Automatic Error Detection For Image To Text Tools

Suzanne Considine$^{1[0000-0001-8053-3250]}$, Dr. Haithem Afli[1], and Dr. Andy Way[2]

[1] Cork Institute of Technology, Rossa Avenue, Bishopstown, Cork, Ireland
suzanne.considine@mycit.ie
[2] Adapt Centre, School of Computing, Dublin City University, Dublin, Ireland

Abstract. The focus of the work discussed in this paper is to present a breakdown of the most prevalent errors that are present in automatically generated text and present the results of different methods of automatically identifying and correcting these errors. Inspired by analysing the errors present in text generated by image to text generating tools this work is also beneficial to other fields which use automatic text generating, for example bots in help centres and automatic tweeting tools. While bots and tweeting tools will not reproduce the exact errors as the image to text generation tools they will both benefit from being able to identify the errors as this will make them appear more human like in their interaction with clients and tweeters. This work centres on a dataset of captions which were automatically generated by NeuralTalk2 from data provided by NIST. This dataset was manually classified and then compared to other datasets of captions generated by other image to text tools to ensure that the errors identified as being significant were in fact significant across other datasets of automatically generated text

Keywords: Errors · Automatically generated text · Image to text.

1 Introduction

Error detection and correction has received a lot of attention over the past sixty years. The work so far has mostly focused on errors as a result of human errors, dyslexics, non-first language speakers, and errors as a result of using Optical Character Recognition (OCR) tools (e.g. of an error that can be caused by an OCR tool confusing lr with h).

Methods have been developed for correcting spelling errors or correcting confusion words ('peace of cake', 'their/there/they're', etc.). The methods developed for these errors include the edit distance-[1] for spelling errors (where once a word is identified as a misspelling, depending on the parameters of the tool used, a set of valid words one letter away from the misspelling, and another set two letters away, and so on, are presented as possible corrections). The word from the sets is then usually chosen by using a corpus broken into n-grams of 2 or 3 to see what is the most likely word from the set to fit in. Other methods include rule based systems-[2] which tended to be inflexible and needed constant upgrading to keep up with evolving language and the n-gram method-[3] which examines all n words, (usually n is 2 or 3 maximum) which appear consecutively in the text being tested and alerts the system if there in an n-gram in the testing text that does not appear in the training corpus.

From manually classifying 2,387 captions generated by image to text captioning tools we found the following four main types of errors to be prevalent: incomplete sentences, grammatically correct nonsense, redundant repetition and general grammar mistakes. Please see Table 1 for examples of these errors and their prevalence in the dataset.

Table 1. Error type and prevalence.

Error Type	Total	% of Total Classified
Correct	1775	74
Incomplete sentences (A woman is sitting on a big)	113	5
Grammatically correct nonsense (A man has a dog in his mouth)	194	8
Redundant repetition (The man is drinking tea and drinking tea)	197	8
Grammar	108	5

Apart from grammar errors, the methods discussed above, which work so well for the human and OCR errors, do not work as well or even at all, for the other three error types present in automatically generated text; grammatically correct nonsense, incomplete sentences and redundant repetition (errors where the system has repeated strings). When using the n-gram approach for error detection the parameter for n is unlikely to be set higher than 3 because once you go beyond 3 there are usually so few examples of the exact same 4 word or more string in the training corpus that it is not fit for purpose. So, the sentence, 'The woman is holding a baby in her mouth' would not be detected as incorrect using 3- grams, as on their own, each set of three concurrent words makes sense. An incomplete sentence, 'The man was running away from a very big' would also escape detection as an error with the n-gram method for similar reasons, and the error ,“ a man is walking a dog and walking a dog”, would also go undetected using this method

This study will focus on this area of the detection and correction of errors present in automatically generated text, and in particular, those related to text generated by image to-text tools.

2 Related Work

There doesn't appear to be any published effort on the detection of incomplete sentences or redundant/error repeated strings within sentences. While identifying whether sentences make sense or not is gaining increasing attention, a method still hasn't been devised that can satisfactorily detect nonsense. Nor is there much documentation showing the results of how good systems are at detecting nonsense.

