A Survey on Approaches to Computational Humor Generation

Miriam Amin and **Manuel Burghardt**
Computational Humanities Group, Leipzig University
miriam.amin@studserv.uni-leipzig.de,
burghardt@informatik.uni-leipzig.de

Abstract

We provide a comprehensive overview of existing systems for the computational generation of verbal humor in the form of jokes and short humorous texts. Considering linguistic humor theories, we analyze the systematic strengths and drawbacks of the different approaches. In addition, we show how the systems have been evaluated so far and propose two evaluation criteria: humorousness and complexity. From our analysis of the field, we conclude new directions for the advancement of computational humor generation.

1 Introduction

Since the early 1990s, computer scientists and linguists have been trying to create jokes using algorithms. Despite this longstanding research and the current evolution of powerful approaches in deep learning and neural networks – which are very well able to simulate human behavior and creativity, e.g. with the composition of pop music (Chu et al., 2017), poetry (Ghazvininejad et al., 2017) or abstract paintings (Li et al., 2019) – no system has been able to imitate human humor and to generate jokes like a human until today. From a research perspective, the task at hand is called *humor generation*. It is part of the broader field of *computational humor* and the output of humor generation systems are typically, but not exclusively, jokes. Besides the creation of humor, the following aspects are relevant for the field of computational humor as a whole:

- **Humor recognition** is dealing with algorithms deciding whether a given sentence expresses a certain degree of humor (Mihalcea and Strapparava, 2005; Barbieri and Saggion, 2014; Cattle and Ma, 2018; Morales and Zhai, 2017). A related task is humor prediction (Chen and Lee, 2017).
- **Humor adaption systems** are systems that learn a person's humor preference and adapt their performance according to it (Weber et al., 2018; Winters et al., 2018).
- **Computational humor evaluation** research is concerned with the problem of how to evaluate a humor generating system (Valitutti, 2011; Braslavski et al., 2018).
- **Computational humor applications** are wide spread and range from systems that help mentally challenged children to learn and comprehend complex communication situations (Ritchie et al., 2007; Shah et al., 2016) to applications that can be used to make human computer interaction more pleasant on the human side (Binsted, 1995; Stock, 2006; Iwakura et al., 2018).
- **Computational humor datasets and corpora** are collected and published with the purpose of facilitating humor research methods using data analysis, statistical learning or deep learning. They range from data about humor norms of English words (Engelthaler and Hills, 2017), over data that conveys pairs of newspaper headlines with a humorous counterpart (Hossain et al., 2019) to multimodal data that includes not only linguistic, but also acoustic and facial expression features as reaction to humorous expressions in TED talks (Hasan et al., 2019).

Proceedings of LaTeCH-CLfL 2020, pages 29–41
Barcelona, Spain (Online), December 12, 2020.

While humor generation can be researched from both the textual and the visual perspective (Oliveira et al., 2016), this survey article is dedicated to providing an overview of only text-based systems and approaches for the computational generation of humor.
Although only a minority of the existing systems refer to humor theory explicitly, we assume that it is necessary to also consider some of the most current linguistic humor theories when discussing different computational approaches to humor generation. Among the most prominent humor theories are the following:

- Surprise disambiguation model (Ritchie, 1999; Minsky, 1984; Paulos, 1982; Shultz, 1974)
- Suls' two-stage model (Suls, 1972)
- Script-based semantic theory of humor (SSTH) (Raskin, 1984) and its extension, the General theory of verbal humor (Raskin and Attardo, 1991)

These theories agree that humor is evoked by incongruity within a text. Incongruity theories assume that a humorous text conveys two different, incompatible interpretations (also called scripts[2] or frames) that share a common part that allows us to shift from one script to another. One interpretation is usually more obvious than the other, so that the recipient begins to process only this script, until parts of the text contradict with the initial interpretation, revealing the before hidden, second interpretation. This sudden revision of understanding causes the emotions of surprise and satisfaction that is perceived as humor (Krikmann, 2006). The various theories differ in how the incongruity is created or resolved (Ritchie, 1999).

Although research on computational humor generation has been going on since the early 1990s, the field is still very heterogenous. Research is conducted by at least two different communities: humor theory/linguistics (e.g. Hempelmann, 2008; Raskin, 2012; Taylor, 2017) and natural language processing/computational linguistics (e.g. Binsted and Ritchie, 1994; Hong and Ong, 2009; Yang and Sheng, 2017). There have been attempts by the humor theory community to streamline the research activity in a series of workshops (Nijholt, 2012), however, the vast majority of existing humor systems originates from the NLP community and does not seem to pick up those more theory-driven impulses. Interestingly enough, most of the older NLP approaches also do not take into account related and previous work and for the most part seem to exist in isolation from each other (Winters et al., 2019). After all, a more recent approach aims to standardize terminology and generalize the computational template-based stream of research (Winters et al., 2019). This framework is, however, not able to capture the entirety of the existing systems and the only existing survey paper that aims at capturing activity in the field dates back to 2001 (Ritchie, 2001). With the present survey paper, we try to fill that gap by providing a comprehensive overview of the existing approaches. To be able to compare and discuss these approaches, we propose *humorousness* and *complexity* as the two main criteria for the evaluation of humor generation systems. By analyzing the strengths and weaknesses of the respective systems with regard to those criteria, we conclude directions for future research in the field.

2 Humor Generation Systems

Humor generation can be viewed as a special case of automatic text generation. Accordingly, the existing systems can be categorized as belonging to one of two major approaches to text generation: Templates and neural networks. Until Yang and Sheng's publication on an LSTM RNN for joke production in 2017, all of the previous published systems were template-based. While the early template-based systems had to be equipped with handcrafted lexicons, more recent systems have a variety of external lexical resources as source of material for generation.

