CMPS 140 Final Report: Lazytag Bot

Camille Rogers
Baskin School of Engineering
University of California, Santa Cruz
carogers@ucsc.edu

Edgar Martinez
Baskin School of Engineering
University of California, Santa Cruz
emarti43@ucsc.edu

Crystal Lin
Baskin School of Engineering
University of California, Santa Cruz
crklin@ucsc.edu

***Abstract*—The main goal for this project was to create an artificial intelligence that can produce a comment that can fool the common redditor. Our other goals include taking in the input of an image, choose the correct subreddit to train our Markov model on, produce a comment based on the image with our Markov model, and post that comment to Reddit.**

Our final results of the Lazytag Bot is having to manually input the image universal resource locator (URL) directly to our image classification model, Clarifai, and produce a comment with our Markov model with the most relevant subreddit based on the image.

***Index Terms*—artificial intelligence, machine learning, image classification, markov chains, natural language generation**

I. Introduction

"Wouldn't social media be so much easier if you didn't have to agonize over how to comment that friends mundane photo? You want to seem like you paid attention– without spending the time. To solve this worldwide problem, we decided to come up with Lazytag Bot, the comment generator for millennials with ultra-short attention spans."

– The Creators of Lazytag Bot

Natural language generation (NLG) is a research domain that sits at the forefront of the field of artificial intelligence. Ever since artificial intelligence was conceptualized in science fiction, researchers have been trying to create a machine– an algorithm– that can generate speech comparable to human speech. In our project, we wanted to explore NLG using Markov chains, a classic method of text generation, in a fun, silly context. We are interested in discovering if higher-level NLG ideas, such as complex sentence structure and topic persistence, can be achieved through the simple Markov chain probability models.

Our project aims to implement an NLG solution that combines both image classification and Markov chain text generation that can generate a sentence around an object identified from an input image. This workflow has three separate phases: image classification, keyword filtering and corpus generation, and sentence generation. Our implementation deviates from the general Markov chain implementation because we want our Markov chain model to generate a sentence specific to a topic, not just any random topic from our corpus. To test our model, we decided to use Reddit [10] as our platform of choice. We chose Reddit because it has a well-structured collection of Reddit comments that make gathering training data a less cumbersome procedure.

A. Goals and Impacts

As described in the quote above, Lazytag Bot was created with the silly idea of automatically captioning friends photos for convenience. Aside from the fun, we think there are also more serious applications of our project as well. The most direct application is to use artificial intelligence to automatically generate image alt-text for web images. This is extremely helpful to users with vision disabilities as it allows text-to-voice systems to "read" images that have not been manually captioned by the author.

Aside from direct image to text applications, Markov chains that generate from specific keywords have interesting applications on their own. One example would be the increase of "creativity" in artificially intelligent assistants. Rather than always repeat the same response when given a unique keyword, assistants that could generate many different sentences about the same topic are much more interesting and seem more intelligent.

II. Related Work

When we started searching topics that involved both image classification and text generation, we encountered two works that inspired us and helped guide the realization of our project. They introduced us to many different tools of the trade, and ultimately, we ended using some of the same libraries they did.

A. Automatic Image Captioning

During our research we found an article titled, "Automatic Image Captioning using Deep Learning in Pytorch," by Faizan Shaikh [12]. This piece was a wonderful, in-depth explanation of a project that generated textual image captions based on the objects and actions in the image– very similar to what we were attempting to create.

Shaikh's algorithm utilizes Microsofts COCO dataset [7] to train a model comprised of a convolutional neural network (CNN), the encoder, fed into a recurrent neural network (RNN), the decoder. When we first found this example, we had not progressed far enough in our coursework to attempt creating neural networks, but we were very interested in the

COCO dataset. As we will discuss later, we originally did try to use the COCO dataset in our own project.