ELMo(Embeddings from Language Models)-[4] and BERT (Bidirectional Encoder Representations from Transformers)-[5] are two tools which compute vectors for words (tokens). Both tools are similar in that they both work on the premise that the words that come after the token being vectorized, are just as relevant as the words that come before, when determining the context of the token. ELMo first takes into account the words before the token, then takes into account the words after the token, and thereafter, concatenates this information to compute the vector for the token. BERT however is capable of taking in the words before and after the token simultaneously and using this information to predict a masked word.

Both ELMo and BERT differ from and outperform previous vectorization applications such as 'word-to-vec' and 'GloVE', in that they can take the full sentence into account to get the vector representation for a token, thereby obtaining a more representative vector.

ELMo and BERT were not designed specifically to focus on detecting nonsense, but it has been reported-[6] that ELMo is 69.1% accurate and BERT 70.4% accurate at detecting nonsensical statements. These figures were arrived at by building a dataset of 2021 rows, each row comprising of the 5 separate parts (namely 's1', 's2', 'r1' ,'r2' ,'r3'; where firstly, 's1' and 's2' were statements of which one of 's1' or 's2' made sense while the other was nonsense, and secondly, one of 'r1', 'r2', 'r3' was the reason why either 's1' or 's2' was nonsense while the other two were illogical statements), and then feeding this labelled database into ELMo and BERT to calculate their relevant accuracies at detecting nonsense.

While their experiments did not focus on detecting nonsense, an earlier paper-[7] developed a technique they call verbaphysics that could be used to build a classifier to detect whether statements are plausible or not. Verbaphysics involved providing an application with knowledge of what would be a plausible concept and what would not, by gathering and applying specific information regarding the objects being assessed, namely, size, weight, strength (e.g. diamond is stronger than glass), rigidness (e.g. glass is more rigid than wire) and speed. During their trials, they selected 50 Levin verbs (Levin, 1993) which corresponded to 1100 unique verbs; then picked out the top 100 most used verbs from these. They then obtained approximately 9 frames per verb. Example of a frame: "The man threw the ball" and from that they gleaned information such as, the man is stronger than the ball, that the man is bigger than the ball, the ball is faster than the man. The end product is a database of objects and how these objects should relate to each other, but only as regards these five qualities. And they used this to determine whether a concept is plausible or not. For example, a man cannot throw a house as it is bigger, a man cannot chase a tree as a tree has no speed.

(Wang et al., 2018) built on the work of Forbes et al., 2017, showing again that world knowledge is necessary for semantic plausibility. They build a dataset of an equal number of plausible and implausible statements while Forbes et al. graded their 5 variables (viz. size, weight, strength, rigidness and speed) on as a value of either -1, 0 and 1, which meant the statement, “the man hugged the ant”, would be as plausible as the statement, “the man hugged the cat”, according to Forbes et al. as the information they have stored is that the man is bigger than both the an and cat, when in fact, it is not as plausible. To address this problem, Wang et al. developed a more finely grained grading system for the variables thereby allowing distinctions to be made between scenarios that wouldn’t have been under the Forbes et al. model. So that under the improvements made by Wang et al., a man hugging a cat would be deemed possible, but a man hugging an ant, not so as the ant is so much smaller than the man. Wang et al. created a dataset of 3062 statements, which for testing to ascertain whether world knowledge improves the accuracy of semantic plausibility classification. They ran (Van de Cruys, 2014) neural net classifier and found that it was only 64% successful at distinguishing between the plausible and implausible statements, but when world knowledge was injected/projected onto the neural net classifier, the accuracy rose to 76%.