It is also worth mentioning that the nature of the humorous texts produced by different systems is rather diverse. While a number of systems was focused on the generation of question-answer jokes (Raskin and Attardo, 1994; Binsted and Ritchie, 1994; Ritchie et al., 2007; Sjöbergh and Araki, 2008; Hong and Ong, 2009; Labutov and Lipson, 2012), others aimed at creating narrative jokes (Sjöbergh and Araki, 2009; Yang and Sheng, 2017; Yu et al., 2018). Furthermore, three systems generate humor

[2] A script, also known as frame, is 'an organized chunk of information about something' and 'contains information which is typical, such as well-established routines and common ways to do things and to go about activities' (Attardo, 1994). In a broader sense, every content word can be a script, as for example 'teacher' or 'workplace'. Scripts can also be more complex like 'packing for holidays'.

through lexical replacement, in acronyms (Stock and Strapparava, 2005), proverbs (Sjöbergh and Araki, 2008) or in SMS (Valitutti et al., 2016), and one system creates witty analogies (Petrović and Matthews, 2013).

In the following, we will give an overview of existing humor generation systems. Since neural networks have recently achieved state-of-the-art results on many tasks in natural language processing, we will start with the group of systems that apply them for the creation of jokes. This group, however, contains only two systems, whereas the vast majority (ten systems) belongs to the group of template-based approaches.

2.1 Neural systems

The most recent line of work on computational joke generation trains neural networks for language generation in a way that the output is a short humorous text. As was already mentioned, work in this branch started with Yang and Sheng's (2017) current affair jokes generator that is trained on a joke corpus and allows the user to specify the topics of the joke. They use an LSTM RNN as their network architecture and GloVe as vector representation for the training data input. As training data, the authors use a dataset of 7,699 jokes written by Conan O'Brien and news data in order to improve the model for the current affairs related language. The topic words to be selected by the user were extracted from the jokes with a POS-tagger, assuming that nouns typically represent topics. Additionally, the user can specify the words the joke starts with from a list (e.g. '*I don't know why but…*'). The authors attempt to implement a form of incongruity by not outputting the word with the highest probability, but instead picking words from the vocabulary with the probability they were assigned in the output layer. For example, for the topic words 'Kardashian' and 'President', the network generated the joke *'Yesterday to a new attractiveness that allows Bill Kardashian's wife to agree with the U .S . Presidents . In fairness , she said , "My spa".'.* It is evident from this example that the system is not capable of producing a humorous text.

Instead of training their model on a joke corpus, Yu et al. (2018) aim at creating humor through incongruity by training a neural network with a seq2seq model on the Wikipedia text corpus and using one polysemic word and two of its meanings as input for text generation. From that, the language model generates two sentences, each conveying one sense of the word. In the first of the two presented pun models (the authors call it joint model), an encoder-decoder network is trained to generate one sentence containing both senses. The second model (highlight model) is an improved version that adds more words associated with the initial senses to the decoded sentence. That is to make sure both conveyed senses are transferred to the audience explicitly enough. For example, one of the triples consisting of a polysemic word and two of its meanings as input for the joke models could be *'square: 1) a plane rectangle with four equal sides and four right angles, a four-sided regular polygon; 2) someone who doesn't understand what is going on.'* For this input, the highlight model outputs *'Little is known when he goes back to the square of the football club'* and the joint model generates *'There is a square of the family.*' Likewise Yang and Sheng's (2017) system, neither of the models' output is identifiable as joke.

In contrast to template-based humor systems that will be reviewed in the next section, neural systems pose no semantic or syntactic restrictions, which results in a fairly creative output. Despite the high level of creativity, both neural systems ultimately fail to generate actual humor.

2.2 Template-based systems

In linguistics, a template is understood to be a text with slots that can be filled with different variables. In order to compose a joke, a template needs to be associated with a schema, which is the structure stipulating the relationships between the variables of a template (Binsted and Ritchie, 1994). These relationships are typically chosen in a way to provoke incongruity and its resolution. In other words, the schema is the joking mechanism. The variable relations can be a lexical relationship like synonymy, meronymy or hyponymy, but also a phonetical relation, such as (quasi-)homonymy. All template-based systems need a source that provides information about relationships between words in order to fulfill the requirements of the schema. We propose a classification of template-based systems that focuses on the source of these information. It should be noted that template-based systems are typically less creative than neural systems, as they are in any case restricted to the constraints of the template which are predefined by the creator of the template.

Within the template-based strand of research we primarily distinguish between systems that use ontologies for variable selection and systems that apply quantitative measures for that end. While the first group relies on the existence of ontologies or lexicons that store relationship information that is required to fill the template explicitly, the second group calculates predefined metrics on a corpus to find the best fit for a variable arithmetically. As a third group, we have identified hybrid systems that share properties of both aforementioned groups. In the following, we present the different groups with their corresponding systems.

Ontologies for variable selection

Early generators make use of handcrafted ontologies that are tailored to the specific system. This includes the Light Bulb Joke Generator (LIBJOG, Raskin and Attardo, 1994) that produces jokes of the pattern *How many <group name> does it take to screw in a light bulb? <NumberX>. One to <activity 1> and <numberY> to <activity2>*. LIBJOG uses a lexicon of social groups and their stereotypical activities to produce jokes such as *'How many Californians does it take to change a light bulb? Twelve. One to screw it in and eleven to share in the experience.'*. Over several phases from LIBJOG-1 to LIBJOG-4, the system became incrementally more complex, allowing for more activity fields and less restricted number relations.

