Classification of Real and Synthetic Sentences Based on Part-of-Speech Distribution

Maitrey Mehta[1] and Neha Kherde[1]

[1]School of Computing, University of Utah
maitrey@cs.utah.edu, neha.kherde@utah.edu

Abstract

The problem of evaluating Natural Language Generation(NLG) models has been persistent and the solutions currently in existence are inefficient in terms of correlation with human evaluation. Moreover, these automatic evaluation metrics require a ground truth to compare with the generated sentence, which may not always be available. Additionally, the possibilities of valid sentences following a certain context can be large and cannot be sufficiently addressed by human generated baselines. An extensive survey about evaluation metrics for NLG can be found in Reiter and Belz (2009). The inefficiency of automatic evaluation metrics calls for better grammar-based evaluation metrics which show strong correlation with human evaluation in terms of naturalness and quality. A comparative study on automatic and grammar-based evaluation metrics as compared to human evaluation is shown in Novikova et al. (2017) and Shimorina (2018). In this project, we intend to develop the first step towards building a generalized grammar based metric for NLG by analyzing models for classifying sentences into real and fake based on their Part-of-Speech distribution.

Introduction

Classification of textual data into real or synthetic is a known and explored problem. Some of the classification models previously seen in literature for this particular task are CNNs (Kim (2014)) and multi-layer perceptrons (Ranjan et al. (2018)), or statistical NLP methods for text categorization (Jacobs (1992)).These methods, however, represent word as features. We detail an approach which instead uses the Part-of-Speech(POS) tag distribution as features which is of special relevance for two factors-

1. Vectorization of words leads to high dimensional sparse data (average human vocabulary is 35,000 words), hence, increasing the computational complexity. POS tags are rather limited and significantly less in number than the vocabulary size.

2. POS tags imply the grammatical structure of a sentence which is an important indicator for grammatical correctness.

In the following sections, we describe our Data and Feature Extraction from this data; we briefly explain the details of the classification models we experimented with followed by a section which discusses its results. Finally, we have a discussion section which entails certain interesting insights which were drawn from the experiments.

Data Collection and Feature Extraction

This particular project requires two different datasets- real and synthetic sentences. For real sentences, we choose the WikiText-2 dataset which is a collection of over 100 million tokens extracted from the set of verified 'Good' and 'Featured' articles on Wikipedia. WikiText-2 is also a good source to get a domain-independent corpora, hence scaling our models to different datasets. The second dataset is that of synthetic sentences for which we used the following techniques-

1. Fake sentences are created by word shuffle(picking any two words and swapping) and word dropping(pick a word and remove it from sentence)Ranjan et al. (2018). This is done multiple times and in combinations to create more sentences.

2. Language Models- Synthetic sentences were generated by SRILM (Stolcke (2002)) on the WikiText-2 corpus and a built-in Harry Potter corpus(to test domain-independence) and Bi-LSTM which was trained on the WikiText-2 corpus.

We convert the sentences into sequence of POS tags by using the SpaCy POS Tagger which has a reported accuracy of 97%[1] and has 17 distinct POS tags. The idea is to represent a sentence by its POS distribution. We consider frequency counts of POS tag bigrams and each POS tag bigram represents a dimension in our feature space with the frequency of the respective bigram being the value along that dimension. We normalize the frequency by the length of the sentence to avoid bias due to sentence length. Given

[1]https://spacy.io/usage/facts-figures

17 POS tags, the number of dimensions would be atmost 17$\times$17=289 which is way lesser than the vocabulary size of the corpus(33279). We observe that our training set could be represented in 205 dimensions + 1 ("OTHER") bigram dimensions. Label 1 is assigned to real and 0 to synthetic sentence vectors. For convenience, we ignore sentences more than 40 words in length.

Classification Methods

The training and development data is created by dividing our sentence set such that one tenth part represents the development data and the rest is training data. The training, development and test data remains the same for each algorithm for a given synthetic corpus to give a fair and consistent comparison. The development set is used to perform analysis on different set of hyperparameters for each algorithm. For convenience, this analysis is done on one of the synthetic datasets. The set of hyperparameters with best accuracy reported on development set is chosen as the set of hyperparameters for testing on test set. These are reported in Table 1 for every algorithm. Moreover, the number of sentences chosen from each synthetic corpus is also the same for consistency. Note that, we have considered equal number of real and synthetic sentences in the training set, development set and test set, i.e, the dataset is not skewed. We have explored the following classification models-

1. **K-NN Classifier-** This algorithm is based on feature similarity. We classify the given sentence as grammatically correct or wrong sentence by determining how closely it resembles to the features of the other sentences. The Euclidian distance from the test point to the training points is calculated and a majority vote among the k closest points decides the label for the test point. We experimented with different k values (k = [10, 30, 35, 45, 50]) and the best k decided from the developset was then used on the testing set.

