ISEA2015
Proceedings of the 21st International Symposium on Electronic Art
ISSN: 2451-8611
ISBN: 978-1-910172-00-1

Comedy53: An Approach for Creating Computer-Generated Humorous Comics

Daniel Hawkins, Devin Cook, and Philippe Pasquier
School of Interactive Arts and Technology, Simon Fraser University
250 – 13450 102nd Avenue
Surrey, BC, Canada, V3T 0A3
drhawkin@sfu.ca, djc6@sfu.ca, and pasquier@sfu.ca

Abstract

This paper proposes a system's design for generating comics based on the incongruity theory of humour. We describe the field of Computational Humour, while also examining the nature of humour in the context of images and text to help provide frameworks for developing a system, comedy53, to produce computer generated comics. Based on the results of comedy53, we then propose strategies to help improve the future development of comic generation in the context of Computational Humour.

Author Keywords

Computational humour; automatic comics generation; incongruity theory; text-image relationship

Introduction

Computational Humour is a relatively new subfield of artificial intelligence given that it was only first explored in the early 1990's [1] and that AI research in modeling human behaviour and intelligence has existed for over a half century. In addition to being a new field, our understanding of humour itself is poorly realized, as no agreed upon general theory of humour exists. This is likely due to the fact that humour comes in many forms and is studied across many disciplines, spanning such fields as philosophy, psychology, linguistics, sociology, and more recently computer science [2]. One popular theory of humour that has emerged, and which computational humour has used almost exclusively [3], is the incongruity theory. The incongruity theory posits that humour is created from a conflict that exists between what is expected and what actually occurs, essentially producing an element of surprise in the viewer, which when resolved can produce a sense of mental ease and humour [4] Furthermore, the extent of the divergence (the difference in magnitude between the expected outcome and the true outcome) produces greater humorous effects during information processing [2,5].

Despite the fact that humour comes in many forms, (e.g., satirical, dark, irony, etc.) practically all computer-generated humour programs rely only on text-based humour, particularly wordplay [6]. These programs take advantage of the ambiguities found in natural languages, such as multiple interpretations and double meanings (e.g. homophones), to produce an incongruity (divergence of expectation) which is then later resolved in the punch-line [4]. While a few programs have explored the use of algorithms to produce incongruities and humour through purely text-based jokes, predominately in the form of puns, there is a lack of work exploring the use of images and text to produce humorous comics. We refer to comics in this paper, as defined by McCloud as a text image

The aim of the system design, comedy53, proposed here is two-fold: 1) develop a system that explores the relationship between image + text, to create humorous outputs based on the incongruity theory of humour and 2) use the results of comedy53 to explore future developments of computational comics, using our knowledge of existing comics and humour.

Related Work

Although a majority, if not all, of computer generated humour systems rely on a model of incongruity at some level, the concept is poorly defined in the literature. An explanation of incongruity by Nerhardt [5], comes from a study in the 1970's where he asked blind-folded participants to hold out their hands to receive objects of varying weights. By giving the weights in incremental order, the participants began to expect increased weights as the experiment continued. But when their expectation of the weight was violated, the reaction of the participant was amusement, resulting in humour. Furthermore, Nerhardt found that the extent to which participants found it funny was proportional to the divergence from their expectation.

We can begin to understand why jokes are humorous when we examine them through the incongruity theory. Take, for example, the following joke:

> *Two fish are in a tank. One fish looks over to the next fish and says: "Do you know how to drive this?"*

Here the punch-line comes at the realization (i.e. a resolution of the diverged meaning) that the two fish are in a military tank (double-meaning), and not a fish bowl tank, as one would expect. While this joke may be simple, it relies on certain contextual awareness – a known challenge for computers. As a result, most computer-generated humour involve self-contained (non-contextually integrated) use of wordplay (or puns).

One of the very first systems to generate jokes was Tom Swiftly, developed in the early 1990's [7]. A typical joke goes as follows:

"*Turn up the heat", said Tom coldly.*

Tom Swiftly produces these short quips by reconfiguring the root-word (i.e. cold, from *heat,* through antonym association) into the adverb "*coldly",* to produce contrast between the first and second utterance. JAPE [8], another program developed in the 1990's, which later became STANDUP [1] in the early 2000's, also uses self-contained puns. JAPE jokes come in the form you may expect from a children's joke book of one-liner puns. A typical joke runs as follows:

What do you get when you cross a murderer with a breakfast food? A cereal killer.