B. /r/SubredditSimulator

The Subreddit Simulator [11] is a fully-automated algorithm that simulates a subreddit with bots that creates both posts and comments, by redditor Deimorz. This project inspired us to create a "bot" of our own that generates a comment. The Subreddit Simulator also introduced us to Markovify [4], a nifty library that has the basic functions of Markov chains. Although we were not able to actually go in and see the code behind the magic, finding Markovify helped us better understand ways of implementing Markov chains.

III. Previous Trials

In these first few weeks of the project, we really focused on finding the right dataset and methodology for this Lazytag Bot. When we wrote our project proposal, we had a general idea of the concepts we wanted to incorporate into our project– but due to the newness of it all, we weren't certain on what we were going to implement our machine learning and natural language generation ideas with.

We first struggled with finding the right dataset to use for Lazytag Bot. Initially, we found a few different open source datasets (including COCO [7] and ImageNet [6]) but needed to look for some tutorials on how exactly to use the datasets. During this time, we tried all kinds of tutorials– including Shaikh's "Automatic Image Captioning using Deep Learning in PyTorch" [12], but through trying all these different options we came to the conclusion that the time (and computational power) we spent training an image classifier could be better spent on more interesting parts of the project.

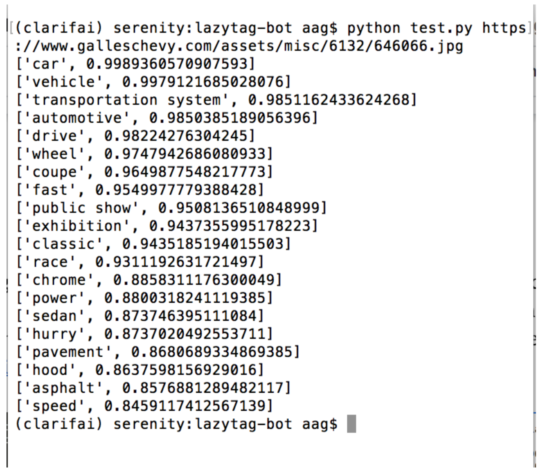
In the end, when looking for pre-trained models, we found Clarifai [2]. This model takes in the input of an image and returns a list of probabilities and associated labels of what may be contained in the image. This is very helpful for us because we can take the keywords with the highest probabilities and write a comment associated with the objects found.

We eventually decided to use the popular method of using Markov chains to generate sentences with the Markovify model, but we knew we needed to alter the model to be more specific to our keyword inputs. During this point in time we had come up with three separate solutions: A) playing with the weights within the Markov model, B) tailoring the corpus to be more domain specific (to our keywords), and C) searching through the Markov model for our keyword and working out a path from that node.

After considering on feasibility, we decided to go for the second solution, as we did not have enough time to try all three. We adjusted our model to add a segment where we select a specific subreddits comments to train on. In order to decide which subreddit we would train our model on, we use the labels outputted by Clarifai and compare them to the descriptions of different popular subreddits.

IV. Data

Lazytag Bot utilizes three separate corpora: the Clarifai training data [2], a subset of the Reddit Comment Dataset by Pushshift.io [9], and the "news" subset of the Brown University Standard Corpus of Present-Day American English (Brown Corpus) [1]. The Clarifai training data is not directly used in our model, it is used to train the Clarifai image classifier ahead of time. When we make the API calls to the Clarifai servers, we are accessing a pre-trained model. The other two datasets are both used in the training of our Markov chain model.



```
(clarifai) serenity:lazytag-bot aag$ python test.py https
://www.galleschevy.com/assets/misc/6132/646066.jpg
['car', 0.9989360570907593]
['vehicle', 0.9979121685028076]
['transportation system', 0.9851162433624268]
['automotive', 0.9850385189056396]
['drive', 0.98224276304245]
['wheel', 0.9747942686080933]
['coupe', 0.9649877548217773]
['fast', 0.9549977779388428]
['public show', 0.9508136510848999]
['exhibition', 0.9437355995178223]
['classic', 0.9435185194015503]
['race', 0.9311192631721497]
['chrome', 0.8858311176300049]
['power', 0.8800318241119385]
['sedan', 0.873746395111084]
['hurry', 0.8737020492553711]
['pavement', 0.8680689334869385]
['hood', 0.8637598156929016]
['asphalt', 0.8576881289482117]
['speed', 0.8459117412567139]
(clarifai) serenity:lazytag-bot aag$ 
```

Fig. 1. A example output of a Clarifai classification of a sample image of a car.