Binsted and Ritchie (1994) follow a similar approach by basing their punning riddle generator on a manually edited lexicon that stores lexical relationship information, such as synonymy, hyponymy and associated verbs and the phonetical feature homophony. Their system called JAPE-1 uses question-answer templates, as for example *What is <adjective> and <verb>?– <noun phrase>*. With *<noun phrase>* as user input, JAPE fills the two remaining slots based on the words' homophony, exploiting phonological ambiguity to induce humorousness. An example of such a joke is: '*What's green and bounces? A spring cabbage!'*. Figure 1 illustrates the mechanism of that joke. In total, JAPE has 15 joke templates with schemes.

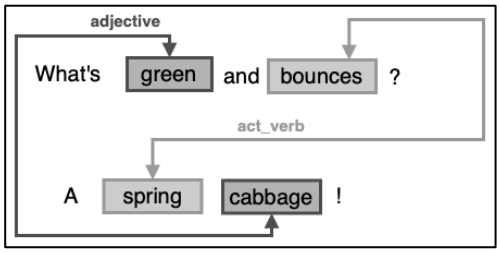

Figure 1: Illustrated example of the joke mechanism in JAPE. From: Binsted and Ritchie, 1994

Another line of systems uses freely-available ontologies and databases. Resources used by the existing systems are WordNet (Princeton University, 2010) for lexical relationships, ConceptNet (Speer et al., 2017) for semantic relationships and UniSyn (Fitt, 2002) or CMU pronouncing dictionary (Lenzo, 2007) for phonetical relationships. Furthermore, systems make use of thematic word lists for slang words or profanity. With STANDUP, Ritchie et al. (2007) present an extension of JAPE that has a user interface makes use of WordNet and UniSyn instead of the handcrafted lexicon.

HAHAcronym (Stock and Strapparava, 2005) is a system that writes out common acronyms in a humorous manner. For example, the acronym *FBI (Federal Bureau of Investigation*) becomes *Fantastic Bureau of Intimidation*. HAHAcronym makes use of WordNet, augmented with domain labels for the entries and the CMU pronunciation dictionary to take into account word rhymes and rhythms. The systems parses a given acronym to detect the syntactical form and the highest-ranking noun phrase, which remains unchanged. The other words get substituted while preserving the initial letter, the word class as well as the overall rhyme and rhythm. A substitution is chosen by exploiting semantic field opposition.

Sjöbergh and Araki's (2009) ambiguous compound generator also leverages WordNet to create jokes, this time however focusing less on inter-word relationships, but more on the provided definitions and example sentences. The system consists of two modules, creating two types of jokes, each making use of possibly ambiguous compound nouns. The first module creates jokes with the template *'I saw a <noun1> <noun2>. She (<noun1>) <WordNet example sentence>.'* For the two noun slots, the system selects compound nouns from WordNet where both parts also occur as single entries, the first as noun and the second as noun and verb. For the verb, one of the example sentences is extracted in order to complete the template to a joke such as '*I saw a fish stick. She (the fish) stuck her thumb in the crack.*' The audience initially assumes that *fish stick* is a composite, as this is the more obvious interpretation. However, this expectation is disrupted in the second sentence, when the rear part of the supposed composite turns out to be a verb. The first sentence is now reanalyzed, and the incongruity is resolved. The

second type of joke Sjöbergh an Araki's system produces, follows the template *'<WordNet definition <noun1> <noun2>>.'* For this, compounds are selected when they contain at least one word that is included in WordNet itself, and by changing one letter that word must become another word present in WordNet. This compound is combined with one entry that has the original compound in its WordNet definition. From the exemplary compound '*god of war'* this would render '*Ares: (Greek mythology) Greek god of car*'. To reduce the number of jokes, the authors implement a measure of funniness and only output jokes above a certain threshold. The measure is calculated by the relation of the frequency of the selected compound between a joke corpus and a non-humorous corpus.

Hong and Ong's (2009) system uses WordNet's relation synonymy and UniSyn's pronunciation information alongside with any semantic relationship retrieved from ConceptNet. Their system creates punning riddles similar to Binsted and Ritchie's JAPE. This time, however, the algorithm is designed to first learn question-answer joke patterns from examples and then create new jokes according to the patterns. To create joke patterns, the algorithm called T-PEG marks nouns, adjectives and verbs in every joke from a POS-tagged joke corpus as candidate variables. For every tuple of candidate variables in a joke, T-PEG determines the lexical, phonetical or semantical relationship between the two variables. Those candidate variables with at least one relationship with another candidate are the final variables in the learned template. For example, from the source pun *'Which bird can lift the heaviest weights? The crane.'*, T-PEG extracts the sentence template *'Which <X1> can <X3> the heaviest <X6>? The <Y1>.'* and the word relationships *'X1 ConceptuallyRelatedTo X6', 'X6 ConceptuallyRelatedTo X1', 'Y1 IsA X1', 'X6 CapableOfReceivingAction X3', 'Y1 CapableOf X3'* and *'Y1 UsedFor X3'*. From the 39 templates the algorithm extracted, the authors found 27 (69 %) to be usable. To generate jokes, T-PEG uses the library of patterns and WordNet, Unisyn and ConceptNet to find fitting words for the slots and a keyword input from the user as a starting point.

Quantitative measures for variable selection

On the contrary to the aforementioned systems, joke generators using quantitative measures implement hypotheses about the nature of the relations, which are not expressed in explicit variable relationships but in quantitative measures, for instance n-gram co-occurrence.

