Automated Pun Generator

Pooja R. Shah
K. J. Somaiya
College of Engineering
Ghatkopar,Mumbai-77
Maharastra India.

Tarini D. Shah
K. J. Somaiya
College of Engineering
Ghatkopar,Mumbai-77
Maharastra India

Swati Mali
Department of Computer Engineering
K. J. Somaiya
College of Engineering
Ghatkopar,Mumbai-77
Maharastra India

ABSTRACT
Children suffering from Autism Spectrum Disorder (ASD) suffer from slow learning and grasping issues. Vocabulary building for such kids is a major problem. A word is learned better and the meaning of it is understood well if it is in sentence format. As English is a funny language, and a particular word has multiple meanings, it is difficult for children with ASD to grasp it.

Using artificial intelligence we aim to build the automated pun generator, pun i.e., a riddle-answer format to ease learning for children suffering from Autism Spectrum Disorder. The pun-generator will have an interactive interface, lacking in most pun generators today, thus providing a rich learning experience.

General Terms
The frequently used terms that the reader will come across this paper are stated below

(i) **Pun generator**: A system that uses punning words to generate riddles/jokes with an intention of making it humorous.
(ii) **Homophone**: Two or more words having the same pronunciation but different meanings, origins, or spelling (e.g. new and knew).
(iii) **Homonym**: Two or more words having the same spelling or pronunciation but different meanings and origins (e.g. pole and pole).
(iv) **Rhyming words**: Words that end with the same sounds. E.g. are cat, hat, bat, mat, fat and rat.
(v) **Punning words**: A form of word play that suggests two or more meanings, by exploiting multiple meanings of words, or of similar-sounding words, for an intended humorous or rhetorical effect

Keywords
Pun generator, puns, riddles, jokes, computational creativity.

1. INTRODUCTION
The power and potential of Artificial intelligence and machine learning is being understood and consequently the increasing interest of Forbes 500 companies like Google and Facebook in the field of Artificial intelligence and the acquisition of DeepMind start-up firm by Google for $600 billion has boosted the research in the field of Artificial Intelligence to make human life more efficient. Among the various applications of Artificial intelligence in Computer vision, Virtual reality and Image processing, Diagnosis of diseases, Game theory and Strategic planning, Games, Computer game bot, Natural language processing and robotics, is automated creative generation. This encompasses automated generation of poetry or proses, automated answering machines, Chat bots etc. humans by machines for creative productivity.

Inspired by these ideas, we aim to build an automated pun generator that will spontaneously generate puns (riddles in question answer format) using richness of English language and concepts like synonyms, homophones to help children build their vocabulary in a fun-loving way.

The sections discussed in this paper are as follows

Section 2: Related work, in this section ,the research papers and implementations done in the field of NLP related to computational humour are discussed.
Section 3: Proposed System, in which the system architecture and data flow is discussed
Section 4: Algorithms , in which the algorithm implemented along with an example is explained
Section 5: Implementation and Statistics, in which the different system performances are discussed.
Section 6: Limitations
Section 7: Conclusion
Section 8: Acknowledgements
Section 9: References

2. RELATED WORK
The field of natural language processing is relatively new and the WordNet project was started in 1985.Research in this field of computational humour was started in the early 2000s and has very limited resources present since.

A few systems like Jape[1] and Standup[2] were implemented in the field of computational humour.

2.1 Jape
It was developed by Binsted, using a set of symbolic rules and a large natural language lexicon to produce puns such as E.g.) what do you call a murderer with fiber? A cereal killer. But, there was no real user interface; the user would invoke the program from a simple command interface and hence was difficult to use by prospective users since commands had to be known. Another disadvantage is that the synonyms used were not very accurate and the quality of jokes generated was very poor. The pun generation mechanism of JAPE[1] was based on the type of jokes in Crack-a-joke book i.e., question answer format. Also, few popular basic puns generated same output as that in the book.

Moreover, it was difficult for kids to use it on their own since no GUI was there. The humor factor of the jokes was due to the punning nature of words rather than the subject matter. Jape used predefined fixed templates for jokes such as: “What is _____ and _____?” and several others.

