AAAI Technical Report FS-12-02
Artificial Intelligence of Humor

Constructions for Joke Recognition

Lauren M. Stuart
CERIAS, Purdue University, West Lafayette, IN, USA
lstuart@purdue.edu

Abstract

The notion of constructions, from Construction Grammar, is borrowed for use in joke recognition by a knowledge-based computational text analysis system. The joke recognizer is a proposed addition to an existing text analysis framework, Ontological Semantic Technology. Joke recognition is based upon calculation that the candidate text exhibits qualities similar to jokes already collected and represented in a taxonomy, with other processing input. Joke templates, based on constructions, provide semantic scripts against which texts are judged. With these scripts, meta-jokes, which conform almost but not completely to a known joke script, may also be recognized.

Introduction

In the Semantic Script-based Theory of Humor (SSTH), Raskin (1985) uses the theory of scripts in explaining the operation of a joke. The SSTH posits that a text is a joke if it is (i) fully compatible with two scripts, and (ii) that the two scripts contrast. In the next development of the theory, the General Theory of Verbal Humor (GTVH), Attardo & Raskin (1991) identify further distinguishing components of a joke, which contribute to the identification and resolution of the scripts. In an attempt to provide a source for some simple, easily-identifiable scripts for recognizing jokes in text, this work borrows and generalizes the concept of grammatical constructions in a manner similar to Antonopoulou & Nikiforidu (2009), but from a computational and OST-centered perspective rather than the cognitive linguistics approach. The joke recognizer described here is a theoretical addition to the Ontological Semantic Theory of Humor (OSTH), as developed in Raskin et al. (2009), and relies on components of the existing Ontological Semantic Technology (OST), a knowledge-based text processing framework.

Criteria for a Joke and a Recognizer

In order to have a "script", we must have some collection of known information about a situation. In the case of a verbal joke, the two opposing scripts may be situational: they encompass events, expectations, and required elements of a particular situation or happening. For illustration, we follow an example analysis from Raskin (1985) of the Doctor's Wife Joke.

> (i) "Is the doctor at home?" the patient asked in his bronchial whisper. "No," the doctor's young and pretty wife whispered in reply. "Come right in."

The first script is that of a patient's visit to a doctor; this script is activated and matched by certain details in the text of the joke that are expected or normal elements of such an event. The second script is that of a lover's visit; like the first, we recover this script with various details in the joke, but we are notified of the need for a second script by the seeming incompatibility of the punch line with the first script.

For a computer to detect a joke, it must have some scripts, some notion of what constitutes opposition between two scripts, and some way of identifying and resolving the opposition to arrive at an interpretation of the joke that is compatible with that of a human reader. We discuss here some tools for working on a level closer to the "surface" (where semantic analysis is "deep") to recognize incongruities that may be components of, or constitute in themselves, verbal jokes.

Resources

First, we must outline OST in order to properly anchor the joke recognizer in the resources it uses. OST has an ontology, a graph of semantic concepts which are related to each other and have some default or optional attributes. Any object, idea, or event represented in a text under

analysis can be expressed as an instance of a concept from the ontology, modified according to other information in the text which may characterize it as an individual or as related to other concepts. In fact, analysis in the OST consists of a kind of translation into the "language" of the ontology, for storage as an item of knowledge in the Knowledge Base, another component. The contents of the Knowledge Base can be used, along with some common sense rules entered into another module, to reason. The contents can also be re-translated out into any language for which the OST has capabilities; the interpretation language with references to the ontology acts as an intermediate language in this instance. Such translation capability is added when a lexicon and grammar for the language are acquired, enabling translation from the intermediate language into the new language. Analysis uses the morphological, syntactic, and semantic information found in the lexicon and grammar to arrive at an interpretation of a text.

The lexicon is analogous to a language dictionary in the range of its entries, but contains more information than is present in usual dictionaries in order to make up for the knowledge gap between a human dictionary reader and a computer: humans who use a dictionary have some underlying knowledge (about their language, about certain classes of words, about usages and forms) that a dictionary entry merely supplements, but the OST lexicon (and the parsers or other modules active in analysis) must compensate for a computer's lack of that knowledge. An entry in the lexicon concerns a base word. A morphological field identifies the part-of-speech category of the base word form and holds rules for inflecting the word into different tenses, from singular to plural, into different categories, etc. as appropriate. A syntactic field concerns the structures that the word may be found in or that it requires in a sentence. A semantic field references the ontology to build an expression of the idea that the word references. Analysis of a text requires references to all of these fields, as fitting, in order to determine the lexical entries used in the text and the syntactic and semantic relationships present.

