ToastBot: An Automatic Compliment Generation System

CS 497: Applied Deep Learning

Oghenemaro Anuyah and Daniele Moro

---

1 INTRODUCTION

We conducted this project in order to build a deep learning model that can automatically generate compliments for users, given their picture and a short description of their mood. With this in mind, we anticipate that users feeling sad or depressed would feel better by receiving automatically generated personalized compliments.

Although there are a number of existing models for machine translation and image captioning, to the best of our understanding, none of these models are built to generate compliments given the input of the user's current emotional state and the image of the user. Online compliment generators are widespread, but most of them simply choose a random compliment out of a pre-written list [1–3]. Unlike these systems, our work is able to generate compliments that are more relevant and personalized for the target the user as it leverages both the user's written emotional state and the image of the user.

2 PRIOR WORK

Due to the fact that our task is to predict a sequence of texts given both visual and textual features, we leverage a sequence-to-sequence (seq2seq), many-to-many architecture [10]. As a starting point to designing our model, we discovered two relevant seq2seq model architectures: (i) models that take text as input and generate text as output, and (ii) models that take images as input and generate text as output. For the former, this architecture is commonly utilized for machine translation tasks, i.e., translating texts from one language (e.g., English) to another (e.g., French). For the latter, these models are commonly used for image captioning tasks, i.e. describing what is occurring in an image. We expand upon these works by combining both strategies to solve the task of automatic compliment generation in a novel way.

3 DESIGN SETUP

3.1 Tools

We used a Jupyter notebook for design, exploratory analysis, and experiments. We wrote the project in Python 3 and used Keras with a TensorFlow backend as the preferred deep learning library. We used the Flask package to create a simple web application that provides an interface for the user to interact with our model.

3.2 Data

For creating our model, we took advantage of data from Reddit (specifically the subreddit r/toastme). On this platform, users post an image of themselves, along with a short description of their mood (labelled *title*). Given this information, several other Reddit users respond with compliments and positive messages with the goal of improving the mood of the original poster. In order to create a dataset comprised of this information, we utilized the Python library PRAW. We used this to act as a wrapper around the Reddit API. Since the Reddit API does not allow us to collect more than the top 1000 posts from any one category, we downloaded the top 1000 upvoted posts, the top 1000 most commented posts, the top 1000 newest posts, and the top 1000 hottest posts. We combined all of these posts together to build our corpus of 4000 reddit posts and their corresponding top-level comments

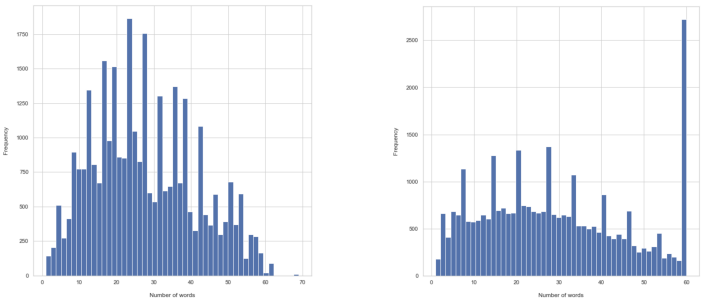
The data gathered was in the following format: $< title, comments, image\_URL >$

3.3 Pre-processing and Exploratory Analysis

Data gathered from Reddit, although structured, was not in a "cleaned" format (i.e., noise-free such that our model can work best with). We therefore performed a number of pre-processing steps on our data before feeding it to our model.

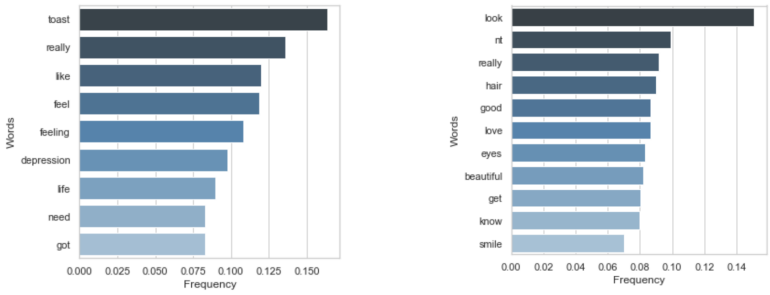
3.3.1 Top-k comments. Given that the 4,000 posts we gathered from Reddit each have a number of comments associated with them, we only selected their top 10 comments respectively. We empirically verified that building a model limited to only these number of comments led to better results. For each given title, we created comment-title pairs and stacked them resulting in a total of 31,685 examples of a title, comment, and URL tuple.