3 Dataset

The dataset which we manually classified, consists of captions generated by the image captioning tool, NeuralTalk2$\hat{1}$, using data (viz. videos) provided by National Institute of Standards and Technology (NIST). NeuralTalk2 generates captions by passing the image through a Convolutional Neural Network (CNN) which extracts concepts from the image. Depending on the data used to train the model, the extracted information from the images (referred to as the concepts) can be a noun (e.g. vase, horse, car, etc.) or an observation (e.g. a man on a horse, a frisbee in the air, etc.). These concepts are passed to a Recurrent Neural Network (RNN) which uses the concepts along with a corpus (database of manually generated captions from the same domain) to generate the captions. The RNN uses the concepts as inputs and calculates the probability of the next word in the caption based on the text in the corpus. This method of generating a sentence is unique to this domain, therefore some of the errors in the sentences are unique to these text generating applications.

Our dataset of captions generated by NeuralTalk2 consisted of 8,000 unique captions of which 2,387 were randomly and manually analysed. The errors detected are shown in Table 1 above.

We then examined a number of smaller datasets of captions generated by image to text generating tools, which were provided to us by NIST. The same errors as above were present in each dataset we examined but to varying degrees of severity.

4 Experiments

Our experiments revolve around the three error types from our manually classified dataset which cannot be solved by traditional n-grams method. While each of the three error types needs a different technique for detection, our aim is to automate the necessary pre-processing of the data so that we can then feed the features necessary for detection of each error into a learning model.

4.1 Incomplete Sentence Errors

The maximum length of our captions is 14 words long. In order to use each word as a feature, we need to have an equal number of words for each caption. We do not want to lose information by shortening the captions, so we padded each caption in the dataset, where necessary, at the start with "unk" to make each caption 14 words long. We then convert each word to its Part Of Speech (POS) tag using the NLTK library. We then hash all the POS giving each POS a unique number. See Table 2 for an example of the various stages a caption of insufficient length goes through to be prepared for the learning model.

The insertion of the word "unk" at the start of each caption which is less than 14 words long, does not influence the learning model as we drop these "unk" features and only focus on the pattern of the POS tags at the end of the sentences to detect the incomplete sentences.

We consistently get a success rate of over 80% for identifying the incomplete sentences.

Table 2. Preprocessing Example for Learning Model.

Step of Process	State of Data
Caption	a bike is parked next to a wall
Caption Expanded	unk unk unk unk unk unk a bike is parked next to a wall
POS	JJ JJ JJ JJ JJ JJ DT NN VBZ VBG IN DT NN
Hash (POS)	1 1 1 1 1 1 2 3 4 5 6 2 3

4.2 Redundant Repetition (* and *)

Our approach to identifying this error type is based on the observation that any caption which has repeated nouns or adjectives, is usually symptomatic of this error type. While there could be occasional exceptions to this rule e.g. "a woman is handing a phone to a woman", we have not encountered a caption like this yet. If a caption repeats a noun (e.g. "a man is wearing a tie and a tie", "a group of people sitting around a table with a table", etc.), or an adjective (e.g. "a girl is holding a green and green box", etc.), we will tag this caption as having the "* and *" error type. The learning model needs information specific to these captions, but we will not interfere with the POS tags already prepared for the incomplete sentences.

Instead we extract the nouns and adjectives from each caption and write them to an excel spreadsheet. A Python script goes through each row checking if there are duplicate nouns or duplicate adjectives and tags each caption accordingly. This tag is an additional feature which we add to the features already prepared for incomplete sentences. We add this feature to the start of the rows being fed into the language model, and now we have 15 features. Note again we have not interfered with the information needed to identify POS tags.

The learning model now detects these duplication errors over 80% of the time.

4.3 Nonsense Errors

Nonsense is the most challenging of the three errors to detect as there are so many reasons why a caption can be considered nonsense. The nonsense can revolve around a verb, a subject doing something impossible (e.g. “A man is flying through the air”,” A dog is holding a glass”, etc.) or adjective (e.g. “ a fluffy woman”, etc.). An item can be in an implausible location or position (e.g. “A bench in a car”,” A cat on a cats head”, etc.). There are many different variations of this theme.

In this paper we will focus on what is plausible regarding verbs. We begin by extracting the verbs from our manually classified dataset, yielding a total number of 22 verbs. We then make a decision on what subject types can precede each verb in the dataset, and we end up with 5 sets of subjects, see Table 3.

Table 3. Subject Types and Corresponding Verbs They Can Precede.

