Predicting Summary Redundancy

Group 11

I. INTRODUCTION

Text Summarization is a field of Natural Language Processing used to capture salient information from a piece of text. Measuring the success of Text Summarization models is a challenging problem in Natural Language Processing. Conventional methods rely on automatic metrics- metrics that are computed without human intervention - to measure the performance of summarizer output.

Right now, the only way to truly measure the success of our summaries is by using Human Annotators. It is the closest available measurement to the ground truth success. Opting to use an automatic metric instead of human annotation implies that the scores from the automatic metric are highly correlated to the human annotator scores because an increase in a metric's score should indicate an increase in the success of the summarizer's output. However, it turns out that the correlation between human annotator scores and automatic metrics often turns out to be quite poor.

The reason to want to use automatic metrics instead of human annotators is to save cost. Getting humans to annotate summaries costs money and there are also a lot of factors that can throw off scores. If human annotators are mostly giving subjective scores and rarely agree on scores, then the score will not be effective.

In this paper we will try to create a metric that measures the redundancy of a summary. Some of the signs that a summary is redundant are whether or not it repeats spans of words or contains paraphrases of similar sentences. It is helpful to those building summarizers to look at the redundancy of the output because summaries that are redundant may not come across as fluid and human-written to the reader. In order to ensure that this redundancy score is actually a true measurement of the summary's redundancy, we will test our score against a human annotator's score for redundancy.

We derive features from the text of the summaries and input the features into a model to predict a redundancy score. With our best model, we are able to get a redundancy score that has a 0.75 Pearson correlation with the human annotator score for redundancy.

II. DATA

The dataset we will be using is extracted from the CNN and Daily Mail dataset[4]. The CNN and Daily Mail dataset contains various genres of news articles coupled with short summaries. This dataset has been used several times to test summarization models. We will be utilizing a small portion of the dataset that was extracted by Chaganty et al.[2].

The dataset contains summaries created by state of the art text summarizers given the CNN and Daily Mail articles. For each of the 2000 summaries created by the summarizers, two annotators score the summary for redundancy on a 3 point scale, either giving the summary a -1, 0, or 1. In this scale, -1 means the summary is very redundant, and 1 means the summary is not redundant. We will use the annotators' redundancy scores as a response variable to try and ultimately predict the score an annotator would give to a summary. For each summary, there are 2 annotators who provide scores for it. We average the scores given by the annotators to get one average annotator score per each summary. Averaging the annotator scores gives a new domain of the possible scores: $[-1, -0.5, 0, 0.5, 1]$. The dataset does not initially come with any features so we must next derive features from the text of the summary to help our model predict the redundancy score.

III. FEATURES

A. Max N-Gram Repetition

Within the summary, we will count the number of occurrences of each unique unigram, bigram and trigram, separately. This will give us lists L_i for $i = 1..3$ that contain lists of the number of occurrences for n-grams of size i. Then we will compute $\max L_i$ and get three features, one for unigrams, bigrams and trigrams.

B. Longest Repeated Substring

The longest repeated substring problem is a problem where we try to find the longest substring of a string that occurs more than once[5]. In this case, we will

treat each word in a summary as letters, and the whole summary as the string, so that we instead get the longest repeated string within the paragraph. Before calculating, we will remove all punctuation and stop words.

C. Repeated Substrings Combination

This is a metric proposed in a paper from Fan et al. to measure redundancy[3].

$$\text{REDUNDANCY}(S) = \frac{1}{|S|} \sum_{f \epsilon F'(S)} (\#f' \times |f'|)^2$$

where $F'(S)$ contains a set of fragments at least three tokens long that are repeated within the summary, and $\#f'$ is the repetition frequency for fragment f'. Intuitively, longer fragments and more frequent repetition should be penalized more heavily.

D. Embedding Similarity Scores

We want to not only be able to find occurrences of repetition of the same words and phrases, but also find occurrences of paraphrasing. In order to do this, we retrieve embeddings representing each of the sentences in a summary. The first encoder we use is the Universal Sentence Encoder[1]. The Universal Sentence Encoder can create embeddings for many different sizes of text. It provides a very general purpose embedding that can be used in a variety of situations. The next encoder we use is ELMo[6]. ELMo provides embeddings that capture varying characteristics of complex word usage. For both of these encoders, we pass in each sentence of the summary individually and retrieve a list of vectors V.