2. **Neural Networks-** We implemented a simple neural network with a single hidden layer from scratch. We choose sigmoid as the activation function at the end of each layer. This enables us to choose 0.5 as the most appropriate threshold at the output layer to decide the class of the point. The algorithm minimizes cross entropy loss which is suitable given that this is a binary classification problem. The only hyperparameter in this case was the learning rate for which we considered different values for cross validation. To compare the efficiency of our neural network, we compare our model with scikitlearn's MultiLayerPerceptron(MLP) function where the hyper-paremeters were the activation function and the α values (L2 regularization factor). Note that, the number of epochs were decided by looking at the development set accuracy curves (Figures 1 and 2), and the epochs were chosen when the accuracy curve seemed to flatten and showed very marginal increase. Note that it was interesting to see a spike in Figure 1, which was possibly due to overshooting.

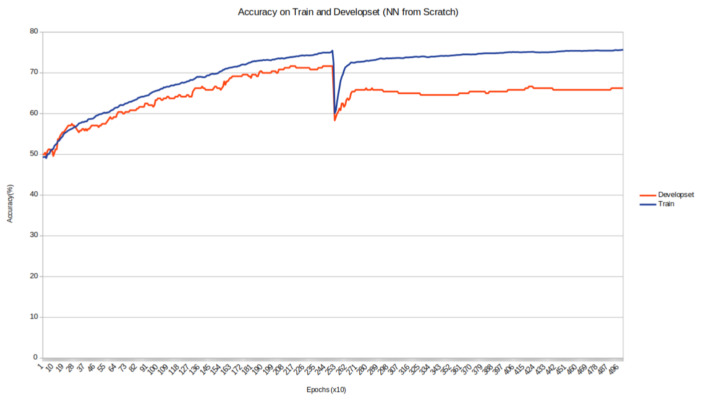

Figure 1: Accuracy Curves for Simple Neural Network.

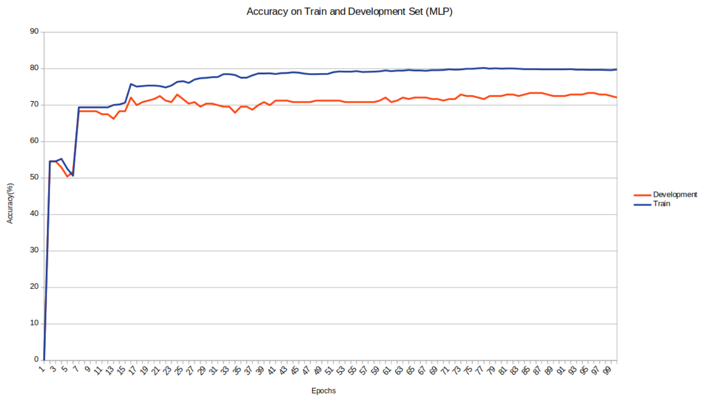

Figure 2: Accuracy Curves for Multi-Layer Perceptron (MLP).

3. **Convolutional Neural Networks (LeCun et al. (1995))-** We implemented a CNN using Keras with two convolutional layers with feature windows of 3x3 and 2x2 for the respective layers followed by a softmax layer at the output layer. To avoid overfitting, we have added dropouts after each convolutional layer. We experimented with the activation functions(ReLU,tanh and ELU) and different dropout values as hyperparameters. The optimal number of epochs was again decided by observing CNN's accuracy curve for development set (Figure 3).

4. **Support Vector Machines (Vapnik (1999))-** Support Vector Machines (SVM) is a linear classifier. This can also be used to prove the existence of the non-linearities in our dataset. We have used the stochastic subgradient

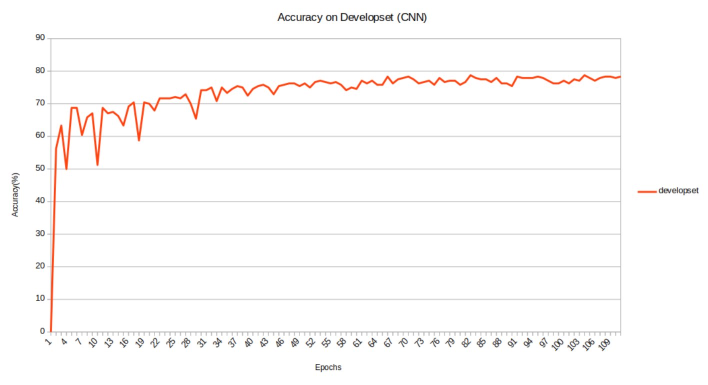

Figure 3: Accuracy Curve for CNN.

approach with hinge loss. The learning rate decays over epochs. We experimented with the different values of learning rate (η) and the regularization factor (C). The accuracy curve for SVM is given in Figure 4.