Labutov and Lipson (2012a) are the first (and only) authors to explicitly implement a linguistic humor theory, based on cycles within the knowledge graph of ConceptNet. The theoretic framework they refer to is the *script-based semantic theory of humor* (SSTH) (Raskin, 1984), according to which a text is humorous (i.e. a joke) if it satisfies the following conditions: it suits partly or completely to two different scripts and these two scripts are opposite. Instead of explicitly defining the relationships between the concepts, like in STANDUP, the authors model their assumptions of how to retrieve concepts and their relations that form a joke from ConceptNet probabilistically. Based on the premise that paths in ConceptNet can be represented as scripts, the authors hypothesize that circuits, i.e. two paths that have the same root and end node, are joke candidates. Figure 2 shows an example of a circuit. From a seed concept, the algorithm performs a depth-first search and selects candidate paths, represented as a chain of relations between concepts. Modeling SSTH's script overlap, one condition for selection of a candidate path is the maximum path overlap. In order to create a meaningful and reasonable script in human language, the maximum likelihood of transition between the relations is also considered. The transition probabilities were learned before. From the found script pairs, the ones with maximum inter-script incongruity are selected. To measure incongruity, an algorithm clusters all concepts in ConceptNet, so that related concepts form one cluster. The jokes are created with a small number of templates similar to the form *'Why does the A do B? Because the A is D.'* An example of a joke created by the system is '*Why is the computer hot in MIT? Because MIT is hell.'*

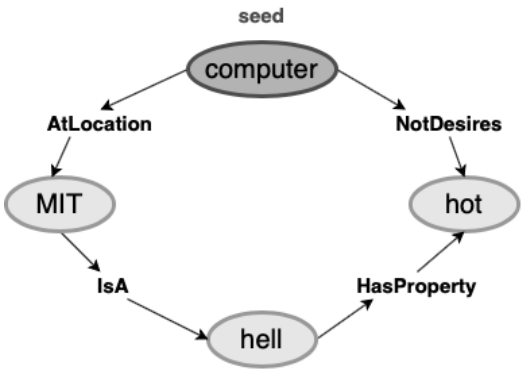

Figure 2: Example of a semantic circuit that constitutes a joke. From: Labutov and Lipson, 2012

Petrovic and Matthews' (2013) approach strikes a similar chord by encoding the semantic relationships between the variables in a template as functions and finding the words that would minimize the result. This time, however, the candidate variables are calculated based on frequencies and co-occurrences within the Google n-gram corpus, which marks a leap towards the independence of such systems from ontologies and knowledge graphs. The joke template here is *'I like my X like I like my Y, Z.'*. The authors formulate assumptions about the variables that must be met to make the completed template funny. On the contrary to Labutov and Lipson (2012a), these assumptions, however, do not seem to originate explicitly from the theory of humor, but from a self-conceived heuristic of the authors. Besides the precondition that X and Y are nouns and Z is an adjective, the model is based on the following hypotheses: (1) The adjective should be relevant to both nouns. The model calculates the co-occurrence probability from the Google 2-gram corpus (ignoring 2-grams with less than 1,000 occurrences). (2) The less common the attribute Z, the funnier the joke. This is encoded as the reciprocal frequency of Z in the corpus. (3) The more ambiguous the attribute, the funnier the joke. The authors represent this as the reciprocal of the number of senses of Z, retrieved from WordNet. (4) The more dissimilar the nouns are, the funnier the joke. To measure the similarity, the model calculates the cosine of the angle between the two nouns, represented as vectors of all attributes used to describe them in the corpus. Dissimilarity is represented as the reciprocal similarity of X and Y. Overall, Petrovic and Matthews assume that a joke is funnier, the higher the product of all factors is, i.e. the funniness probability, given a triple (X, Y, Z), normalized over all triples. For joke calculation, the authors fix one of the nouns because of limitations in computing capacity. An example of a generated jokes is *'I like my relationship like I like my source, open.'*.

Hybrid systems

A number of systems combine ontological and quantitative approaches. Sjöbergh and Araki (2008) present a system performing Japanese stand-up comedy implementing three different joking techniques. The system starts with a proverb joke, continues with a riddle joke and ends with a joke retrieved from a human generated joke database. The succeeding joke is found on the base of the words used in the previous joke and connected by the phrase '*Speaking of <keyword>, <next joke>*'. The system generates proverb jokes by selecting a proverb from a list of common Japanese proverbs and changing two words to similar sounding dirty words. Dirty words are taken from a collection of dirty words and information about similar pronunciation from a list of the phonetic distances between Japanese graphemes. The proverb '*jack of all trades, master of none'* would for instance be modified to '*jacking off all trades, masturbate none*'. For the riddle jokes, the system checks the nouns of the previous joke for similarity with a word of the dirty words list. The jokes follow the template '*A <noun> is a <noun>, but what kind of <noun> is <hint>? – <dirty word>*'. The hint is a word that describes the noun as well as the dirty word and is generated by searching '*a <dirty word> is <hint>*' in an internet search engine. The hint found is then reviewed for its co-occurrence with noun and dirty word, again using frequencies retrieved from a search engine. An example of such a joke is *'"Speaking" is "speaking", but what is a naughty kind of speaking? – Spanking the monkey*.'.