Subjects	Verbs
Human	“brushing”/ “cutting”/ “smiling”/ “talking”/”posing”/”riding”
Human/animal/primate	“jumping”/ “playing”/”laying”/”running”/”wearing”
Human/animal/primate/bird	“watching”/ “eating”/“holding”/“looking”/“sitting”/“sleeping”/“standing”/“walking”
Human/primate	“throwing/spitting”
Human	“driving”
Bird/other objects	“flying”

We utilise a variation of the n-gram method to identify nonsense statements which contain these 22 verbs. We gather captions from Wiki Commons, Google Image, and GCC containing these 22 verbs. We then extract: (a) all the nouns preceding the verb in the captions, (b) the verb, and (c) the first noun following the verb. Refer to Table 4 below for examples of captions and corresponding 3-grams. We build a corpus of 3-grams relating to the 22 verbs we extracted from the dataset.

Taking a similar approach to our dataset, we iterate through the captions extracting: (a) The nouns before the verb, (b) The verb, and (c) The first noun after the verb.

We then check the nouns before each verb, and check that one of these nouns belongs to a member of the subject type that we have deemed capable of executing this verb. If this is the case, we discard all the other nouns preceding the verb so that we are left with first 2 words of our 3 gram namely the subject and the verb.

If this is not the case, we check this caption to see if we need to include a new noun in our subject type for this verb, and if not we label the caption as nonsense as we have a subject preceding a verb where it should not, (e.g. a dog is singing, etc.).

Next, we extract the first noun after the verb, and now we have our 3-grams from our dataset, taking the following form: ‘the subject before the verb+’ ‘the verb+’ ‘+ the first noun after the verb’.

We check if these 3-grams exist in our corpus of 3-grams, and if they don’t, we label the sentence as nonsense. If it does, we label the sentence as plausible.

Our method correctly identifies the nonsense and plausible sentences with 63% accuracy. Analysing the results further, we have 125 nonsense captions, 97 of which are correctly detected giving us a 78% success rate. We have 670 plausible captions of which 410 are correctly detected, giving us a 61% success rate.

The nonsense captions which go undetected had the following errors which were not picked up by our method, “The woman is holding a puppy in her mouth”, or “ the girl is holding a pink cat”. Our method produced the following 3-grams “woman holding puppy” and “girl holding cat”, these n-grams did exist in our corpus so the tool labelled the captions as plausible, and as we are only working on verbs we do not expect to pick up on these errors in this work.

5 Conclusions and Future Work

The main contribution of this paper is to present an ongoing Automatic Error Detection for image to text tools projects that aims at building the first corpus, at the best of our knowledge, to be used for these types of tasks.

We believe that injecting knowledge into applications is necessary for improving common sense in artificial intelligence. In the short term/immediate future we intend to continue our work regarding extracting subject-verb-object relationships for detecting nonsense and in particular, we want to place a special focus on extending the corpus, and developing further techniques that will enable the maintenance of a compact corpus capable of providing extensive knowledge relating to verbs to our application.

References

1. Levenshtein, L.: Binary codes capable of correcting deletions, insertions and reversals. 1966. Soviet Physics Doklady, 10, 707–710.
2. Heidhorn, G., Jensen, K., Miller, L., Bird, R., Chodrow, M.: The EPISTLE text-critiquing system. 1982. IBM Systems Journal 10.1147/sj.213.0305.
3. Mayes, E., Damerau, F.: Context Based Spelling Correction.: 1991: Information Processing and Management 27(5): 517-522.
4. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations.: 2018, In Proc. of NAACL

5. Devlin, J., Chang, M., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding.: 2018, arXiv preprint arXiv:1810.04805
6. Wang, C., Liang, S., Zhang, Y., Li, X., Gao, T.: Does it Make Sense? And Why? A Pilot Study for Sense Makeing and Explanation.: 2019, arXiv preprint arXiv:1906.00363.
7. Forbes, M., Choi, Y.,: Verb physics: Relative physical knowledge of actions and objects.: 2017.: ACL.