References

1. Raskin, V.: Semantic Mechanisms of Humor. Dordrecht–Boston–Lancaster: D. Reidel Publishing Company (1985)
2. Attardo, S., Hempelmann, C.F., di Maio, S.: Script oppositions and logical mechanisms: Modeling incongruities and their resolutions. International Journal of Humor Research 15(1), 3–46 (2002)

3. Fromkin, V., Rodman, R., Neijt, A.: Universele taalkunde. 3e druk edn. Foris Publications, Dordrecht (1986)
4. Liu, H., Singh, P.: Commonsense reasoning in and over natural language. In: Negoita, M.G., Howlett, R.J., Jain, L.C. (eds.) KES 2004. LNCS (LNAI), vol. 3213, Springer, Heidelberg (2004)
5. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
6. Wallace, D.R., Tomabechi, D.H., Aimless, D.D.: Chatterbots go native: Considerations for an eco-system fostering the development of artificial life forms in a human world (January 2003)
7. Qiu, L., Kan, M.Y., Chua, T.S.: A public reference implementation of the rap anaphora resolution algorithm. In: Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004) vol. 1, pp. 291–294 (2004)
8. Lappin, S., Leass, H.J.: An algorithm for pronominal anaphora resolution. Computational Linguistics 20(4), 535–562 (December 1994)