Similar to their approach, Dybala et al. (2008) develop a pun generator that creates Japanese puns as humorous responses to a user's input as part of a conversational system. In a first step, the algorithm called PUNDA generates a candidate word for a pun. The user's input is morphologically analyzed and if any word is parsed as a noun, this word becomes the base of the pun. Pun candidates are generated from the base word using one of four possible generation patterns: homophony, initial syllable addition, internal syllable addition and final syllable addition. For example, the following candidates can be generated from the base word *katana* by 1. homophony: *katana,* 2. initial syllable addition: *akatana, ikatana, ukatana, ...* 3. final syllable addition: *katanaa, katanai, katanau, ...* and 4. internal syllable addition: *kaatana, kaitana, kautana, ...* . If any of the candidates is recognized as an actual word by a morphological analyzer, the number of hits in a Japanese search engine is determined. The word with the highest hit rate is the final candidate. In a second step, a list of all sentences that contain this word is retrieved from a sentence database. The shortest sentence with exclamation mark, or if no such sentence is found, with period is selected as the final sentence. This sentence is connected to the user's input with the pattern *'Speaking of <base word>'*. An example conversation could be the following: User: *'Natsu wa atsui desu kedo, Toukyou hodo ja nai desu'. (The summer is hot, but not as hot as in Tokyo.)* with the base word: *natsu (summer)* and the pun candidate *natsukashii (emotionally attached; loved)*. – System:

Natsu to ieba, natsukashii Nose de, kyuuyuu to waiwai! (Speaking of summer, it will be fun to meet some old friends in my beloved Nose [a town near Osaka]!)

Similar to Stock and Strapparava's (2005) HAHAcronym, Valitutti et al. (2016) generate humorous texts by word substitution. They substitute a single word in a given short text message (SMS) that serves as a template. Their 2016 study is an enhanced version of a 2013 paper by the same authors (Valitutti et al., 2013). They assume different kinds of lexical constraints that determine the funniness of a substitution: (1) Taboo: The substitution is a taboo word (dirty word e.g.) or connoted with a taboo. (2) Coherence: The substitute forms a coherent compound with its neighbors. (3) Position: The substitution takes place among the last words of the text. (4) Form: The substitute is phonetically or orthographically similar to the original word. They use WordNet, the CMU pronunciation dictionary, the Google n-gram corpus after 1990, two online dictionaries of slang words and an example list of funny autocorrection mistakes from the internet, to find suitable candidates and check them for compliance with the constraints. As input for the manipulation they use an SMS Corpus. An example of a joke that leverages the constraints form and position is *'Tmr u going to school? I meet u in pool?'* (school/pool replacement).

3 Evaluation of Humor Generation Systems

In order to compare the existing systems to each other, some basic measures for the quality of humor generation systems are needed. Although a multitude of systems has been developed so far, a standardized methodology of assessing the quality of such systems is still missing. Most previous systems chose the humorousness of the output as an evaluation criterion. The majority of approaches was evaluated by means of user studies where actual humans rated the generated jokes on a Likert scale that typically ranges from 0 (not a joke) to 5 (really funny). The first authors to introduce this method were Binsted and Ritchie (1994) and many followed their example, such as Sjöberg and Araki (2008) or Hong and Ong (2009). Yang and Sheng (2017) as well as Valitutti et al. (2016) employ a four-step Likert scale. However, in most of the papers it remains unclear how the samples for evaluation were selected. Petrovic and Matthews (2013) for example disclose that they hand-picked the best jokes for the human assessment. Obviously, it matters a lot whether the average humor of a system is calculated on the basis of random jokes or on the basis of jokes that were pre-selected by the authors.

The main challenge in the design of humor evaluation measures is that humor is highly subjective and what makes one person laugh might not have the same effect on another person. Winters et al. (2018) show that the funniness judgements of jokes differ significantly among the test persons. Circumventing this issue, Valitutti (2011) proposes the HF (humorous frequency) as measure for the evaluation of computational humor generators. He defines HF as the fraction of funny items in a randomly generated set of jokes. Any item is either funny or not funny, i.e. in this context humorousness has a Boolean value rather than a score. To calculate the HF of a system, Boolean humorousness ratings have to be assigned to every item of a randomly selected set of jokes generated by the system. However, how a single joke is assessed remains highly subjective.

Binsted and Ritchie (1997) suggest to classify a computer-generated joke as successful when people cannot tell it apart from a human-generated joke, funniness left aside. Yu et al. (2018) conduct such an evaluation. The authors design what they call a Soft Turing Test to let test persons rate the humanness of the jokes generated by their system on a three-score Likert-scale (2–definitely human/0–definitely machine).

What all these approaches have in common, however, is that they do not consider the humor capability of the systems as a whole and rather evaluate individual jokes, or, as in the case of the HF, derive an overall evaluation of the system from the scores of the individual jokes. Consequently, the diversity of the produced material is completely disregarded in these evaluations. This overlooks the fact that a system that produces a row of almost similar jokes would not be very enjoyable for a human. Taking this factor into account, the humor capability of a system with a low complexity, i.e. a system that produces similar jokes with almost the same wording and the same production principle, should be rated lower than a system that produces a variety of jokes and joke mechanisms.

To our knowledge, none of the evaluation procedures yet has attempted to apply objective criteria to the decision of whether a given piece of text is humorous or not. However, from the insights of humor theory and research, criteria for what constitutes a joke can be concluded. Consequently, a useful

measure would be not if a text is humorous or how funny a joke is, but rather whether a text is identifiable as joke or not, taking into account objective criteria.

4 Comparison of Humor Generation Systems based on their Humorousness and Complexity

The considerations in the last section give rise to two main criteria for the evaluation of humor systems: humorousness and complexity. In the following, we provide a discussion of the benefits and limitations of different systems based on these criteria. We define *humorousness* as the ability of a humor system to output text that is identifiable as a joke or a text with a humorous intent. The above introduced incongruity theories can contribute objective criteria towards this end. In this sense, a text is humorous if it conveys two different concepts, whose incompatibility is resolved in an unexpected way. We define *complexity* as the ability of a humor system to generate a variety of humorous texts, concerning syntactical and lexical features, as well as the joking mechanism. The joking mechanism describes the way in which the incongruity and its resolution is evoked.
Based on these two criteria, Table 1 provides a rating of the aforementioned systems. Due to the lack of computable metrics, we do not assign scores to the single systems. Instead, we rank the systems against each other in terms of the introduced criteria on a scale from 1 to 3 where 1 is the highest rank.