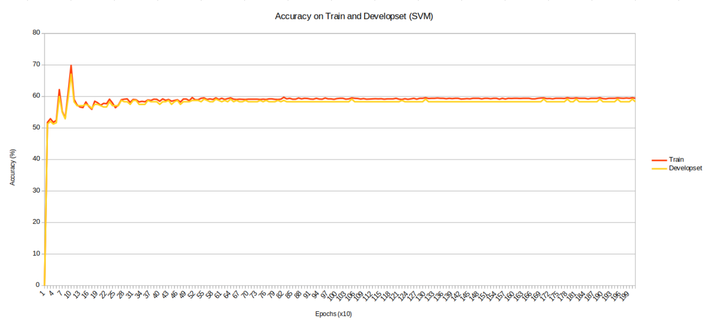

Figure 4: Accuracy Curves for Support Vecor Machines.

5. **Non-Linear SVMs-** One of the shortcomings of SVMs is that it cannot model non-linear data being a linear classifier. Hence we introduce kernels to map the data via a feature transformation into a space where the data is linearly separable. We implemented this model using libsvm library. We experimented with different kernels and regularization factors (C), on the development set. The accuracy of this model over epochs for the training and development set is shown in Figure 5.

6. **Ensemble Method-** We implemented an ensemble of MLP, CNN and non-linear SVM to get relatively better predictions. The final prediction of the ensemble was based on the majority vote of the three classifiers. The hyperparamters for each of these classifiers remain the same as retrieved for them individually.

7. **Bagging using Decision Trees (Dietterich (2000))-** As the main principle behind the ensemble model is to group weak learners together to form a strong learner, we combine at most 10 decision tree classifiers to produce better productive performance than utilizing a single decision tree as part of training. This is done by creating 10 bootstrap samples each of size 870 as the training sample chosen at random from the entire training data with replacement. Then, each collection of subset data is used to train their decision trees. As a result, we ended up with an ensemble of different models. Majority of all the predictions from different trees are used which is more robust that a single decision tree.
In order to build a decision tree, we require discrete feature values. But, as the values we had for the bigram features were continuous, we transformed them to discrete values by binning.
We explored two ways of binning:
   (a) First method is, for each feature, identified minimum and maximum value that it takes and divided it into 10 equal partitions and each partition is assigned with a bin value corresponding to a number between 1 to 10
   (b) Second method is, we sort the complete data with respect to every feature and assign a bin value for a range of feature values when there is a change in the corresponding labels, i.e, binning was done adhering to clusters of labels.

   We experimented this model with different depths of decision trees and varying the number of decision trees in the ensemble.

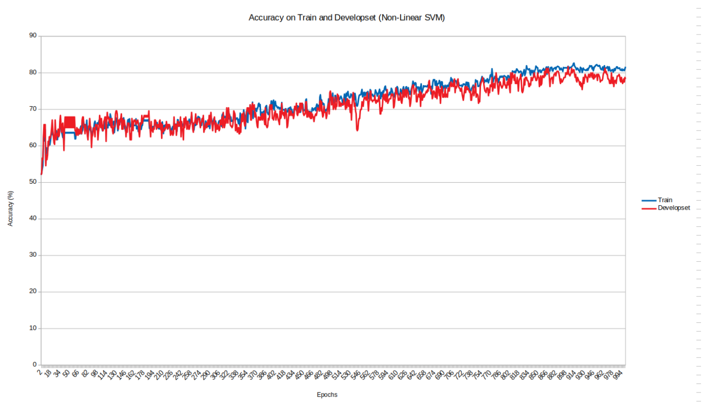

Figure 5: Accuracy Curves for Non-Linear SVM.

8. **Random Forest (Ho (1995))-**Random forest using decision tree is almost similar to bagging using decision tree except that while creating a group of classifiers, we pick bootstrap sample with replacement alongwith feature sampling for building a new classifier. This was also experimented with different number of decision trees in the ensemble, however, we don't limit depth .