JAPE has many different programmed schemas (scripts) to generate jokes. In this example, the joke is produced by working backwards by first selecting a hyphenated two-worded answer (e.g. serial-killer) and finding similar homophones to one of these words (e.g. serial = cereal) through the linguistic database WordNet, and finding related words (i.e. hypernyms) to cereal (e.g. cereal = breakfast food) to generate the question. The question component of the joke is formed using another schema that uses basic sentence structures to pose questions. JAPE does produce the rare humorous output (subject to human evaluation) but because multiple word matches and sentence structures exist for any given schema, JAPE also produces a numerous amount of "bad" or incomprehensible jokes for every "good" one. This is because these systems do not evaluate what is funny or not through defined rules - a major challenge in computational humour [4].

Throughout the last two decades, many systems have been developed that have generated some form of verbal humour – the use of natural language, conveyed by either text or speech [1]. Only one system, AUTEUR [9], has been developed without the use of natural language or wordplay (e.g. puns, acronyms, etc.) to create humour [1]; AUTEUR instead generates visual humor by manipulating and editing videos to create humorous film sequences.

For the purpose of this research (i.e. using text and image to produce humorous outcomes), it is also important to understand the techniques used in single and three-frame comics. Short text-alone jokes (i.e. all computer generated humour hitherto) are restricted in that they are self-contained; the humorous effect of the words lies strictly within text, offering no referential link to the external world or current context. The addition of images, commonly found in comics, allows for contextual integration: images can create a situation or environment that is both recognizable to the reader and also fundamental to the joke. Thus, the approach and technique to create humour by combining text and image (i.e. multimodal) must apply a different technique than commonly seen in strictly computer generated verbal jokes. Comics, defined as "juxtaposed pictorial and other images in deliberate sequence, intended to convey information and/or produce an aesthetic response in the viewer" [10], provide a context for the viewer, and a platform for a text -image relationship. For the purposes of this research, we are interested in exploiting this text-image relationship (i.e. juxtaposition) to produce a humorous effect. Text and images can interact in three different ways to achieve humour: i) the joke lies solely in the text, while the image only provides supplementary illustration; ii) the image itself is the joke and the text is unnecessary; and iii) the joke is dependent on the interaction between the image and text, either complementing or contradicting each other [11]. Here we are interested in this latter approach: the interaction of text and image.

Humour is often produced in comics by the same two-stage process that we see in verbal humour: an incongruity is made and later resolved [12]. For instance, in the single-frame comic "Freudian slide" by Gary Larson [13] (Fig. 1), the viewer finds a man (represented as your stereotypical professor) 'sliding' into a base in baseball with the caption reading "*Freudian slide*". The viewer's belief and expectation of the phrase "Freudian slip" is challenged, and the culmination of the humour process comes when the viewer resolves that the man is Sigmund Freud who is sliding into the base. Note that the comic would not be (as) funny if the man sliding into base was an ordinary baseball player, or if the viewer had no knowledge of the phrase "*Freudian slip*".

The same two-stage approach (i.e. incongruity and resolution) can be applied to multiple-frame comics. In the four-frame comic "*Suck Note*" by Nicholas Gurewitch [14] (Fig. 2), the viewer is lead to believe that the first character is writing his own suicide note. This notion is drawn out over the first three frames, creating an expectation within the viewer. In the last frame, however, the viewer realizes that he is not writing about himself, but rather the victim.

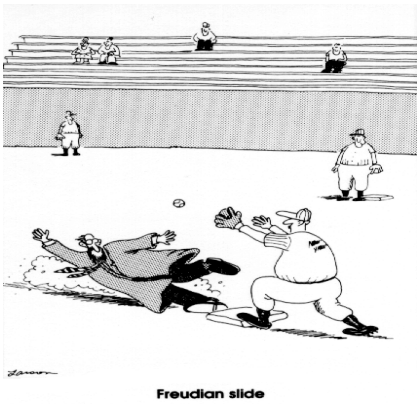

Figure 1: "Freudian slide"(The Far Side, by Gary Larson)

Figure 2: "Suck Note" (The Perry Bible Fellowship by Nicholas Gurewitch)

Based on Nerhadt's blind-fold and weight experiment to produce humorous incongruities, it is perhaps reasonable to assume that a greater humorous effect can be achieved in comics as well by raising the expectation through multiple frames [5]. Time and sequence build familiarity. In the instance of Nerhadt's study, participants didn't just receive one weighted object, they received multiple objects with increasing weight which created familiarity and expectation. In the case of "Suck Note", the multiple frames produce a narrative and a familiar context with the viewer: depression and suicide.