Type of joke/system name	**Authors**	**Year**	**Class**	**H**	**C**
Current affair joke generation	Yang and Sheng	2017	Neural	3	1
Homographic pun generation	Yu, et al.	2018	Neural	3	1
Lightbulb jokes/LIBJOG	Raskin and Attardo	1994	Template-based	1	3
Punning riddle generators/ JAPE and STANDUP	Binsted and Ritchie	1994/2007	Template-based	1	2
HAHAcronym	Stock and Strapparava	2005	Template-based	2	2
Japanese Standup	Sjöbergh and Araki	2008	Hybrid	2	2
PUNDA	Dybala et al.	2008	Hybrid	2	2
Question-Answer jokes patterns	Hong and Ong	2009	Template-based	1	2
Ambiguous compound joke generator	Sjöbergh and Araki	2009	Template-based	2	2
Humor as circuits in ConceptNet	Labutov and Lipson	2012	Template-based	1	2
Humorous analogy generation	Petrovic and Matthews	2013	Template-based	1	3
Humorous SMS generation	Valitutti et al.	2016	Hybrid	2	2

Table 1: Ranking of humor systems with regards to humorousness (H) and complexity (C)

For the assessment of humorousness, we gave rank 3 to systems that do not fulfill the criteria for jokes at all and rank 1 to systems whose output is obviously identifiable as joke. Rank 2 was given to systems that produce text that fulfills at least some joke criteria (e.g.: Valitutti's Humorous SMS generator introduces ambiguity to the texts, but it lacks unexpectedness). However, the evaluation of the jokes could only consider the jokes published by the authors, usually 3-5 examples. As mentioned before, it can be assumed that the authors manually filtered out the best jokes for the papers. Petrović and Matthews (2013) among others, admit this and Hong and Ong (2009) explicitly mention that much of the output is not funny.

With regard to complexity, we assigned rank 3 to systems without template that restricts syntactical and lexical features, rank 2 to systems that have either a variety of different templates or template with variable slots that are large enough to allow a lexical diversity and rank 1 to systems that have only one template.

This analysis reveals the strengths and weaknesses of the two approaches to humor generation. The table clearly shows that none of the systems reaches the highest rank in humorousness as well as in complexity. The template-based approaches fill the higher ranks for humorousness by implicitly including humor theory through their constraints and metrics. The neural systems on the other hands are not capable of generating humorous text at all, scoring the lowest ranks. Nonetheless, these systems reach the highest scores in complexity through their learned language model, consequently being able to generate a large variety of text. The template-based approaches fail to reach that level of complexity.

5 Conclusion: Toward Better Humor Generation Systems by Means of Linguistic Humor Theories

Our survey shows that even after more than 25 years of work on humor generation, further research needs to be conducted in order to create a system that generates text that fulfils theory-driven criteria for humor and that is able to produce a variety of texts, regarding syntactical structures and joke mechanisms. It is somewhat surprising that almost none of the present approaches refers to a linguistic humor theory explicitly. The initially introduced humor theories and their evolution, however, make very clear that textual humor is evoked by complex semantic mechanisms. A number of requirements about the involved concepts and their semantic relationships needs to be fulfilled in order to create a text that is actually humorous rather than just puzzling. These theories enable us to explain why some texts are funny and make people laugh and others do not. Hence, we believe that considering these existing humor theories will be an important step to overcome the limitations of existing joke generators.

We regard the *surprise ambiguation model* and the SSTH as the most interesting theories. The *surprise ambiguation model*, as formalized by Ritchie (1999), defines a small set of properties and entities. Based on the three entities M_1 – the obvious interpretation of the set-up, M_2 – the hidden interpretation of the set-up and M_3 – the meaning of the punchline, the model defines the following relations and properties involved in humorous texts: *Obviousness* (M_1 is more likely than M_2 to be noticed by the reader), *conflict* (M_3 does not make sense with M_1), *compatibility* (M_3 does not make sense with M_2), *comparison* (there is some contrastive relationship between M_1 and M_2) and *inappropriateness* (M_2 is inherently odd or a taboo). Winters et al. (2019 and 2018) already used these properties to identify and validate a set of metrics that are used by machine learning algorithms to extract features from existing jokes. Also, the concept of humor postulated by the SSTH has very few requirements that could be formalized pretty easily. The SSTH even implicitly gives an instruction of how a joke can be constructed from non-humorous content. In Suls' two-stages model on the other hand, whether something is humorous or not is dependent on the presence of a cognitive rule. However, a cognitive rule can be anything and is therefore not very feasible for being modeled computationally. The GVTH focuses more on a representation of the differences of jokes rather than on the humor mechanism itself.

We see great potential for the improvement of existing approaches to computational humor generation by taking into account these theoretical approaches to humor and hope that this survey will pave the way for further, more theory-driven research in this direction.

References

Salvatore Attardo. 1994. *Linguistic Theories of Humor*. Walter de Gruyter.

Francesco Barbieri and Horacio Saggion. 2014. Automatic Detection of Irony and Humour in Twitter. In *Proceedings of the Fifth International Conference on Computational Creativity, ICCC 2014*, pages 155–162, Ljubljana, Slovenia. Association for Computational Creativity.