The search for word substitution took hours and thereby it was not that user friendly. Out of 3 strategies that are used to generate puns , i.e. , word substitution, syllable substitution and metathesis, JAPE mechanism was based on word substitution since it is easier to find word substitutes and

replace entire words rather than substituting parts of a word.

Word substitution was done by analysis as follows:

1. Valid English word
2. Meaning of the word
3. Phonological similar substitute for the word
4. Meaning of substituted word
5. Sentence formation

After pun generation was done, there was a checking phase in which is was checked whether the word used to generate the question and the answer are accidentally identical.

2.2 Standup

The core ideas for the joke-construction mechanisms are closely based on those in the JAPE program. STANDUP[2] pun generator was intended for younger population for playing with words by building punning riddles through an interactive child friendly GUI .The targeted audience was especially children with impaired speech and Complex Communication Needs(CCN).Major difference between JAPE and STANDUP is that words in STANDUP were mapped to the pronunciation and root of words so words such as 'road' and 'rude' could be used in a single joke. STANDUP used a large, general purpose lexicon called WORDNET[3].It provided a real good interface for easy interaction with kids and also pun generation was quite quick as compared to JAPE.STANDUP could only enhance the quality of jokes but it could not eliminate the unintelligent search for words.

2.3 WordNet

The database in WordNet links English nouns, verbs, adjectives, and adverbs to sets of synonyms that are in turn linked through semantic relations that determine word definitions.

In WordNet[3], a form is represented by a string of ASCII characters, and a sense is represented by the set of (one or more) synonyms that have that sense. WordNet contains more than 118,000 different word forms and more than 90,000 different word senses, or more than 166,000 (f,s) pairs. Approximately 17% of the words in WordNet are polysemous; approximately 40% have one or more synonyms. WordNet respects the syntactic categories noun, verb, adjective, and adverb—the so-called open-class words (see Table). For example, word forms like "back,'' "right,'' or "well'' are interpreted as nouns in some linguistic contexts, as verbs in other contexts, and as adjectives or adverbs in other contexts; each is entered separately into WordNet. It is assumed that the closed-class categories of English—some 300 prepositions, pronouns, and determiners—play an important role in any parsing system; they are given no semantic explication in WordNet.

WordNet includes the following semantic relations:

- Synonymy is WordNet's basic relation, because WordNet uses sets of synonyms (synsets) to represent word senses. Synonymy (syn same, onyma name) is a symmetric relation between word forms.
- Antonymy (opposing-name) is also a symmetric semantic relation between word forms, especially important in organizing the meanings of adjectives and adverbs.
- Hyponymy (sub-name) and its inverse, hypernymy (super-name), are transitive relations between synsets. Because there is usually only one hypernym, this semantic relation organizes the meanings of nouns into a hierarchical structure.
- Meronymy (part-name) and its inverse, holonymy (whole-name), are complex semantic relations. WordNet distinguishes component parts, substantive parts, and member parts.
- Troponymy (manner-name) is for verbs what hyponymy is for nouns, although the resulting hierarchies are much shallower.
- Entailment relations between verbs are also coded in WordNet.

Table 1. Semantic Relations in WordNet

Semantic Relation	Syntactic Category	Examples
Synonymy (similar)	N, V, Aj, Av	pipe, tube rise, ascend sad, unhappy rapidly, speedily
Antonymy (opposite)	Aj, Av, (N, V)	wet, dry powerful, powerless friendly, unfriendly rapidly, slowly
Hyponymy (subordinate)	N	sugar maple, maple maple, tree tree, plant
Meronymy (part)	N	brim, hat gin, martini ship, fleet
Troponomy (manner)	V	march, walk whisper, speak
Entailment	V	drive, ride divorce, marry

Note: N = Nouns Aj = Adjectives V = Verbs Av = Adverbs

2.4 ConceptNet

ConceptNet[4], is a knowledge representation project, providing a large semantic graph that describes general human knowledge and how it is expressed in natural language. ConceptNet provides a combination of features not available in other knowledge representation projects:

- Its concepts are connected to natural language words and phrases that can also be found in free text.
- It includes not just definitions and lexical relationships, but also the common-sense associations that ordinary people make among these concepts. Its sources range in formality from dictionaries to online games.
- The concepts are not limited to a single language; they can be from any written language.
- It integrates knowledge from sources with varying levels of granularity and varying registers of formality, and makes. [4].