While these previous two examples produced a joke by creating a final resolution, comics, or jokes in general, can also be humorous by introducing an incongruity that ends with no, or a partial, resolution [11]. This structure, commonly known as *nonsense* or *surreal* humour, leaves the viewer with a sense of absurdity that is not logically congruent or resolved in the end. These are often produced in comics via bizarre juxtapositions or non-sequiturs. In Gurewitch's comic "*Bear Police*", (Fig. 3*)*, two boys are in an alley writing graffiti, which are later spotted by a police officer, in the form of a bear [14]. The police bear then chases down and mauls the two boys. The comic is concluded with the police bear peacefully enjoying a coffee and donut, presumably in a coffee shop, with no resolution. The viewer is never given a reasonable explanation of the bizarre juxtaposition of a bear dressed as a police officer who enforces the law. While the comic could have ended after three frames, the absurdity is carried out into the fourth frame, whereby any salvaged resolution the viewer may have leading up to that point is now completely removed in the final frame.

Figure 3: "Bear Police" (The Perry Bible Fellowship by Nicholas Gurewitch)

One key component of comics that should be addressed is the relationship between text and image. Many comics we encounter use images to simply illustrate the narrative of the text; these comics could exist as stand-alone text comics without using images. On the other hand, some comics use images to convey the joke and message, where text is either absent or plays a minimal role (e.g. "*Bear Police*", Fig. 3). In between these extremes, humour and meaning is created by a synergy of image and text, where both components are fundamental for constructing the joke (e.g. "*Freudian Slide*" Fig. 1; "*Suck Note*" Fig. 2). As explained above, crucial to the comic is the knowledge of "Freudian Slip" (found in text), identifying Freud (found in image), context of baseball and sliding (found both in the image of "sliding" and synonym of *slip* meaning *slide*). Thus, both the image and text are constructing and supplementing one narrative. However, a fourth category of comics exists in which there is a loose relationship between text and image, and where both elements are constructing their own narrative. An example of this approach is found in the comic strip "*A Softer World*", created by photographer Emily Horne and writer Joey Comeau [15]. *A Softer World* is produced by combining three photographs, often in sequence or of related theme and juxtaposing them with text, often in the form of a personal monologue. The process of matching text and images together varies; sometimes there is a relationship (e.g. Fig. 4), while other times there is no discernable association between text and image (e.g. Fig. 5), as if both elements are telling their own narrative. Despite having two very different approaches, both styles are effective in creating comics.

Considering the novelty and limitations of computational humour at this point in time, it is perhaps both feasible and practical to take the same approach as *A Softer World* (juxtaposition of image and text) to create computational

Figure 4: "untitled" (A Softer World by Emily Horne and Joey Comeau): Connection between text and image

Figure 5: "untitled" (A Softer World by Emily Horne and Joey Comeau): No or little relationship between text and image

comics. Though, it should be noted that in most cases the text found in *A Softer World* often is the source of humour. Thus, one of the challenges of producing computational humorous comics is crafting and selecting 'humourous' text, not just the relationship between text and image.

Computational comics have explored the use of generating image and text compilations [16], but this work is primarily focused on the graphical layout, rather than text and its relationship with images, let alone humour or sequence of events. Artist John Pound [17] however, has produced randomly generated comics which attempt to follow a loose narrative. Using PostScript, Pound published several computer generated comics (PoundArt; Fig. 6) that follow a three step process of 1) Random Production, 2) Evaluation, 3) Revision. The system Pound produces the images (i.e. drawings) and text (i.e. script) separately, and combines them together in the end process. While the finished product is aesthetically interesting, it does not provide insight into how narrative structures can be produced, nor does it offer any insight in creating humorous effects.