Classifier	Best Hyperparamters	Accuracy on Development Set
KNN	$K = 50$	98.3%
Simple Neural Network	$\eta = 1$	77.08%
CNN	$activation = ReLU, dropout = 0.75$	77.92%
MLP	$activation = tanh, \alpha = 1$	72.5%
SVM	$\eta = 0.1, C = 1$	70.833%
Non-linear SVM	$kernel = RBF, C = 100$	78.33%
Bagging using Decision Trees	$depth = 30, numberoftrees = 5$	60.42%
Random Forest	$numberoftrees = 125$	64.5%
Bagging using Decision Trees(New Binning)	$depth = 15, numberoftrees = 7$	60.42%
Random Forest (New Binning)	$numberoftrees = 125$	55.80%

Table 1: Best values of hyperparamters as tested on Development set. For detailed results go to - https://goo.gl/ePyx2u. Note that here η means learning rate, α corresponds to the L2 regularization factor and C refers to the SVM regularization factor.

Results

As mentioned earlier, we analyze the performance of our models for different set of hyperparameters on a development set and choose the best set of hyperparamters for analysis on the test set. The best values of hyperparameters along with their respective accuracy on the development set is given in Table 1.

We were aware that for a fair assessment, we need to ensure that we hadn't polluted the test dataset by invariably looking at it. Hence, as a good rule of thumb, we have generated the test dataset a week before the submission and ran our trained model on this test set. A comparative study of accuracy of different classifiers with varying synthetic test datasets is given in table 2.

Development Set Results:

There are few interesting insights which can be drawn from the accuracy on development set. Firstly, the optimal k value for K-NN was strikingly close to the square root of the value of the training dataset. It also reported astonishingly high accuracy for the same dataset as compared to other models. Another interesting insight drawn from the table is the fact that the best development set accuracy for CNN was when the dropout probability on the final layer was 0.75, which was the maximum of value that we had considered which means that CNN seemed to overfit the data and hence dropout helped. The fact that SVM reported relatively much lower accuracy that other models seemed to indicate non-linearity in the data. Moreover, we also infer that the binning by clusters (the second type of binning) for Bagging on Decision Trees resulted in almost the same accuracy as compared to the other kind, however, with different hyperparamter values. However, the accuracy on new binning for Random Forest is observed higher than the old binning strategy.

Test Set Results:

We show evaluation of all models on convergence. We had previously reported that the scikit-learn's MLP converged much faster than our neural network which, in our opinion, is due to the fact that sklearn MLP has some built-in optimizers and regularizers which boosts the performance. We, moreover, compare the results of our model when it converges.

We see a pattern over different synthetic datasets that is almost consistent across all models which is that of BiLSTM data being the easiest and Word Shuffle and Drop being the most difficult to classify. This was expected since the sentences generated by BiLSTM lacked the naturalness of human language; the sentences were short and consisted of irregular POS bigrams like $DET \mid DET$ which are non existent in the real data. Word Shuffle and Drop dataset, on the other hand, was the closest to human language since it involved only a few number of swaps and drops to corrupt the sentences hence having preserved the majority of sentence structure.

The difference between the accuracy for the dataset generated by SRILM trained on WikiText and Harry Potter corpus is likely to be due to the similarity of sentences in the former with the real sentences. This makes us think that the classification models may not be domain-independent after all and language models like SRILM do capture the grammatical structure of the corpus alongside the context.

We observe, the tree based approaches perform miserably as compared to the neural network and SVM approaches. A possible explanation to these results may be poor choice of depth limitation which would have caused the data to underfit. The maximum depth taken was almost half the number of features. For Random Forests, we had considered 15 feature samples which may again have been a poor choice of hyperparameter.

To further reinforce the idea of ensembles, we observe that our Ensemble model performs the best classification on two of the texts and rates second best in the other two. This

Classifier	BiLSTM	SRILM WikiText2	SRILM HP	Word Shuffle and Drop
K-NN	97.5%	**95.0**%	86.66%	74.16%
MLP	97.3125%	76.4375%	88%	75.75%
Simple Neural Network	97.5625%	73.875 %	87.625%	71.75%
CNN	97.8125%	76.0625%	87.875%	75.5625%
SVM	97.125%	57.75%	81%	61%
Non-linear SVM	**97.9375%**	74.375%	88.1875%	71.875%
Ensemble (Non-linear SVM + MLP + CNN)	97.875%	76.6875%	**88.75**%	**77.25%**
Bagging using Decision Trees	81.25%	56.67%	57.083%	52.083%
Random Forest	83.75%	63.03%	69.67%	56.20%
Bagging using Decision Trees (New Binning)	63.749%	55%	60%	59.583%
Random Forest (New Binning)	74.08%	53.67%	70%	62.08%

Table 2: Comparitive study between accuracy of different models with different synthetic datasets.

is also due to the fact that the underlying classifiers (Non-linear SVM, CNN and MLP) also report good accuracies.