A. Data for Image Classification

Clarifai Training Data. For the image classification portion of this project, we have decided to leave this part to a pre-trained model from Clarifai [2]. The models from Clarifai take an image as input and returns the probability of objects that may be presented in the image. Clarifai uses deep convolutional neural networks in order to train their model on billions of training samples. Specifically, they use their Clarifai Object Recognition Engine, which added millions of images to their models repository, to train their model.

Clarifai claims its unbiased artificial intelligence because of how they trained their model. The Data Strategy Team at Clarifai made sure to train the model on different variations of objects and emotions. For example, if they wanted their model to recognize the emotion happiness they made sure to train Clarifai with multiple variations and examples of the emotion. If you were to train the Clarifai model to satisfy your own project you need to make sure your data is as relevant as the association of the project. Meaning, if your AI needs to accept the input of blurry images, you should train the model to identify blurry images. The team at Clarifai makes sure to be aware of web scraping when training their models and assessing their neural network models and is careful at identifying the biases of web scraping.

B. *Data for Sentence Generation*

Reddit Comment Dataset. Around the time of the progress report, we were initially planning to download the set of comments of a given month, but this no longer became a viable option as we realized that reading a 99 GB every time we ran the bot was too time-consuming. It had comments from every subreddit at the time, and filtering through the subreddits was not efficient. So we switched to using the Pushshift API [9] to get live comments from Reddit, reducing the response time on our bot dramatically. We generated a text corpus from 300 Pushshift API calls, each containing around 100 comments from the chosen subreddit.

Brown News Dataset. The Brown Dataset was created in the 1960s and compiled of around 500 examples English language text separated into subsections [1]. We used the "news" section of the corpus in our Markov model portion of the Lazytag bot because we realized, depending on the subreddit we trained the model on, that the comments outputted were severely not grammatically correct. An example being a comma starting at the beginning of the comment. In order to fix this issue, we added the Brown Corpus in addition to the Reddit comments to train our Markov model because it added more examples of sentences with correct grammar structure for the model to train on. This also added some issues for the comments generated because although it fixed the grammatical issues, it added random topics that are not related to the image we wanted to comment for (see Section 6b).

V. Methodology

In this section, we will clarify the input and output behavior of Lazytag Bot and will detail the step by step sections of our project (especially since those steps have been updated). Lazytag Bot takes an image from a social media post and generates a complete (albeit silly) comment that relates to whats going on in the image. The structure of our system is depicted in Figure 2.

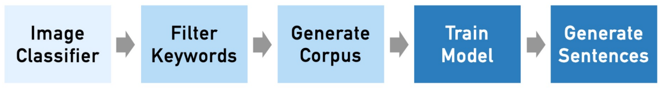
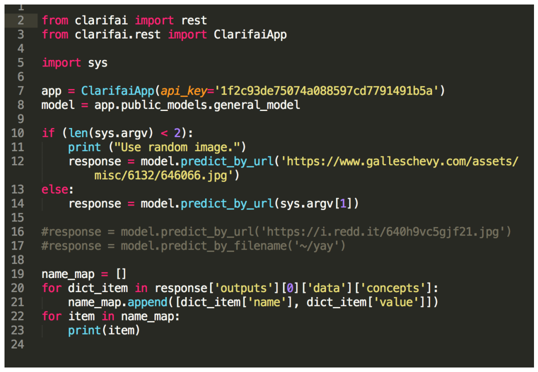


Fig. 2. The workflow diagram depicting the phases of our project.