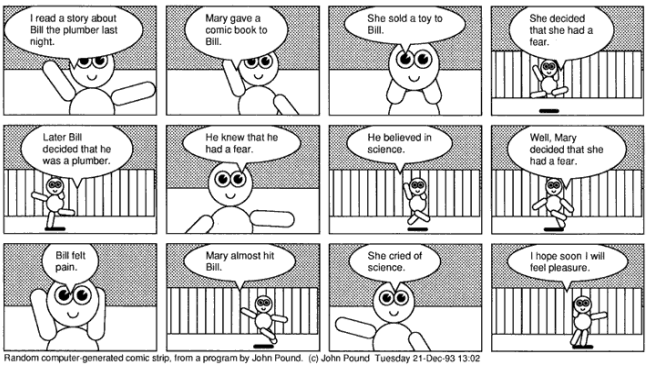

Figure 6: "This Is Not a Comic!" (Pound Art by John Pound

In her work *Grafik Dynamo* and *Why Some Dolls Are Bad*, artist Kate Armstrong produces computer generated net art by using online images from Flickr to create a live action comic strip (Fig. 7)[18,19]. The images are randomly generated based on certain search tags created by the author and mixed together with preexisting original text. The text is primarily composed of philosophical musings and maxim-like utterances, which creates a peculiar juxtaposition with the accompanying images. The comics, Armstrong believes, produces "a strange, dislocation of sense and expectation in the reader, as they are sometimes at odds with each other, sometimes perfectly in sync, and always moving and changing" [18]. A sense of expectation from the reader has the potential to be exploited with incongruity to produce a humorous effect. While the artwork does present interesting questions into the relationship between text and image, the comics themselves are not absolutely computer generated, as the text used in the final product is manually written. To our knowledge, no computer-generated systems have explored the use of image and text to produce a humorous effect.

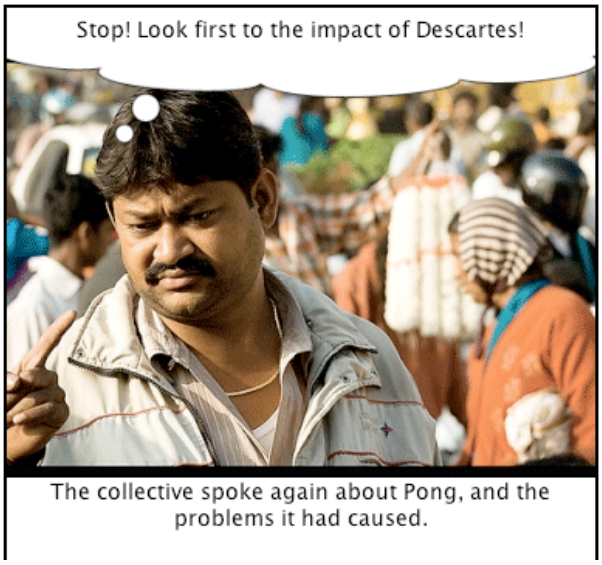

Figure 7: Why Some Dolls Are Bad by Kate Armstrong

The comedy53 program presented in this paper, while a prototype system, offers insight into basic frameworks for how we could design a system to generate humorous comics.

Comedy53 Overview

The overall approach of comedy53 is to match text with images to create humorous comics, based on the principles of incongruity in humour [5]. In doing so, we have created three approaches for generating online comics: HystLyrical, JuxtaQuotation, and FamilyTweets (available online at www.metacreation.net/comedy53).

HystLyrical

We first we examined the process of generating comics in a more narrow and simplified form by using images of classic and popular Hollywood film screenshots together with matching song lyrics. The goal of HystLyrical is to combine these two elements to produce humorous three-panel comics with a short narrative. We propose this selection of images for five reasons: 1) there exists a database of the images (screenshots) accessible online (http://film-grab.com), 2) the image size, dimension and resolution are constant across all screenshots, 3) the images contain a text label which categorize the screenshot's context (Fig. 8) 4) users are more likely to identify and reference the image (building familiarity), and 5) there is a greater likelihood for creating a short narrative using screenshots (three frames in sequential order). These last two points have the potential to produce expectation and anticipation in the viewer, which can result in an incongruity, and later a resolution (for instance, in the final third frame).