Constructions and Joke Templates

The general form of entries in the lexicon already falls in with a constructionist approach. According to Goldberg (2003), "[c]onstructions are stored pairings of form and function, including morphemes, words, idioms, partially lexically filled and fully general linguistic patterns." In reconciling local patterns presented by individual lexical entries' syntax and semantic fields, an analysis module uses larger rules about sentence structures and their associated meanings – also constructions.

A joke recognizing module, driven by specialized constructions unique to jokes – and making use of the general constructions – seems a logical extension to the analysis already undertaken by OST. As well, a construction – at any level, from morpheme to sentence or narrative structure – provides a script. Successively "deeper" (more semantic, less syntactic) scripts can contain smaller scripts. The meaning of a construction, its function, is necessarily not recoverable from its component parts (or else there would be no need to declare the aggregation a construction in order for its function to follow from its form) so no construction can be composed of only other constructions without some meaning that is captured at the higher level.

General sentence-level constructions are already available in OST; joke-specific constructions can be derived from a joke taxonomy. For illustration, take a relatively simple joke: the knock-knock. A potential knock-knock joke template looks like (ii):

(ii) Knock, knock.
Who's there?
X
X who?
pun on X

The template that this kind of joke uses is relatively "shallow", so many elements are rigidly-required (the opening line is classic) or require identifiable computation to resolve linkages (Hempelman (2003) and Taylor & Mazlack (2007) represent previous work in which puns are generated or recognized computationally). A recognizer making use of this construction will fit the input to the template: general analysis provides interpretations of the individual utterances in the exchange, but the joke recognizer's specialized knowledge will enable it to test the utterances against known joke templates. If the utterances appear in interaction with a human, a software agent equipped with the joke recognizer can switch out of its "bona fide" communication mode (in which it may be gathering information) and may try to answer the joke using a phonological punning module based on the above work.

Templates may not be so rigid in surface form; take, for instance, (iii):

(iii) *X1*, *X2*, and *X3* walk into a bar. *X1 does something reasonable in the bar. X2 does something else, also reasonable. X3 does something that conforms to a stereotype about X3.*

Very little of the template is pinned down in specific words' appearances or syntactic structures; instead, matching a joke to this template requires some semantic evaluation to fit the text to the script. "Deeper" templates

may allow for recognition of jokes that are not so "canned" – a longer form may employ a certain narrative strategy during the setup that is similar across several instances. The strategy may be extracted for inclusion in the taxonomy along with the "shallower" templates given here.

Recognition Using Templates

Testing input against known joke scripts is not simply a matter of one-to-one content matching and a subsequent decision that what follows must be a joke. A joke taxonomy for use by the recognizer must contain several joke classes, against which a candidate joke is audited for potential membership. The classes may be collections of templates, ranging from rigid and "shallow" as the above to more flexible and "deep" templates that attempt to capture more subtle jokes. As well, it would be prudent to include classification by features other than structure: for instance, ethnic jokes, while sharing some structural templates, form a class by virtue of their subject matter. The classing should also cover some common actor, act, and setting types that appear in jokes, so that a text using common tropes triggers analysis even if it does not conform to more rigid surface structures imposed by templates attempting to cover its class.

The strength of a guess that the candidate text is a joke depends on its membership to the classes in the taxonomy. A membership function, comparing the input to the classes, is an integral part of the recognizer; membership to a particular class may be expressed as a weight towards interpretation of the text as a joke. The taxonomy may be acquired from humor research, and the membership function weighted by the same and/or by training on a joke corpus. Thus, a text that shares some surface similarities to a joke – after all, serious texts must have been written about changing light bulbs – should ideally not score so high above the joke-candidate threshold as to trigger the joke analysis mode. The decision about whether the text has membership in the general class of "joke" is, effectively, fuzzy reasoning for a computer system as described in Klir & Folger (1988).