3.3.2 Text processing. To process our text, we normalized both the title and comment text to lowercase, removed all punctuation (e.g., periods, commas, and so on) and digits. In each title and comment, we only retained the top three sentences, as we found that using all sentences often led to noise in our training model. If the first sentence was longer than 60 words, we truncated it to the first 60 words. Finally, we include start ("START_") and end ("_END") tokens on every title. Based on this strategy, we display the distribution of the number of words in both the title and the comment in Figure 1. This figure shows the distribution of the number of words in a title approximates a normal distribution, while the distribution of of the number of words in a comment is right skewed, with a high number of outliers which we truncated to 60 words. We also visualize the top-10 most frequent words that appear in titles and comments in Figure 2.



(a) Title words. (b) Comment words.

Fig. 1. Distribution of words in the titles and comments after limiting them to only three sentences.



(a) Title words. (b) Comment words.

Fig. 2. The top-10 words in titles and comments of our data (excluding stop words).

3.3.3 Image Processing. For input images, we leveraged the pre-processed functions made available by the Keras VGG 19 library. After the pre-processing step, we fed them through the VGG 19 convolution neural network, which achieved state-of-the-art results in the ImageNet Challenge in 2014. We extracted the visual features of the images from the prediction layer of the model, and mapped them to their respective title and comment. This resulted in a vector of 1000 features for each image.

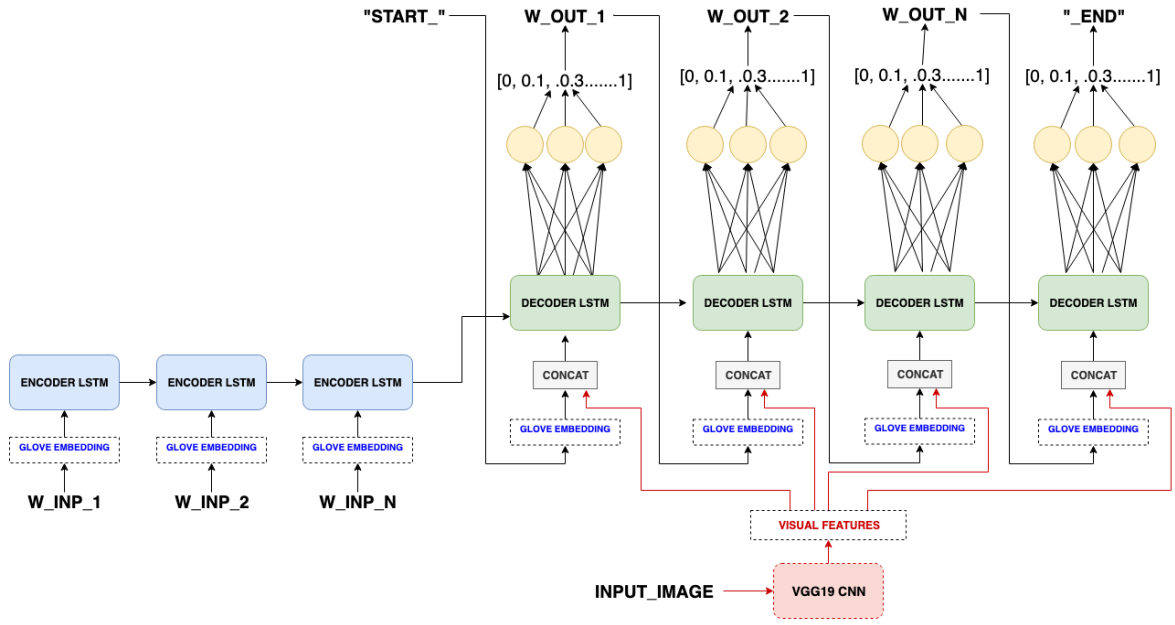


Fig. 3. ToastBot model pipeline.

4 METHODOLOGY

To build our model, we expanded upon the work of existing machine translation codebases [4–6]. Although we started by using a Keras sequential model with a TimeDistributed layer [4], we realized that this architecture was not sufficient to allow our decoder LSTM to leverage the state of the encoder LSTM. Moreover, the decoder LSTM did not adequately take the word predicted in the previous time step as input for the current time step. As a result, we used a sequence to sequence Keras model through the functional API [6]. It is important to note that the specific code base provided was a character-level model. Although we initially examined how this model could be utilized for our task, we observed that a word-level model would be more applicable for learning the semantics of the English language, as it can take advantage of pre-trained word embeddings. Based on our observations, we decided to build upon a codebase which utilizes the Keras functional API with word-level embeddings [5]. We expanded upon this model by adding the image features as input to the decoder LSTM, including pre-trained Glove embeddings, and modifying the task of the model to make use of our Reddit data (instead of a translation task).