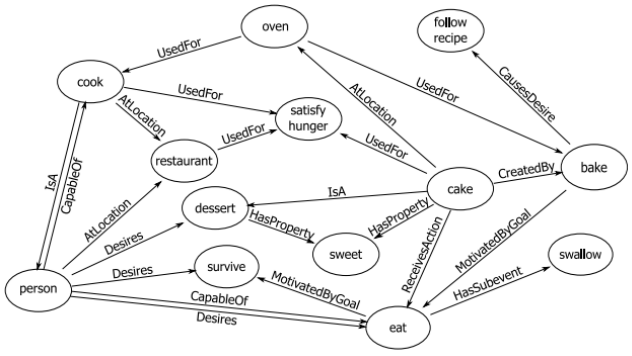

Fig 1: ConceptNet cluster of related concepts

3. PROPOSED SYSTEM

Jape does not have a GUI and STANDUP is not easily accessible. Our System is a web based approach which guarantees ease of use and coded in such a way that it does not require a lot of processing on host PC.

User interface layer is the one which directly interacts with the user, it is web based and its only functionality is prompting the user for a keyword and displaying the generated joke. The next layer in the bottom-up approach is the keyword validation layer. It returns an error if the user has malicious intent and inputs an abuse word, else it forwards the keyword to the further layers for processing. The processing of the keyword is explained in the fifth section.

The system of Automated Pun Generator is such that the User Interface is web based and coded using DJANGO for dynamic keyword input and the keyword is then passed to the python module for tokenization using NLTK. Further processing of the keyword is done based on the category and template that is selected and Homonyms or Synonyms are found using various tools cited below. The set of keywords generated after the processing are then inserted in the template and the template is sent as a string in the question answer format to Django for displaying.

