CS 229 Autumn 2018: End Mark Prediction

Eric Mark Martin
Stanford University, Computer Science 2021
ericmark@stanford.edu

Jonathan Zwiebel
Stanford University, Computer Science 2021
jzwiebel@stanford.edu

Abstract

Our project developed models for the task of assigning one of three end-marks - periods, question marks, or exclamation points - to variable-length English sentences. Our models were trained with a labeled data set extracted from English novels and scored using labeled examples from the same set. We tested logistic regression, naïve bayes, random forests, and SVMs, and found them all to be more effective than our baseline, proportional guessing, and less effective than our oracle, human-level. We found that random forests had the strongest performance of the developed models, although the similarities in performance make it impossible to conclude that they outperform the other models for this task.

1 Introduction

Practically all of our mobile devices use some form of autocorrect, predictive typing, or dictation to complete our sentences for us. Yet, if you open your phone and type a sentence your device will almost certainly punctuate it (if at all) with a period whether it's "Talk to you later.", "Come over!" or "You up?". We present results and analyses of our experimentation with several different methods for predicting the terminal punctuation of a sentence.

The terminal punctuation, or end mark, of a sentence provides important semantic and syntactical information about a sentence. In the question "What is that?", for instance, we don't think of "What" as some subject that is existing ("is") as some object "that". Without the knowledge that this is a question, however, it can be much more difficult for a computer to determine this information. The question of end marks also poses a simple problem: given a sentence, classify it as ending with either a period, question mark, or exclamation point. We limited our exploration to consider only English-language sentences and only these three end marks.

The goal of our project is to be able to correctly punctuate variable-length English sentences with one of three end marks: periods, question marks, or exclamation marks (denoted PERIOD, QMARK, EXPOINT in this poster). We want to punctuate sentences drawn from the distribution of English sentences so we did not re-weight our data to have equal proportions of each punctuation mark.

2 Related Work

Punctuation Prediction for Unsegmented Transcript Based on Word Vector by Xiaoyin Che, Cheng Wang, Haojin Yang, Christoph Meinel (4). This project used CNNs to classify and place punctuation (periods, commas, or question marks) within constant-length texts. We found that this paper did a great job of identifying punctuation in certain contexts but could not generalize to end-marks or short phrases.

Naïve Bayes for Text Classification with Unbalanced Classes by Eibe Frank and Remco R. Bouckaert (5). This project used multinomial Naïve Bayes for multiclass detection and showed that re-balancing classes leads to imrpoved ROC. We ultimately did not use this approach as it did not fit our problem formulation but strongly considered it.

Deep Learning for Punctuation Restoration in Medical Reports by Wael Salloum, Greg Finley, Erik Edwards, Mark Miller, and David Suendermann-Oeft (6). This project used bidirectional RNNs to restore punctuation on text that was generate from transcriptions. This paper was the most complex model that we read about related to our project. This project did a good job of providing clear metrics and establishing a point of comparison for a similar (albeit different given the kinds of punctuation and presence of context) problem.

It is clear that the state-of-the-art for a problem like this is an RNN. It is also clear from reading these papers that this is a difficult problem even for humans, given both the complexity and subjectivity of assigning punctuation. Almost all of the paper ran into the issue that text corpuses extracted from novels and online-sources are written by human authors who often deviate from standard punctuation rules. Two identical text segments may be punctuated differently when written by different authors. This in turn suggests that algorithms may do better when trained on a single source, which is promising for applications such as smart keyboards and speech-to-text engines.

3 Dataset and Features

Our goal was to be able to correctly punctuate variable-length English sentences with one of three end marks: periods, question marks, or exclamation marks (denoted PERIOD, QMARK, EXPOINT in this poster). We want to punctuate sentences drawn from the distribution of English sentences so we did not re-weight our data to have equal proportions of each punctuation mark.

We drew data from 10 of the top English-language books available at project Gutenberg, available for free use. We wanted to ensure that we could extract usable examples even from complex grammar structures such as dialogue and clauses. Additionally to maximize the number of question mark and exclamation point samples we needed to ensure that we could extract standalone sentences within quotations (eg: "'How are you?' said Frankenstein."). Our definition for a sentence was a sequence of space-separated words starting with a capital word, ending with an end mark, and unbroken by any single or double quotation marks. We wrote a parser to extract sentences from the texts according to this definition.

We then tokenized the dataset with tokens for each of the top 20,000 words from the Google Trillion Word Dataset, with five special-use tokens: <NUMBER>, <COMMA>, <SEMICOLON>, <PROPER>, <UNKNOWN> to handle important cases not counted in this dictionary. Commas and semicolons were made into their own words in the parsing stage so they could be captured by the tokenizer. We made this decision because of the valuable structural information these non-terminal punctuation provide about the sentence they are in.

We merged this tokenized data and then vectorized each tokenized sentence in a number of different ways, including binary vectors, bag-of-word vectors, and tfidf-transformed bag-of-word vectors.

4 Methods

Because this is a relatively unexplored problem, there isn't a well established baseline for a naïve algorithm. Random guessing would given an unfairly low benchmark as the vast majority of the data belongs to the period class.