A joke-candidate's membership of or near 100% ought to reflect its status as a variation on a well-known joke which is completely covered in the taxonomy: the joke recognizer recognizes a joke it already knows. If it has a high score, it is still highly likely to be a joke and may be added to a joke-store and/or better represented in the taxonomy and weighting functions so that it can be recognized some other time with a savings on the processing required. In the case of a novel knock-knock joke, the pun and its input can be entered as an attested pair. The next time the joke recognizer sees the input (or if a software agent relies upon the joke-store and taxonomy in order to tell a joke itself) the new pun shows up as a possible punch line.

A low membership score may indicate either a non-joke or a new joke that is not well covered in the taxonomy. OST's semantic analyzer will attempt to make sense of the text anyway; if it scores low in terms of sense-making and logic, it may require the attention of a human attendant anyway – to correct OST's reasoning shortfall, to recognize a joke and enter it in the taxonomy or flag it for learning, to add missing information which will lead to resolution of the inconsistency (or joke), or to do something else if none of these are possible.

Additional joke-related processing can still recover a joke that is not well-represented in the taxonomy. The phonological module can employ its punning capabilities to recognize puns made in the text: an acceptable pun, by the module's calculation, can tip the analysis in favor of interpretation as a joke, since pun setups may add some amount of nonsense, unnecessary, or incongruous information that interferes with straightforward semantic processing. The semantics module may also be employed to concepts in the text that may have relationships to known actor, act, or situation concepts that have been attested in jokes and are entered in the existing taxonomy. Such relationships are encoded in the ontology; recovering them is a matter of tracing instances, present in the interpretation (the end-product of OST), back to their generalized concepts in the ontology and navigating parent, child, and part relationships to evaluate the strength of connections from present-in-text to present-in-taxonomy. In some cases, syntax-only constructions appear to imply a relationship between elements (equality, parent/child, etc.) but bona-fide semantic analysis (operating from the ontology) does not deliver a basis for that relationship. The syntactic module may be re-employed with reference to flexibility of syntactic constructions; Antonopoulou & Nikiforidu (2009) cover such instances more than adequately.

Meta-jokes, in which a text appears to be a conventional joke but frustrates even that expectation, present a challenge: they deviate from known joke templates, possibly enough to sometimes look like non-jokes to the membership function. Their inclusion in the taxonomy may be too specific for substantial gains in processing time or recognition accuracy, or too general not to wrongly flag non-jokes. However, we can still rely on the taxonomy by returning to the dual-script theory. A common joke template functions as the "first script", and a significant deviation from it that is still somehow covered by known humor strategies (punning, absurdity, etc.) constitutes the script opposition. The "known humor strategy" is then the second script, and a meta-joke has been recognized for its "meta" status.

Conclusion

The functionality of a joke recognizer thus naturally extends from the OST and from a constructionist approach. The proposed recognizer is a module within the larger OST text analysis system, synthesizing existing work in constraint-relaxing and content-matching at different levels of text analysis. The identification of common elements in certain classes of jokes, short-form and long-form, forms the basis for a taxonomy that can be used to classify jokes and, with fuzzy-like reasoning, texts which potentially carry jokes. These common elements can be thought of as semantic scripts, and are especially productive in that role when considering the recognition of meta-jokes.

References

Antonopoulou, E., and Nikiforidu, K. 2009. "Deconstructing Verbal Humor with Construction Grammar", in G. Brône and J. Vandaele (eds) *Cognitive Poetics: Goals, Gains and Gaps*, pp. 289-314. Berlin and New York: Mouton de Gruyter.

Attardo, S. and Raskin, V. 1991. "Script theory revis(it)ed: joke similarity and joke representation model." *Humor*, 4(3), pp. 293-347.

Goldberg, A. E. 2003. "Constructions: A new theoretical approach to language." *Trends in Cognitive Science*, 7(5), pp. 219-224.

Hempelmann, C. F. 2003. *Paronomasic puns: Target recoverability towards automatic generation*. Ph.D. thesis, Purdue University.

Klir, G. J. and Folger, T. A. 1988. *Fuzzy Sets, Uncertainty, and Information.* Englewood Cliffs, New Jersey: Prentice Hall.

Taylor, J. M. and Mazlack, L. J. 2007. "An Investigation into Computational Recognition of Children's Jokes." *22nd Conference on Artificial Intelligence*, Vancouver, Canada, pp. 1904-1905.

Raskin, V. 1985. *Semantic Mechanisms of Humor*. Dordrecht: D. Reidel.