Then, for each unique pair of sentence vectors A and B in V, where $A \neq B$, we compute the cosine similarity:

$$\theta_{A,B} = \frac{A \cdot B}{||A||\,||B||}$$

and we get a list L of similarities between the vectors for which we compute

$$\text{maxScore}(L) = 1 - \max L$$

and

$$\text{meanScore}(L) = 1 - \text{mean}\, L$$

The first score that computes the max gives us a score close to 1 if there are no paraphrased sentences, and a score close to 0 if there is at least one pair of sentences that are paraphrases. The second score will average the similarity scores and similarly, will give us a score close to 1 if there are no paraphrased sentences and a score close to 0 if all of the sentences are paraphrases of each other.

IV. METHODS

The models used for prediction are Ridge Regression and Support Vector Machine Regression. The response variable implies that this is a classification problem. However, if we treat it like a classification problem then we lose the ordinality of the scores. For example, a summary that is given a score of 1 when it is actually a 3 should not be given the same penalty as if it we predicted a score of 2 for a summary that actually received a 3. For this reason, we chose to predict using a regression. Making it a regression problem not only makes the penalties for the loss function better, but also makes measuring the performance of our model better. Once we predict the scores of our test set, we can simply compute the Pearson correlation score to see whether our model is doing a good job of mimicking human annotation.

We use lasso regression because of its ability to avoid overfitting and perform variable selection. We will test various values of the regularization strength α which is the coefficient of the l1-norm of θ. The objective function we use is:

$$(1/(2 * n\_samples)) * ||y - X^T\theta||_2^2 + \alpha * ||\theta||_1$$

Lasso regression provides a model with low flexibility and to test whether we can improve our results with a more flexible model we will also use Support Vector Regression with more complex kernels. The first kernel we will use is the RBF kernel which is:

$$K(x, x') = \exp(-\frac{||x-x'||^2}{2\sigma^2})$$

For the RBF kernel we will test multiple values of C and γ. The second kernel we will use is a polynomial kernel:

$$K(x, x') = (x^T y + C)^d$$

We will test polynomial kernels of degrees 2 and 3 with multiple values of C, and γ.

The values of C corresponds to a range of how much to allow points to fall outside of the margin on either side of the projected hyperplane. The values of γ correspond to a range for the kernel coefficient.

We split our dataset into partitions of 80% and 20% test. To test the various combinations of parameters we will perform an exhaustive search using 5-fold cross validation on our training set. This means for every combination of parameters we will perform 5-fold cross validation, splitting the training data into 5 partitions and testing on each partition using the other 4 partitions as training. Then, we will average the performance over the 5 tested partitions. The parameters tested for Lasso can be found in Table I and the parameters tested for Support Vector Regression can be found in Table II.

TABLE I
PARAMETERS TESTED FOR LASSO

Parameter	Values
α	0.0001, 0.001, 0.01, 0.1, 1, 10, 100

TABLE II
PARAMETERS TESTED FOR SVR

Polynomial Kernel	
Parameter	Values
$degree$	2, 3
C	0.001, 0.01, 0.1, 1
γ	$n\_features^{-1}$, 0.001, 0.01, 0.1

RBF Kernel	
Parameter	Values
C	0.001, 0.01, 0.1, 1, 10
γ	$n\_features^{-1}$, 0.001, 0.01, 0.1, 1

V. RESULTS

For both of our methods, we retrieve the best parameters by using cross validation. Then, we test on our test set. The best parameters chosen from the exhaustive search with cross validation can be found in Table III.

For Lasso Regression we get a Pearson coefficient, r, of 0.734. This implies that our data may be linearly separable. Additionally, we see the coefficients of features that Lasso chooses to drive close to 0. Lasso decreases coefficients for 4 of the 9 parameters to a value close to 0. We will use the remaining 5 features and test the performance of the Support Vector Regression without these 4 parameters and the full set.