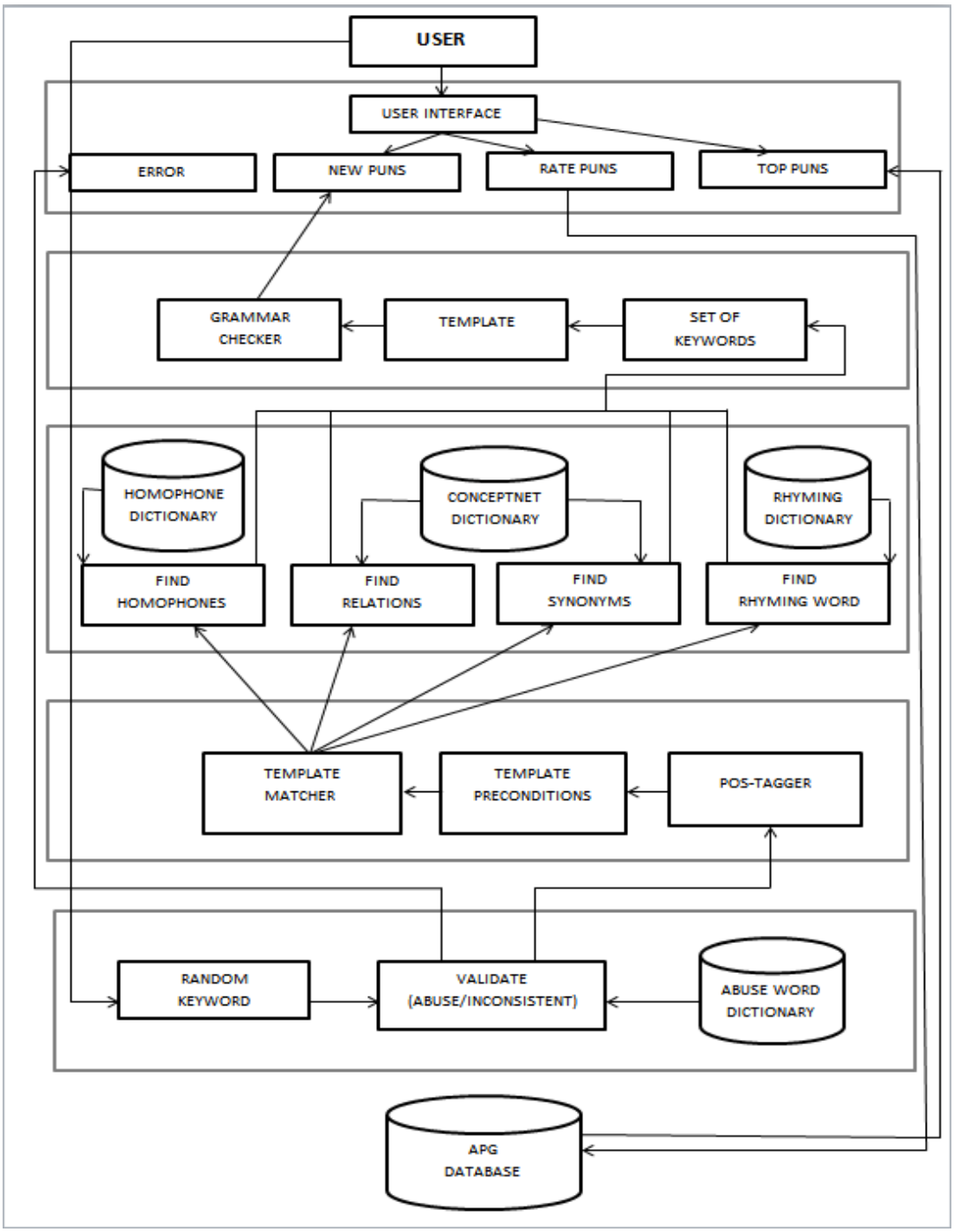

Fig 2: Multi Tier Architecture of the System

4. ALGORITHM

The algorithm can be explained with the following example:

What do you call a uninterested plank? A bored board.

The template extracted for this example will be: *What do you call a [adj A][nounA]?A [adj B][noun B].*Let the input be adj B, adj B and noun B are homophones, adj B and adj A are synonyms, noun A and noun B are synonyms

Step 1: Input: A keyword entered by the user. For the above example, input keyword is bored. Therefore, adj B is bored.

Step 2: Check whether keyword is not abusive word, generate error if found abusive.

Step 3: Find whether it is noun or adjective using POS-tag in NLTK.

Step4: Select particular template if it matches it's pre-condition.

Step 5:Accordingly find homophones, synonym , rhyming word or relationship required. Consider the example again, noun B=homophone(adj B)=board, adj ,A=synonym(adj B)=uninterested, noun, A=synonym(noun B)=plank.

Step 6: Required set of words are obtained and template is filled up.

variable set(uninterested, plank, bored, board)

Step 7: Riddle or pun generate is displayed to the user.

5. IMPLEMENTATION DETAILS AND STATISTICS

The APG system was implemented and tested on two processors AMD Fx and Intel i7 , Windows 64-bit.The performance of AMD and i7 majorly varies due to the built in GPU but since APG does not require gaming like GPU , it will not affect the system performance majorly. The factors that should be considered are clock speed and availability of cache.

The statistics were observed by individually running the system on both processors and for each joke generated, it was restarted again. The efficiency of the system can be observed by the joke referred to and the joke created by the system.

Table 2. Efficiency of the system and processing time

Sr no.	Training Template	APG generated riddle	Time in AMD (secs)	Time in i7 (secs)
1	What do you call a strange market ?A bizarre bazaar	What do you call a weird market ?A bizarre bazaar	1.87	0.97
2	What do you call an unable to bear children minor royalty? A barren baron.	What do you call a infertile nobleman ? A barren baron.	3.57	2.66
3	What do you call a superior one who bets? A better bettor.	What do you call a superior gambler? A better bettor.	2.78	1.104
4	What do you call a courageous rock? A bolder boulder.	What do you call a courageous rock? A bolder boulder.	13.45	11.0
5	What do you call a bath tour? A tub crawl.	What do you call a bathtub travel? A tub crawl.	16.23	10.59

