CIS 530 - Milestone 1

Nathaniel Fessel fesseln@sas
Ben Goodman bengoo@sas
Brandon Obas bobas@sas
Palmer Paul palmerpa@seas

April 2, 2019

1 GAN Recipe Generation

1.1 Problem Definition

Generative Adversarial Networks (GANs) are primarily being applied to image data, but some research has been done experimenting with GANs for natural language processing tasks. The problem we want to solve is to see whether GANs can be used to effectively generate text. As an interesting test of this bleeding edge technology applied to NLP, we propose the construction of a GAN which can produce realistic, human-quality food recipes. We will then provide every member of CIS 530 with our GAN-tastic recipe book.

At a high-level, GANs work as follows: Based on a single training data set, one neural network is trained to generate data and the other is trained to classify the data that the first generates. This cycle is repeated using the feedback from the classifier network to help the generative network improve and vice-versa. The generative networks works to generate examples of the positive category, and only examples that are accepted by the adversarial neural network are accepted. That is, the generative neural network is trying to trick the adversarial neural network (hence the name GAN).

1.2 Relevant Papers

- Language Generation with Recurrent Generative Adversarial Networks without Pre-training
- Adversarial Generation of Natural Language
- DP-GAN: Diversity-Promoting Generative Adversarial Network for Generating Informative and Diversified Text
- Recipe generation char-rnn

1.3 Evaluation Metrics

The best way to evaluate our model's output would be to submit a large quantity of our GAN's output to Mechanical Turk, intermixed with bad recipes and true, human recipes, and have humans classify whether or not the output was generated by a human. We will then compute precision, recall, and F-score from this information.

1.4 Datasets

- We would need a large database of human-generated recipes.
- Potential data source for recipes (used for the recipe generation char-rnn mentioned above)

2 Amazon Review Classification

2.1 Problem Definition

Fake product reviews on Amazon represent a serious issue for consumers. Typically, small or foreign companies will encourage their employees to write fake reviews in order to inflate the product rating. Few consumers have the time or means to dig into each review and look for red flags, yet they rely on these reviews nonetheless.

Significant effort has been put into trying to identify fake reviews. We will attempt to train a classifier, using linguistic and distributional features, to identify these reviews. One problem we face is how to obtain gold labels for all instances.

2.2 Relevant Papers

- Spotting fake reviewer groups in consumer reviews
- Identifying fake Amazon reviews as learning from crowds
- Distributional Footprints of Deceptive Product Reviews
- Fake Review Detection: Classification and Analysis of Real and Pseudo Reviews

2.3 Evaluation Metrics

Once we have labeled data, we can compute the precision, recall, and F-measure of our model's binary output. Another possible evaluation metric is Normalized Discounted Cumulative Gain (see first paper above).

2.4 Datasets

- Amazon product reviews
- Large amounts of amazon review data segmented by product type
- Sentiment analysis of Amazon reviews
- More Amazon review data hosted on AWS
- Even more Amazon review data hosted on AWS

3 Résumé Normalization and Bias Analysis

3.1 Problem Definition

The problem that we are solving is making it easier to evaluate résumés fairly. The appearance of a résumé can play a disproportionate role in evaluating the quality of a candidate for hiring purposes. Some companies will throw away a résumé just because they don't like the layout used. Our solution is to extract the information from a résumé and display it to the recruiter in a standardized format so that only a candidates actual qualifications contribute to job offer outcomes. The résumés will be input as PDFs or Word documents and output a txt format with the extracted information. Our planned approach will be to start with a named entity recognition algorithm and refine our model from there. This will also allow recruiters to process résumés more quickly and even filter résumés by different criteria (e.g. GPA, school, etc.).

Additionally, we may look into trying to predict whether a résumé is qualified enough to go on to the next round. The main foreseeable issue with that we'd need labeled training data in the form of information about whether an individual received an offer for the job to which they applied. We looked for this data but had trouble finding a suitable data source because companies cannot publicly share this information without raising privacy concerns.

3.2 Relevant Papers

- Résumé information extraction algorithm
- Résumé parsing and standardization
- Résumé information extraction with cascaded hybrid model
- An Approach to Extract Special Skills to Improve the Performance of Résumé Selection

3.3 Evaluation Metrics

For each of the annotated résumés (see section 3.4) we will look at how many fields our model correctly annotated. We will calculate precision, recall, and F-score, from this information.

3.4 Datasets

We need a collection of résumés that are focused on a specific industry such as marketing or technology along with entity annotations for evaluation purposes. We found such a dataset which we are planning to use.

If for some reason, the dataset above does not have all the fields we need, we can use Amazon Mechanical Turk to have human workers extract information for about 100-500 résumés. This data will serve as a gold standard.

Additionally, if we also attempt the stretch goal we would need data about which jobs that person was accepted for.