A. *Classifying Images*

Library: Clarifai API [2]

The Clarifai image classifier is a deep convolutional neural network that uses their Clarifai Object Recognition Engine, which added millions of images to their models repository, to train their model. Anyone also has the ability to train the Clarifai model for their own individual projects. We implemented the Clarifai model for Lazytag Bot in order to identify what is portrayed in an image posted on Reddit. We ran the Clarifai model by using the URL of the image we want to identify. From there Clarifai outputs a list of labels correlated probabilities of what is contained in the image. Figure 1 is the output generated from a sample image of a car.



```python

from clarifai import rest
from clarifai.rest import ClarifaiApp

import sys

app = ClarifaiApp(api_key='1f2c93de75074a088597cd7791491b5a')
model = app.public_models.general_model

if (len(sys.argv) < 2):
    print ("Use random image.")
    response = model.predict_by_url('https://www.galleschevy.com/assets/
        misc/6132/646066.jpg')
else:
    response = model.predict_by_url(sys.argv[1])

#response = model.predict_by_url('https://i.redd.it/640h9vc5gjf21.jpg')
#response = model.predict_by_filename('~/yay')

name_map = []
for dict_item in response['outputs'][0]['data']['concepts']:
    name_map.append([dict_item['name'], dict_item['value']])
for item in name_map:
    print(item)

```

Fig. 3. Snippet of the code used to make a Clarifai API call.

B. *Filtering Keywords*

Library: Natural Language Toolkit (NLTK) [8], Inflect [3]

Given the class labels generated by Clarifai, we generated different forms of the word lemmas then checked for banned words. The reason that we needed to filter some of the keywords provided by the Clarifai model because of some labels outputted could be interpreted by Lazytag Bot as being inappropriate, not safe for work (NSFW), or just not relevant for comment generation, and could, therefore, output inappropriate comments. Examples of banned words are no human and nude.

When generating the comment corpus (the following section) we match the keywords to words in various subreddit descriptions. We found, however, that because the matching algorithm is looking for exact strings, the keyword "car" would not match "cars" in the description. To remedy that, we generated different forms of the keyword from the keywords lemma by first lemmatizing with the NLTK function *lemmatize()*, and then searching through WordNet [13] for other morphological forms with the function *derivationally_related_forms()*. We generated the lemma plurals with the Inflect function *plural()*.

C. *Generating Comment Corpus*

Library: Keyworddit API [5], Pushshift.io API [9]

Using the Keyworddit API [5], we fetch common keywords from of popular subreddits. To find a suitable subreddit, we compare subreddit keywords to our filtered class labels. We use a simple tracking system of weights of one added to the subreddit that has any of the filtered keywords, and the subreddit with the highest weight is chosen to be the source of our Markov model training data.

To train the Markov model we then compile comments from the subreddit as training data for our Markov model by taking them from the Pushshift.io API [9]. We decided not to train our model from all of Reddit because we didn't want our bot to read a large quantity of data every time it had to make an API call, it was too time-consuming. Every time the Lazytag bot makes a call to the Pushshift.io API it grabs 100 comments to train our Markov model on. From grabbing just 100 comments our bot took around a minute to take the comments used to train the Markov model and output a series of comments, so we didn't want to inefficiently grab more comments because the time was already lengthy.

D. Training the Markov Chain Model

Library: Markovify [4]

Given a set of sentences, Markovify tries to generate a sentence based on the corpus of data given. With the corpus of data, Markovify creates a Markov model, where the states are words, calculating the probability of one word following the previous one. An example is shown in Figure 4.

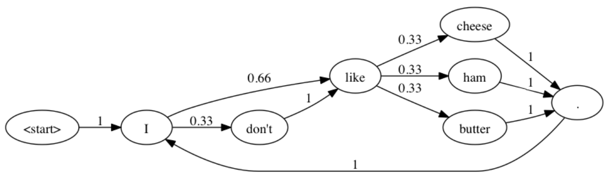


Fig. 4. A simple example of a Markov chain.

The sentence generated can be any given length, although the size of the corpus plays a factor in the success rate of the sentences generation.