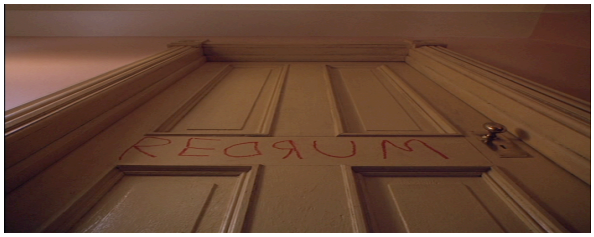

Figure 8: An image labeled "redrum" (The Shining by Stanley Kubrick)

We used a similar strategy in selecting a text database. In our approach we decided to use musical lyrics, based on three principles: 1) there exists a database an online accessible database, Lyric Find (http://www. lyricfind.com /services/lyrics-search/), 2) the lyrics have the potential to be recognizable to the reader, 3) the lyrics form a short (coherent) narrative that can build anticipation within the reader. These last two principles give the comic a greater probability of being identifiable to the viewer and can increase expectation. Venour [2] discusses this approach of familiarity to produce a greater likelihood of incongruity and registered-based humour.

Using Javascript and Ajax, the first stage in the HystLyrical system is to select a film, which is done randomly from a list of over 150 Hollywood films. Using the film *The Shining* as an example, HystLyrical then crawls the DOM (Document Object Model) to select three random images from the film. Following our example, HystLyrical selects:

http://i1.wp.com/filmgrab.files.wordpress.com/-redrum.png,
http://i1.wp.com/filmgrab.files.wordpress.com/-girls.png,
http://i1.wp.com/filmgrab.files.wordpress.com/1-axe.png

The URL images are then split and parsed to produce individual keywords. In this example, we retrieve *redrum*, *girls*, and *axe*. The keywords are then loaded into a request using LyricFind's search algorithm:

String baseURL = "http://www.lyricfind.com/services/ lyrics-search/try-our-search/?q=";

String request = baseURL +keyword1+keyword2 + keyword3;

The request then retrieves lyrics that match the keywords. In our example, LyricFind.com returns ten matches that HystLyrical inputs into a string array. Found below is one lyric result, from the song *Redrum* by *Doomsday Productions*:

"of my white Cadillac and a 9 millimeter gat Redrum evil mean muggin' body snatcha Jack the Ripper ain't got shit up on this ax hacka"

Comedy53 then searches the array (i.e. the ten different results) for the lyrics that contain the most matching image keywords. In our example, the lyrics above are selected based on the two matching words "*redrum*" and "*axe*".

Figure 9: HystLyrical output example (Generated from the film The Shining and lyrics from the song Redrum, by Doomsday Productions)

The chosen lyrics are then parsed into three verses, using the "/" as a delimiter. The original three images are then rearranged to match the sequence of keywords in the three verses. Finally, the text is superimposed onto the three images using HTML Canvas. Below is a HystLyrical output based on our example:

JuxtaQuotations

JuxtaQuotations uses the same framework as HystLyrical, but instead the goal is to combine famous quotes with contradicting images, to make a single panel comic.

Using similar methods as HystLyrical, JuxtaQuotations searches a database of quotes based on keywords (i.e. themes) from http://www.movemequotes.com/ to retrieve a quote and name of the author of the quote. JuxtaQuotations then uses Flickr's API to search and retrieve images using a keyword that is contradictory to the original theme used to find the quote. The example implemented (Fig. 10) uses the theme "Success" to find quotes related to success and then finds contradicting images using the keyword "Poverty" as an antonym to "Success". Note, currently the technique for finding contradicting themes for images is not automated, and only works for generating Success/Poverty comics as a juxtaposition.

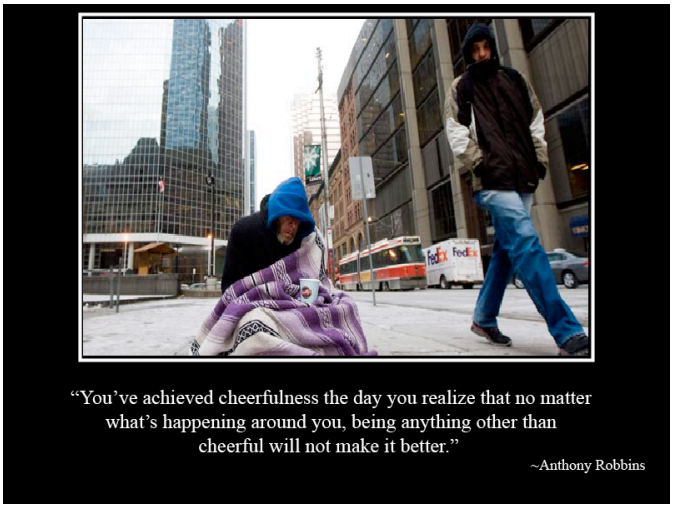

Figure 10: JuxtaQuotations output example (Success vs. Poverty)