Our model takes in as input a sequence of words representing the emotional state of the user. These input words are passed through an Embedding layer with frozen Glove [8] weights, creating 300 dimensional word embeddings, which are given to an encoder LSTM composed of 50 units. The final encoder states are provided as the initial states for a 50-unit decoder LSTM. Separately, the image feature vectors generated from the VGG 19 [9] CNN are concatenated with the word embedding of the word from the previous time step (or "START_" if there is no previous word) and given to the decoder LSTM. The outputs from the decoder LSTM are given to a Dense layer which outputs a probability distribution over a word vocabulary. We use a greedy search and simply take the most probable word to input into the next time step of the decoder LSTM. When the decoder LSTM outputs the token "_END", or outputs more than 100 words, we terminate the prediction and collect our generated compliment. We train the model for 100 epochs on our train set (we initially split our data into train and test sets using a splitting ratio of 90:10), using a batch size of 64 and default learning parameters.

5 EVALUATION

We take inspiration from existing machine translation evaluation metrics by using the BLEU metric [7] to evaluate our model. Although this metric may not fully showcase the capabilities of our model (i.e., even for humans, there are infinite ways we could generate a compliment), we turn to this measure because it is one of the most commonly-used metrics to evaluate generated text. To quantify the importance of the text and visual features, we build three different models for evaluation: (i) a model that only uses text as input, (ii) a model that only

uses images as input, and (iii) a model that uses both text and visual features. As shown in Table 1, we infer that a model built by combining both the visual and textual features produces the best results. Although our model has potential for improvement in gender resolution (i.e., identifying the gender of the user) and providing non-repetitive output text, we are satisfied with its ability to generate compliments much more personal and relevant when compared to the aforementioned tool introduced in section 2.

Only Textual Features	Only Visual Features	Both Textual and Visual Features
0.05927	0.01674	**0.08108**

Table 1. Our experimental results using the BLEU metric on the testing data set. By using both the textual and visual features, we improve results.

6 WEB INTERFACE

To allow a potential user to make use of our automatic compliment generator, we built a simple web server and interface using Python and Flask. As shown in Figure 4, the user can enter their emotional state in a text box, and then upload a picture of themselves. After hitting the "Generate Compliment" button, the user is presented with an auto-generated compliment that we anticipate to be relevant, personal, and flattering.

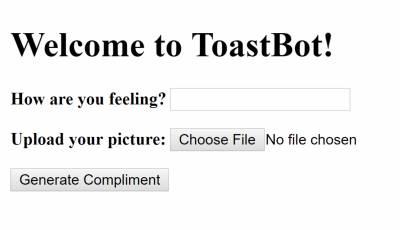


Fig. 4. The ToastBot Web interface. Users can write their emotional state and upload a picture. They will then be shown the generated compliment.

7 CONCLUSION

We present a novel tool for automatic generation of relevant, personal compliments that aims to improve the emotional state of the end user. Our evaluation demonstrates that using both textual and visual features improve results, and our web interface provides an intuitive method for interacting with our model. Future work may lie in gender disambiguation, improving long-distance memory, and collection of more data from Reddit or other sources.

REFERENCES

[1] Accessed: 2019-05-04. Automatic Flatterer. https://www.cse.unsw.edu.au/~geoffo/humour/flattery.html. (Accessed: 2019-05-04).

[2] Accessed: 2019-05-04. Bando Compliment. https://www.bando.com/collections/compliments. (Accessed: 2019-05-04).

[3] Accessed: 2019-05-04. Emergency Compliment. http://emergencycompliment.com/. (Accessed: 2019-05-04).

[4] Jason Brownlee. 2019. How to Develop a Neural Machine Translation System from Scratch. (May 2019). https://machinelearningmastery.com/develop-neural-machine-translation-system-keras/

[5] Maheshwari Devesh. 2018. NMT Keras. (Feb 2018). https://github.com/devm2024/nmt_keras

[6] keras team. 2019. LSTM Seq2Seq. (Feb 2019). https://github.com/keras-team/keras/blob/master/examples/lstm_seq2seq.py

[7] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[8] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[9] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[10] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.