Discussion

There were a few interesting points which were discussed through the course of the project. Firstly, it is a well known fact that given a language in real world, the set of sentences complying to grammatical correctness will be a small subset of the set of all possible sentence. This means that the set of correct sentences would be very less in proportion to the set of incorrect(fake) sentences. The conflict for this project is whether to capture this skewness in our dataset or, as in this project, consider both real and synthetic sentences in equal proportion. Secondly, there was an inherent sense of doubt whether the POS tagger used would correctly tag words of synthetic sentences. From our analysis, we find that the POS tagger accurately tags synthetic sentences for most of the cases. Some examples are given in Appendix . However, we also feel that there is room for improvement in these experiments. Firstly, it would have been better to use Mahalanobis Distance to find K-nearest neighbors as it works best when data is distributed in ellipsoidal form, as compared to Euclidian Distance. Secondly, we took the depth and the number of trees as the hyperparameters for Random Forests and we believe that the number of features sampled could also be hyperparameter to optimize for better accuracy. We also believe that the BiLSTM generated data could have been better. This shortcoming was since, the bilstm module transforms the data into categorical vectors(binary vectors) and hence only a small amount of training data could be fit in memory. This may have led to improper training and hence poor generated results. This being said, we believe that we have applied many ideas that were taught in the class starting from the creation of validation set and its use for getting optimal hyperparamters, and generating final test files to report fair and accurate results. We applied ensemle methods - bagging and boosting on decision trees and many other models. We choose the appropriate number of epochs studying the accuracy curves and finally we applied classification methods which were taught in class and many which were not.

References

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157.

Ho, T. K. (1995). Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, pages 278–282. IEEE.

Jacobs, P. S. (1992). Joining statistics with nlp for text categorization. In *Proceedings of the third conference on Applied natural language processing*, pages 178–185. Association for Computational Linguistics.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Novikova, J., Dušek, O., Curry, A. C., and Rieser, V. (2017). Why we need new evaluation metrics for nlg. *arXiv preprint arXiv:1707.06875*.

Ranjan, V., Kwon, H., Balasubramanian, N., and Hoai, M. (2018). Fake sentence detection as a training task for sentence encoding. *arXiv preprint arXiv:1808.03840*.

Reiter, E. and Belz, A. (2009). An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

Shimorina, A. (2018). Human vs automatic metrics: on the importance of correlation design. *arXiv preprint arXiv:1805.11474*.

Stolcke, A. (2002). Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*.

Vapnik, V. (1999). Support vector method for function estimation. US Patent 5,950,146.

APPENDIX

Part of Speech Tags for Synthetic Sentences

Word Shuffle and drop-

SENTENCE: They then receive to fuel source alien an on how transmissions create endless from

TAGS: [u'PRON', u'ADV', u'VERB', u'PART', u'VERB', u'NOUN', u'ADJ', u'DET', u'ADP', u'ADV', u'NOUN', u'VERB', u'ADJ', u'ADP']

SENTENCE: Giger felt the film and the character were too similar to Alien so pushed for script changes

TAGS: [u'NOUN', u'VERB', u'DET', u'NOUN', u'CCONJ', u'DET', u'NOUN', u'VERB', u'ADV', u'ADJ', u'ADP', u'PROPN', u'ADV', u'VERB', u'ADP', u'NOUN', u'NOUN']

BiLSTM-

SENTENCE: the the 18 with the gods

TAGS: [u'DET', u'DET', u'NUM', u'ADP', u'DET', u'NOUN']

SENTENCE: the the own of the history in the not old the gods of but the a a the .

TAGS: [u'DET', u'DET', u'ADJ', u'ADP', u'DET', u'NOUN', u'ADP', u'DET', u'ADV', u'ADJ', u'DET', u'NOUN', u'ADP', u'CCONJ', u'DET', u'X', u'X', u'DET']

SRILM-

SENTENCE:said Professor McGonagall but last spot. could bear real on the at the as put out a boots.

TAGS: [u'VERB', u'PROPN', u'PROPN', u'CCONJ', u'ADJ', u'NOUN', u'VERB', u'VERB', u'ADV', u'ADP', u'DET', u'NOUN', u'DET', u'ADP', u'VERB', u'PART', u'DET', u'NOUN']

SENTENCE: It is through matures the release two

TAGS: [u'PRON', u'VERB', u'ADP', u'NOUN', u'DET', u'NOUN', u'NUM']