The final output places the Flickr photo inside a white border on a black backdrop, which sits above white text from the quote and author's name. The aesthetic aims to parody popular motivational posters. The instance of combining poverty images with successful quotes also aims to poke fun at ideas of individualism. In the example below, Tony Robins, the motivational speaker, is asserting that one should always be cheerful despite their circumstances, yet the meaning is disrupted when juxtaposed with an image poverty, resulting in a deeper reflection of the text.

FamilyTweets

FamilyTweets examines the production of computational comics by using the latest image from the newspaper cartoon *Family Circus* together with the latest tweets from Twitter. In our example, we explored the use of the latest tweets from Justin Bieber's account using Twitter's API. Thus, FamilyTweets comics offer a more dynamic approach for comic generation in that both elements (text and image) are constantly updated online (image changes daily and tweets approximately ever hour). Figure 11 shows four consecutive Family Circus comics with the latest tweets from Justin Bieber on that particular day.

FamilyTweets are generated by scraping the comic located on the homepage of http://familycircus.com/, using JavaScript and Ajax. Using Twitters API and querying by username, Justin Bieber's latest tweets are pulled and placed under the comic, and over the existing caption to create a new caption (using HTML Canvas). Thus, a new comic is generated every time a new Family Circus comic is released or Justin Bieber tweets. The tweets are styled to match the existing typography of the familiar comic, thus strengthening the incongruity with the reader. The result often places Justin Bieber as the voice of the children from Family Circus, thus making his online comments seem even more juvenile.

Figure 11: FamilyTweets output example

Discussion

While the comics produced by comedy53 are by no means comical genius, they do provide us with an approach for how we can generate humorous comics. It is the hope of this research that we can extend our knowledge of computational humour, and for the first time explore the field of computer generated humorous comics.

We believe the first step in designing an improved comedy53 system lies in using different (e.g. richer) databases for text and images. For example, in HystLyrical we believe our image database (http://film-grab.com) is limited by the fact that the film screenshots labels are often ambiguous or unrelated to the image itself. This limitation of human subjectivity is difficult to circumvent in this instance, but perhaps an image database that included a consensus of labels from multiple individuals would be more appropriate. Secondly, the image labels are often only one word, therefore they do not provide a rich or descriptive context. Lastly, roughly only half of the films found on Film-Grab.com contain keyword labels, and thus the database is limited in size. Based on these three limitations, we would propose using a different database for images in the future (e.g., Google Images).

We also believe our text database was limited in three features. First, while LyricFind.com is robust in the sense that it can transform the keyword to include a wider definition of the word and tenses (e.g. the keyword *swim* can be transformed to include *swim, swam, swimming,* etc.*)* the wider group of keywords are not recognizable when compared to the original keyword image, and thus HystLyrical will not recognize a direct match. Secondly, LyricFind.com does not always return complete verses. In our example of *Redrum*, note that the lyrics retrieved

include the incomplete "*I got a double edge ax in the back*" at the beginning of the verse. To avoid this problem, we have experimented with initializing our selection from the 'second' verse (i.e. starting after the first "/"), but this can also produce further incomplete verses, as the second verse may depend on the initial verse found before the first "/" to read as a complete sentence. Lastly, roughly one-fifth of the lyrics retrieved include undesired song descriptors such as "[Chorus 1]" that tend to spoil the comics. Taking these limitations into consideration, we would propose exploring a different text database for future work. We would also recommend exploring a text database outside of lyrics. For instance text could be generated from different mediums such as poetry or even news headlines.