E. Stretch Goals

Since the beginning, we've had several stretch goals for Lazytag Bot. These goals include grabbing images and posting comments to and from Reddit, combining Markov chains, injecting jargon in generated sentences, and simple sentiment analysis. Of that list, we achieved two, combining chains and jargon injection. The other two goals have been left as possible future work (and is discussed in Section 7A).

Combining Chains. Instead of just having simple sentences with just one Markov Chain (one input classification), we are interested in creating compound sentences that combine two or more chains (more than one classification), thus making the sentence more relevant to the pictures.

We chose to implement this in our project in an attempt to make more coherent comments, and we did so by creating an additional Markov model from the Brown corpus. We chose to combine both the model generated from the subreddit comments and the Brown Corpus to create a more cohesive Markov model.

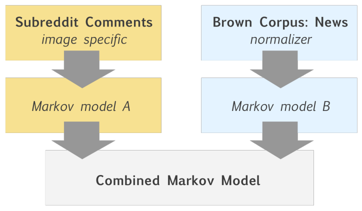


Fig. 5. The structure of our combined Markov chain model.

Jargon Injection. Originally, to make our comments seem more like "internet speak," we thought it would be interesting to add specific internet acronyms (such as the infamous "lol") before or after comments to mimic the way humans do in real life. What we underestimated was the power of the Markov chain model. The model easily copied the pattern of speech redditors used when commenting online. As mentioned above, we actually encountered the opposite problem where the model generated sentences that had too much internet jargon.

VI. Evaluation

Although this project does not have a particular statistical metric for evaluating success, our final goal has always been for Lazytag Bot is to generate a comment that could fool the average redditor! For evaluations during this training process, we manually determined if the sentence generated from an image could pass as a real comment.

In the rest of this section, we detail the observations we had while manually evaluating the sentences generated by Lazytag Bot.

A. Image Classification

The image classifier we used (Clarifai) worked well with providing relevant labels. However, we found that in the cases where a person was doing a particular action, the classifier struggled. The image classifier works better when describing the objects characteristics (people, animals, cars, etc.) rather than when describing the actions a person or animal could be seen doing in the image. But most of the time, the classifier was good enough to find a suitable subreddit.

B. Corpus Generation

When we were generating the training corpus from subreddit comments, we found that although the comments grabbed were good data, if the algorithm chose a subreddit that was not entirely related to the image, the sentences generated based on the comments would be pretty far off topic. An example would be when the algorithm correlated an image of a basketball player to the subreddit /r/gaming. Although the two topics may have many keywords in common (such as "game" or "competition"), the Markov chain model was trained completely on comments about internet gaming, thus generated comments wildly off topic. The difference between basketball and gaming was multiplied many times by the Markov chain model.

To remedy this, we could possibly generate a less-skewed corpus using a set of comments from several subreddits, rather than from just one. However, the bot wouldn't retrieve the same amount of comments from each subreddit but instead would change the amount based on a weighted value of keyword frequencies from each subreddit. Having many comments from a variety of subreddits will help in instances where the top-scoring subreddit chosen for the comment corpus does not describe the image accurately.

C. Sentence Generation

When we were first only using comments as training data for the Markov model, we found that the comments generated were not as structurally sound as we would have liked. The sentences seemed to be a jumble of internet phrases thrown together. We hypothesized that this was because of two reasons: one, we did not have enough training data, and two, internet speaks lacks structure.

The first problem was difficult for us to solve because we were using the Pushshift.io API calls and the number of comments we were requesting was already taking longer than we wanted. Instead, we focused on the second reason and tried to introduce more structure by adding the Brown News Corpus as additional training data.

The Brown Corpus did indeed give our sentences noticeably more structure, but there was one slight problem. We observed that on occasion, the model would generate sentences that clearly reflected the two different types of corpra. One clear example is:

"lol did you go to the gym today bro the flood was devastating"

In this example, the input image was a woman on a treadmill (thus the subreddit chosen was workout-oriented). Both the influence from the image and that from the news corpus can be clearly seen.