Kim Binsted. 1995. Using Humour to Make Natural Language Interfaces More Friendly. In *Proceedings of the AI Alife and Entertainment workshop, Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Quebec, Canada. IJCAI Organization.

Kim Binsted and Graeme Ritchie. 1994. An implemented model of punning riddles. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI '94)*, volume 1, pages 633–638, Menlo Park, CA, USA, July. Association for the Advancement of Artificial Intelligence (AAAI).

Kim Binsted and Graeme Ritchie. 1997. Computational rules for generating punning riddles. *Humor: International Journal of Humor Research*, 10(1):25–76.

Pavel Braslavski, Vladislav Blinov, Valeria Bolotova, and Katya Pertsova. 2018. How to Evaluate Humorous Response Generation, Seriously? In *Proceedings of the 2018 Conference on Human Information Interaction & Retrieval (CHIIR '18)*, pages 225–228, New York, NY, USA. Association of Computing Machinery (ACM).

Andrew Cattle and Xiaojuan Ma. 2018. Recognizing Humour using Word Associations and Humour Anchor Extraction. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING '18)*, pages 1849–1858. Association for Computational Linguistics (ACL).

Lei Chen and Chong MIn Lee. 2017. Predicting Audience's Laughter Using Convolutional Neural Network. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications (BEA@EMNLP '17)*, Copenhagen, Denmark. Association for Computational Linguistics (ACL).

Hang Chu, Raquel Urtasun, and Sanja Fidler. 2017. Song From PI: A Musically Plausible Network for Pop Music Generation. In *Workshop Track Proceedings of the 5th International Conference on Learning Representations (ICLR '17)*, Toulon, France. International Conference on Representation Learning.

Pawel Dybala, Michal Ptaszynski, Shinsuke Higuchi, Rafal Rzepka, and Kenji Araki. 2008. Humor Prevails! - Implementing a Joke Generator into a Conversational System. In Wayne Wobcke and Mengjie Zhang, editors, *AI 2008: Advances in Artificial Intelligence*, volume 5360 of *Lecture Notes in Computer Science*, pages 214–225. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tomas Engelthaler and Thomas Hills. 2017. Humor norms for 4,997 English words. *Behavior Research Methods*, 50(3):1116–1124.

Susan Fitt. 2002. Unisyn Lexicon. http://www.cstr.ed.ac.uk/projects/unisyn/

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada. Association for Computational Linguistics (ACL).

Md Kamrul Hasan, Wasifur Rahman, Amir Zadeh, Jianyuan Zhong, Md Iftekhar Tanveer, Louis-Philippe Morency, and Mohammed E. Hoque. 2019. UR-FUNNY: A Multimodal Language Dataset for Understanding Humor. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP '19)*, pages 2046–2056, Hong Kong, China. Association for Computational Linguistics (ACL).

Christian Hempelmann. 2008. Computational humor: Beyond the pun? In Victor Raskin, editor, *The Primer of Humor Research*, volume 8 of *Humor Research*, pages 333–360. Mouton de Gruyter.

Bryan Anthony Hong and Ethel Ong. 2009. Automatically Extracting Word Relationships As Templates for Pun Generation. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity (CALC '09)*, pages 24–31, Stroudsburg, PA, USA. Association for Computational Linguistics (ACL).

Nabil Hossain, John Krumm, and Michael Gamon. 2019. "President Vows to Cut <Taxes> Hair": Dataset and Analysis of Creative Text Editing for Humorous Headlines. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '19)*, pages 133–142, Minneapolis, MN, USA. Association for Computational Linguistics (ACL).

Ryosuke Iwakura, Tomohiro Yoshikawa, and Furuhashi Furuhashi. 2018. A Basic Study on Generating Back-Channel Humor Phrases for Chat Dialogue Systems. In *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*, pages 1263–1266. Publicity Committee of SCIS-ISIS.

Arvo Krikmann. 2006. Contemporary Linguistic Theories of Humour. *Folklore: Electronic Journal of Folklore*, 33:27–58.

Igor Labutov and Hod Lipson. 2012. Humor as Circuits in Semantic Networks. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 150–155, Jeju Island, Korea. Association for Computational Linguistics (ACL).

Kevin Lenzo. 2007. The CMU pronouncing dictionary. *Carnegie Melon University*.

Mao Li, Jiancheng Lv, Jian Wang, and Yongsheng Sang. 2019. An abstract painting generation method based on deep generative model. *Neural Processing Letters*, 2019:1–12.

Rada Mihalcea and Carlo Strapparava. 2005. Computational Laughing: Automatic Recognition of Humorous One-liners. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society (CogSci '05)*, pages 1513–1518, Stresa, Italy. Cognitive Science Society.

Marvin Minsky. 1984. Jokes and the Logic of the Cognitive Unconscious. In Lucia Vaina and Jaakko Hintikka, editors, *Cognitive Constraints on Communication: Representations and Processes*, Synthese Language Library, pages 175–200. Springer Netherlands, Dordrecht.

Alex Morales and ChengXiang Zhai. 2017. Identifying Humor in Reviews using Background Text Sources. In pages 492–501, Copenhagen, Denmark. Association for Computational Linguistics (ACL).

Anton Nijholt, editor. 2012. *Computational Humor 2012. Proceedings 3rd International Workshop on Computational Humor*. Universiteit Twente, Enschede.

Hugo Gonçalo Oliveira, Diogo Costa, and Alexandre Pinto. 2016. One does not simply produce funny memes! – Explorations on the Automatic Generation of Internet humor. In *Proceedings of the Seventh International Conference on Computational Creativity (ICCC '16)*, pages 238–245, Paris, France. Association for Computational Creativity.