TABLE III
BEST MODEL PARAMETERS AND PEARSON CORRELATION

Model	Parameters	r
Lasso Regression	$\alpha = 0.0001$	0.734
SVR	Kernel=RBF, C=10, γ=0.01	0.76
SVR (w/ Feature Selection)	Kernel=RBF, C=10, γ=0.01	0.75

When we train the Support Vector Regressions, we find that for the two models, with and without feature selection, we get the same set of best performing parameters. This can be seen in II. The SVR with all features gets an r score of 0.76, and without the 4 features that had low coefficients in Lasso, the model gets an r score of 0.75. Considering that the performance with 4 less features is nearly the same as the performance with all of them, we can say that SVR with only 5 features and optimal parameters is the best model. If two models perform nearly the same but one has less features, it makes sense to pick the one with less features because we can get the same performance with lower model complexity.

VI. DISCUSSION

First, the models we use make specific assumptions about the distribution. Ridge regression makes an assumption that the response variable is normal. However, it turns out that we have a left skewed distribution for the human annotated redundancy score, which can be seen in figure 1. We do not choose models that are particular adept at working with a skewed response variable. In future work, we could test models that are specifically geared towards handling skewed data. Another option is to perform transformations to our data to make it more normal. Both of these options could be helpful in improving model performance in the future

From this analysis it seems that measuring redundancy is not that difficult. We are able to get a good Pearson correlation of above 0.7 with less than 10 features. However, we think that this dataset is particularly easy to work with. The summaries were quite short and so we think many of the aspects of redundancy that stick out to annotators would be easily spotted and therefore easily captured by simple features. It would be interesting to see how these features perform within a model trained on a dataset

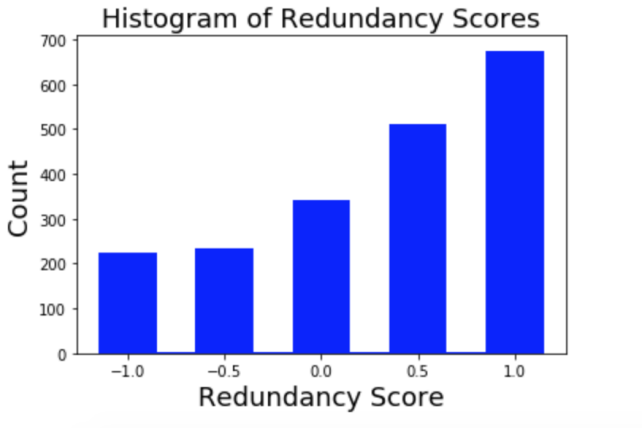

Fig. 1.

with longer annotated summaries.

Additionally, in order to further prove the value of our model, we need to test our model across domains. We want our model to generalize well to many different types of text, not just summaries within the CNN and Daily Mail dataset. Just because it performs well on this dataset does not mean it will perform well on others. The next step would be to find a dataset within a different genre of text and use it in tandem with the CNN and Daily Mail dataset.

APPENDIX

A. *Implementation*

All code and figures used for deriving the features and implementing the models can be found here: https://github.com/AlexanderJGomez/DS5220-Project .

REFERENCES

[1] Daniel Cer et al. *Universal Sentence Encoder*. 2018. eprint: arXiv:1803.11175.

[2] Arun Tejasvi Chaganty, Stephen Mussman, and Percy Liang. *The price of debiasing automatic metrics in natural language evaluation*. 2018. arXiv: 1807.02202.

[3] Lisa Fan, Dong Yu, and Lu Wang. *Robust Neural Abstractive Summarization Systems and Evaluation against Adversarial Information*. 2018.

[4] Karl Moritz Hermann et al. *Teaching Machines to Read and Comprehend*. 2015. eprint: arXiv:1506.03340.

[5] . *Longest repeated substring*. [Online; accessed 5-December-2018]. 2018. URL: https://en.wikipedia.org/wiki/Longest_repeated_substring_problem.

[6] Matthew E. Peters et al. *Deep contextualized word representations*. 2018. eprint: arXiv:1802.05365.

VII. Statement of Contributions

Alexander Gomez: Everything