We also propose future work that would incorporate user evaluation and interaction. Integrating user feedback, whereby viewers classify comedy53 comics into "funny" and "unfunny" classes, could help strengthen the comedy53 model of humour. For example, if users can rate comic output, comedy53 could learn which relationships of text and images are more effective at producing a humorous effect. An example of this approach was proposed by Costa et al. [20], using a Support Vector Machine (SVM), in conjunction with online crowdsourcing, to help classify joke – a task that is particularly subjective. Here, SVM operates by determining an optimal hyperplane between two classes (funny and non-funny jokes), through a supervised classifier. We suggest a similar *training* approach in the future development of comedy53.

Another form of human interaction in comic generation was described by Tobita [21] in which the users of their program, *Comic Computing*, can interact directly with the comic images by manipulating figures and objects in the comic's frames. In Comic Computing, even mundane and boring images can be stretched or deformed to produce interesting or funny results. The role of direct human interactivity with comedy53 could produce improved results as well. One obvious approach would be to allow users to edit and manipulate the text to create their own 'improved' caption to the image. In this sense comedy53 would provide a platform that could inspire creative or funny ideas within the comic.

We propose to analyze the relationship between computer-generated text and images in comedy53's output, in a future study. In this study, we propose generating a series (e.g., 20 series) of four different comic types: 1) comedy53 generated (images with computer-generated matched text), 2) controlled images (same images as comedy53 comic, but with unmatched text), 3) controlled text (same text as comedy53 comic, but with unmatched images), 4) human generated (same images as comedy53, but manual input of keywords to match images). Participants would then rank the four different comics in each series, from most humorous to least humorous. Thus, we could then analyze which approach (1, 2, 3 or 4) tends to create the most humorous comic. We would predict that the 4th approach (human generated) would have the overall highest ranking, but this study would also explore if comedy53 can produce more humorous comics than controlled images and text (approaches 2 and 3). If this is the case, this study would help support the idea that comedy53 has provided a heuristic approach for creating computer-assisted humorous comics – one that we can improve with our suggested future work. We would also be curious to know how well comedy53 would compare next to human generated comics.

We imagine exploring other avenues of image and text. For example, mashing popular Sunday Funnies comics in original ways through simple computation. We propose a few various automated series here:

Figure 12: Garfield Minus Garfield Plus Snoopy

i) **Garfield Minus Garfield Plus Snoopy** (Fig. 12). This scenario only mashes together images (no text) by mixing together images from Garfield Minus Garfield with Snoopy from Peanuts, in the final frame.

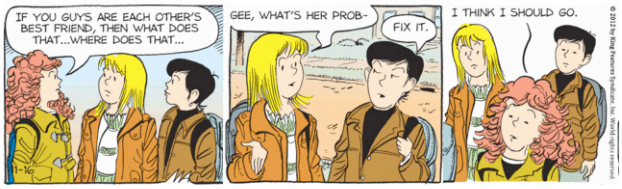

Figure 13: Sally Back and Forth

ii) **Sally Back and Forth** (Fig. 13). This manipulation simply swaps panels. Here the 2nd and 3rd panel have been swapped in Sally Forth comics to create ambiguity.

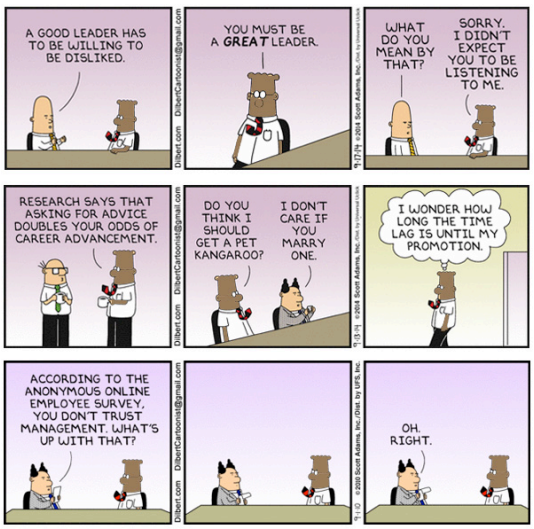

Figure 14: Black Dilbert

iv) **Black Dilbert** (Fig. 14). The comic could also be provocative and make us question racial norms. In this series of three comics Dilbert's skin colour has been darkened, changing the lens of perspective from a white to a black male.

Given the recent developments in computer generated image captions [22], it seems likely that innovative applications for generating comics will follow. For instance, using Deep Visual-Semantic Alignments in conjunction with simple word play substitution and manipulation with the produced captions, generating humorous comics and memes may come sooner than imagined.