John Allen Paulos. 1982. *Mathematics and Humor: A Study Of The Logic Of Humor*. University of Chicago Press, Chicago.

Saša Petrović and David Matthews. 2013. Unsupervised joke generation from big data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–232, Sofia, Bulgaria. Association for Computational Linguistics (ACL).

Princeton University. 2010. About WordNet. https://wordnet.princeton.edu/

Victor Raskin. 1984. *Semantic Mechanisms of Humor*. Springer Netherlands, Dordrecht.

Victor Raskin. 2012. A Little Metatheory: Thought on What aTheory of Computational Humor Should Look Like. In *Artificial Intelligence of Humor, Papers from the 2012 AAAI Fall Symposium*, Arlington, Virginia, USA. Association for the Advancement of Artificial Intelligence (AAAI).

Victor Raskin and Salvatore Attardo. 1991. Script theory revis(it)ed: joke similarity and joke representation model. *Humor - International Journal of Humor Research*, 4(3–4):293–348.

Victor Raskin and Salvatore Attardo. 1994. Non-literalness and non-bona-fide in language: An approach to formal and computational treatments of humor. *Pragmatics & Cognition*, 2(1):31–69.

Graeme Ritchie. 1999. Developing the Incongruity-Resolution Theory. In *Proceedings of the AISB Symposium on Creative Language: Stories and Humour*, pages 78–85, Edinburgh, Scotland. Society for the Study of Artificial Intelligence and Simulation of Behaviour.

Graeme Ritchie. 2001. Current Directions in Computational Humour. *Artificial Intelligence Review*, 16(2):119–135.

Graeme Ritchie, Ruli Manurung, Helen Pain, Annalu Waller, Rolf Black, and Dave O Mara. 2007. A practical application of computational humour. In *Proceedings of the 4th International Joint Workshop on Computational Creativity*, London, UK. Association for Computational Creativity.

Thomas R. Shultz. 1974. Development of the Appreciation of Riddles. *Child Development*, 45(1):100–105.

Jonas Sjöbergh and Kenji Araki. 2008. A Complete and Modestly Funny System for Generating and Performing Japanese Stand-Up Comedy. In *COLING 2008, 22nd International Conference on Computational Linguistics, Posters Proceedings*, pages 111–114, Manchester, UK. Coling 2008 Organizing Committee.

Jonas Sjöbergh and Kenji Araki. 2009. A Measure of Funniness, Applied to Finding Funny Things in WordNet. In *Proceedings of the Conference of the Pacific Association for Computational Linguistics 2009*, page pp 236-241. Pacific Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4444–4451, San Francisco, California, USA. AAAI Press.

Oliviero Stock. 2006. Password swordfish: Verbal humor in the interface. *Humor - International Journal of Humor Research*, 16(3):281–295.

Oliviero Stock and Carlo Strapparava. 2005. The Act of Creating Humorous Acronyms. *Applied Artificial Intelligence*, 19(2):137–151.

Jerry M. Suls. 1972. A Two-Stage Model for the Appreciation of Jokes and Cartoons: An Information-Processing Analysis. In *The Psychology of Humor*, pages 81–100. Elsevier.

Julia M. Taylor. 2017. Computational treatments of humor. In Salvatore Attardo, editor, *The Routledge Handbook of Language and Humor*, pages 456–471. Routledge.

Alessandro Valitutti. 2011. How many jokes are really funny? Toward a New Approach to the Evaluation of Computational Humor Generators. In Bernadette Sharp, Michael Zock, Michael Carl, and Arnt Lykke Jakobsen, editors, *Human-Machine Interaction in Translation: Proceedings of the 8th International NLPCS Workshop*, volume 41, pages 189–200, Frederiksberg, Denmark. Samfundslitteratur.

Alessandro Valitutti, Antoine Doucet, Jukka M. Toivanen, and Hannu Toivonen. 2016. Computational generation and dissection of lexical replacement humor*. *Natural Language Engineering*, 22(5):727–749.

Alessandro Valitutti, Antoine Doucet, Hannu Toivonen, and Jukka M. Toivanen. 2013. "Let Everything Turn Well in Your Wife": Generation of Adult Humor Using Lexical Constraints. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Volume 2: Short Papers*, Sofia, Bulgaria. Association for Computer Linguistics (ACL).

Klaus Weber, Hannes Ritschel, Ilhan Aslan, Florian Lingenfelser, and Elisabeth André. 2018. How to Shape the Humor of a Robot - Social Behavior Adaptation Based on Reinforcement Learning. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, pages 154–162, New York, NY, USA. Association of Computing Machinery (ACM).

Thomas Winters, Vincent Nys, and Daniel De Schreye. 2018. Automatic Joke Generation: Learning Humor from Examples. In Norbert Streitz and Shin'ichi Konomi, editors, *Distributed, Ambient and Pervasive Interactions: Technologies and Contexts*, pages 360–377. Springer International Publishing.

Thomas Winters, Vincent Nys, and Danny De Schreye. 2019. Towards a General Framework for Humor Generation from Rated Examples. In *Proceedings of the 10th International Conference on Computational Creativity*, pages 274–281, Charlotte, North Carolina, USA. Association for Computational Creativity (ACC).

Er Ren Yang and Quan Sheng. 2017. Neural Joke-Generation. In *Final Project Reports of Course CS224n*. Stanford University.

Zhiwei Yu, Jiwei Tan, and Xiaojun Wan. 2018. A Neural Approach to Pun Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 650–1660, Melbourne, Australia. Association for Computational Linguistics (ACL).