The most likely reason this happens is that the method we used to combine the two training corpra has caused a nearly disjoint Markov chain model, where there are two separate clusters of words rather than one merged one. What we could do to fix this is to find a way to extract only the grammatical structure of the Brown Corpus and not utilize the semantic meaning.

VII. Conclusion

Although we did not have a specific metric in which to measure our outputs, we found that using training data compiled on a pool of subreddits comments led to grammar as bad as real comments! The Markov chains didn't pay much attention to the grammatical aspects of sentences, leading to the generation of nonsensical comments. Nevertheless, Lazytag Bot was a rewarding experiment on combining social media and artificial intelligence in order to fool the common redditor. Our project generated fun, quirky sentences that exaggerated the strange correlations found in real Reddit comments.

"Looks awesome tho Thank you, 3/10 landing."
— Lazytag Bot

Fig. 6. Funny quote generated by our beloved Lazytag Bot.

A. Future Work

Moving forward, there are a few aspects of this project we feel like we could improve on or wanted to build on top of (but did not have the time to).

Sentiment Analysis. Among the Reddit Data Tools functions, there is a program that conducts general sentiment analysis on comments in our corpus. This could possibly be interesting in honing-in on the feeling of the comment generated. We were really interested in attaching an emoji to the comment that represented the overall sentiment of the generated comment.

Interfacing with Reddit. In our milestones table below, it can be seen that in week 9, we wanted to implement a Reddit bot that could actually post on Reddit. We found out, however, that creating such a bot required moderator privileges of a subreddit (probably to avoid abuse of an automatic text generator), and we did not have the time to figure that out. In the future, perhaps, it would be fun to create a space where we can actually scrape images and post comments with Lazytag Bot.

Commenting with a Voice. This idea came to us much later, but we were interested in possibly extending the "uniqueness" of the comments (currently it tries to be unique to the image) to a specific user. If could be an interesting experiment to see if we could encapsulate and mimic an individuals commenting style with our bot.

VIII. Milestones

DATE	TASK	PROGRESS
01/23/19 Week 3	Become familiar with the dataset	Done
02/06/19 Week 4	Create a model in order to analyze images	Done
02/15/19 Week 5	Become familiar with parsing was contained in the image	Done
02/20/19 Week 6	Create grammar for English from a given dataset	Not utilized
02/27/19 Week 7	Pull and image from a website and classify it	Done
03/06/19 Week 8	Create a sentence involving what was contained in the image	Done
03/14/19 Week 9	Post the generated sentence to website under image	No Permission*
03/17/19 Week 10	Fix any bugs/Catch up on any tasks not completed	Done

*Allowing a bot to post to Reddit requires moderator-level privileges.

References

[1] Kucera, Henry and Francis, Nelson W. "Brown Corpus." *Brown Corpus*.
[2] "Clarifai Developers." *Clarifai*, clarifai.com/developer/guide/.
[3] Jazzband, "Inflect", *PyPI*, pypi.org/project/inflect/
[4] Jsvine. "Jsvine/Markovify." *Github*, github.com/jsvine/markovify.
[5] "Keyworddit." *Keyworddit*, keyworddit.com
[6] Li, Fei-Fei et al. "Imagenet." *Imagenet*, image-net.org/.
[7] Lin, Tsung-Yi et al. "Microsoft COCO." *COCO*, cocodataset.org/.
[8] "NLTK Project." *NLTK*, 17 Nov. 2018, nltk.org/.
[9] "Pushshift.io." *Pushshift.io*, github.com/pushshift/api.
[10] "Reddit." *Reddit*, reddit.com/.
[11] "r/SubredditSimulator." *Reddit*, reddit.com/r/SubredditSimulator/.
[12] Shaikh, Faizan. "Automatic Image Captioning Using Deep Learning." *Analytics Vidhya*, 2 Apr. 2018.
[13] "WordNet." *Princeton University*, wordnet.princeton.edu/.