The body of research surrounding computer-generated humour, specifically comics, is still in its infancy. Given that comics (i.e. images and text) are becoming "one of the most popular and pervasive media forms of our increasingly visual age" [23], it seems logical to investigate the field of computational comics to meet the growing demand as well as to expand and explore the medium for new possibilities. As our knowledge of computational humour progresses, so will the medium of comics.

Conclusion

This paper has described what has been done in the field of Computational Humour, while also examining the nature of humour in the context of images and text in effort to illuminate the possibilities involving computer generated comics. Here we've presented a basic framework for designing a system, comedy53, that can progress our understanding of computational comics. We suggest integrating user interactivity, subjectivity and evaluation into the comedy53 program to improve its comedic performance.

References

[1] Ritchie, Graeme. "Current directions in computational humour." *Artificial Intelligence Review* 16, no. 2 (2001): 119-135.

[2] Venour, Chris, Graeme Ritchie, and Chris Mellish. "Dimensions of incongruity in register humour." *The Pragmatics of Humour Across Discourse Domains* 210 (2011).

[3] Attardo, Salvatore. *Linguistic theories of humor*. Vol. 1. Walter de Gruyter, 1994.

[4] Stock, Oliviero, and Carlo Strapparava. "Getting serious about the development of computational humor." In *IJCAI*, vol. 3, pp. 59-64. 2003.

[5] Nerhardt, Goran. "Incongruity and funniness: Towards a new descriptive model." (1976).

[6] Ritchie, Graeme. "Can Computers Create Humor?." *AI Magazine* 30, no. 3 (2009): 71.[7] Levison, Michael, and Gregory Lessard. "A system for natural language sentence generation." *Computers and the Humanities* 26, no. 1 (1992): 43-58.

[8] Binsted, Kim, and Graeme Ritchie. *An implemented model of punning riddles*. No. cmp-lg/9406022. University of Edinburgh, Department of Artificial Intelligence, 1994.

[9] Nack, Frank, and Alan Parkes. "AUTEUR: The creation of humorous scenes using automated video editing." In *International Joint Conference on Artificial Intelligence Workshop on AI and Entertainment, Montreal, Quebec*, vol. 21. 1995.

[10] McCloud, Scott. "Understanding comics: The invisible art." *Northampton, Mass* (1993).

[11] Tsakona, Villy. "Language and image interaction in cartoons: Towards a multimodal theory of humor." *Journal of Pragmatics* 41, no. 6 (2009): 1171-1188.

[12] Paolillo, John C. "Gary Larson's Far Side: Nonsense? Nonsense!." *Humor-International Journal of Humor Research* 11, no. 3 (1998): 261-290.

[13] Larson, Gary, Steve Martin, and Jake Morrissey. *The Complete Far Side: 1980-1994*. Andrews McMeel Publishing, 2003.

[14] Gurewitch, Nicholas. "Suck Note" and " Bear Police", accessed May 4th, 2015 http://www.pbfcomics.com/

[15] Horne, Emily, and Joey Comeau. "A Softer World", accessed May 4th http://www.asofterworld.com/

[16] Kurlander, David, Tim Skelly, and David Salesin. "Comic chat." In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 225-236. ACM, 1996.

[17] Pound, John. "Pound Art (2003)", accessed May 4th,2015 http://www.poundart.com/art/randcomix/about.html

[18] Armstrong, Kate and Michael Tippett. "Grafik Dynamo. Turbulence." (2011).

[19] Armstrong, Kate. "Why Some Dolls Are Bad (2013)", accessed May 4th, 2015 http://www.katearmstrong.com/artwork/dolls.php

[20] Costa, Joana, Catarina Silva, Mário Antunes, and Bernardete Ribeiro. "Get your jokes right: ask the crowd." In *Model and Data Engineering*, pp. 178-185. Springer Berlin Heidelberg, 2011.

[21] Tobita, Hiroaki. "Comic computing: creation and communication with comic." In*Proceedings of the 29th ACM international conference on Design of Communication*, pp. 91-98. ACM, 2011.

[22] Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." *arXiv preprint arXiv:1412.2306* (2014).

[23] Varnum, Robin, and Christina T. Gibbons. "*The language of comics: word and image"*. Univ. Press of Mississippi, 2001.