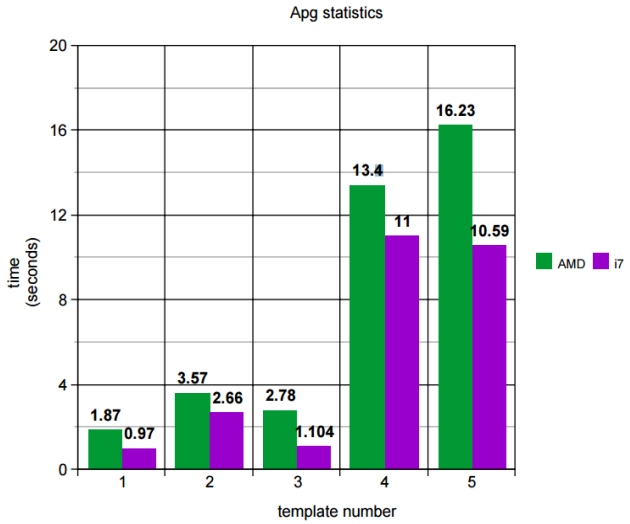

Fig 3: Graphical representation of efficiency of system

We can observe from the table and statistics graph that i7 is more efficient to run the system on. When an average of 10 templates was computed, it was found that the average processing time in seconds for AMD was 3.86 and that for i7 was 1.47seconds.

6. LIMITATIONS

The efficiency of the APG is dependent on the keyword having a homophone or rhyming word. Only then it can be generated into a joke. The second limitation is that the homophone and the corresponding synonyms should be present in the database from which it extracts. Some jokes even though generated lack the humour that we have tried to achieve because the generated riddle varies a lot from the training template. The extensive Apis , databases and the nltk toolkits require quite a lot of computational resources , which has been proved in section 6.

7. CONCLUSION

Computational humour has a long road ahead in terms of research and it is directly dependent upon the research advancing in the field of natural language processing. Quite a lot of tools and APIs are present for different modules of NLP now which enabled us to create a dynamic joke/riddle generating system.

Our system can be further developed to increase the quality and quantity of jokes by using different templates for eg. Knock Knock Jokes. A Restful API can be developed along with the UI for an android app. A feedback mechanism can be established so that user feedback can be used to improve on the quality of the joke.

8. ACKNOWLEDGEMENTS

We would like to extend our sincere thanks to Ms Swati Mali, our professor and mentor who helped us understand the essentials of algorithms and artificial intelligence and introduced us to natural language processing.

9. REFERENCES

[1] Ritchie, Graeme. "The JAPE riddle generator: technical specification." Institute for Communicating and Collaborative Systems (2003).

[2] Manurung, R., Ritchie, G., Pain, H., Waller, A., O'Mara, D. and Black, R., 2008. The construction of a pun generator for language skills development. Applied Artificial Intelligence, 22(9), pp.841-869.

[3] Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. "Introduction to WordNet: An on-line lexical database." International journal of lexicography 3, no. 4 (1990): 235-244.

[4] Speer, Robert, and Catherine Havasi. "Representing General Relational Knowledge in ConceptNet 5." In LREC, pp. 3679-3686. 2012. Applied Artificial Intelligence, 22(9), pp.841-869.

[5] Hong, Bryan Anthony, and Ethel Ong. "Automatically extracting word relationships as templates for pun generation." In Proceedings of the Workshop on Computational Approaches to Linguistic Creativity, pp. 24-31. Association for Computational Linguistics, 2009.

[6] Synonyms in English – Word list – A – F http://www.englisch-hilfen.de/en/words/synonyms.htm

[7] Homonyms and Near-Homonyms - University of Central Missouri https://www.ucmo.edu/PreBuilt/documents/HomonymsandNear.pdf

[8] Big Huge Thesaurus https://words.bighugelabs.com/api.phpnces

[9] Shah, Priyanshi R., Chintan D. Thakkar, and Swati Mali. "Computational Creativity: Automated Pun Generation." International Journal of Computer Applications 140.10 (2016).