We started by looking at the results of guessing all periods (Table 1). This was an important step to identifying which evaluation metrics were important to a "good" solution for this problem, as while this model is trivially useless, it still scores very well on micro-average precision, recall, and f1 scores due to the large proportion of period examples.

One metric it scored very poorly on, however, was macro-averaged even-weighted f1 score (macro f1). The macro average is helpful, as the fact that it evenly weights each class regardless of number of examples exacts a significant penalty on models that over-predict the period class. Using f1 score ensures that we are taking both precision and accuracy into account.

The 0.30 macro f1 from guessing all periods, doesn't serve as a good benchmark to evaluate our models against. To set a lower-bound macro f1, we used proportional guessing (Tables 2, 3), a model that would randomly guess an end mark using proportions taken from the training set.

Table 1: *All-periods classification metrics*

Class	Precision	Recall	F1-Score	Support
Period	0.82	1.00	0.90	1942
QMark	0.00	0.00	0.00	198
ExPoint	0.00	0.00	0.00	240
Micro	0.82	0.82	0.82	2370
Macro	0.27	0.33	0.30	2370

Table 2: *Proportion Guessing Confusion matrix*

	Period	QMark	ExPoint
Period	1568	140	190
QMark	174	16	8
ExPoint	192	19	29

Table 3: *Proportion Guessing Metrics*

Class	Precision	Recall	F1-Score	Support
Period	0.81	0.81	0.81	1932
QMark	0.08	0.08	0.08	198
ExPoint	0.13	0.12	0.12	240
Macro	0.34	0.34	0.34	2370

To set an upper bound, we used an oracle from human-level assessment (Tables 4, 5). Over random splits from a reduced test set, we asked humans to classify sentences the same tokenization scheme (but formatted to read as natural English) as given to the learned models.

Table 4: *Human-Level Confusion matrix*

	Period	QMark	ExPoint
Period	151	1	10
QMark	3	7	1
ExPoint	12	2	5

Table 5: *Human-Level Metrics*

Class	Precision	Recall	F1-Score	Support
Period	0.91	0.93	0.92	162
QMark	0.70	0.64	0.67	11
ExPoint	0.31	0.26	0.29	19
Macro	0.64	0.61	0.62	192

We used three fold cross validation on our aforementioned train-dev-test splits. Each model was trained on the training set, and measured on the development set. In each class, the model with best-in-class performance on the development set was run against the test set. The metrics you see reported in the tables throughout are the result of these runs against the test set.

5 Experiments and Results

We evaluated five different classes of models – logistic regression, naïve bayes, SVM, random forests, and fully connected neural networks – and compared their performance. Each model was evaluated over matching 90-5-5 train-dev-test splits and scored using standard classification metrics.

5.1 Logistic Regression

A standard multiclass logistic regression was run with 20005 features. We tested both binary and bag-of-words feature vectors and found binary feature vectors to be our strongest logistic regression model. See Tables 6 and 7.

Table 6: *Logistic Regression Confusion matrix*

	Period	QMark	ExPoint
Period	1874	26	32
QMark	116	72	10
ExPoint	166	13	61

Table 7: *Logistic Regression Metrics*

Class	Precision	Recall	F1-Score	Support
Period	0.87	0.97	0.92	1932
QMark	0.65	0.36	0.47	198
ExPoint	0.59	0.25	0.36	240
Macro	0.70	0.53	0.58	2370

5.2 Naïve Bayes

We trained two multinomial Naïve Bayes models—one trained on bag-of-words vectors and the other on term-frequency inverse-document-frequency (TFIDF) transformed bag-of-word vectors, one Bernoulli model on binary vectors, and one Gaussian model on bag-of-word vectors. Of the four, the Bernoulli model achieved the highest macro f1. This potentially indicates that word frequency without any data conveying their order is a noisier, or just worse, representation of sentence structure than just the presence of a word. See Tables 8 and 9.

Table 8: *Naïve Bayes Confusion matrix*

	Period	QMark	ExPoint
Period	1606	23	303
QMark	76	47	75
ExPoint	73	4	163

Table 9: *Naïve Bayes Metrics*

Class	Precision	Recall	F1-Score	Support
Period	0.92	0.83	0.87	1932
QMark	0.64	0.24	0.35	198
ExPoint	0.30	0.68	0.42	240
Macro	0.62	0.58	0.54	2370

5.3 Random Forests

We started by training a random forest with aggregation by averaging of 100 decision tree estimators using Gini loss on bag-of-words vectors with a max tree depth of 5. While this model trained very quickly, it predicted that everything was a period. We increased the max depth to 5, 10, and then 50 to try to get better classifications, but even with a depth of 50 the forest still only predicted periods. Finally, we tried allowing the constituent trees to branch until total leaf purity. This model took much longer to train, and while we worried that allowing this depth would result in a model with excessive variance. On the contrary, the bagging kept the model from overfitting, and it performed very well on macro f1 score. Overall, this random forest was the highest performing model statistical model we tested on the test set (Tables 10 and 11).

Table 10: *Random Forest Test Confusion Matrix*

	Period	QMark	ExPoint
Period	1900	10	22
QMark	124	67	7
ExPoint	170	8	62

Table 11: *Random Forest Test Metrics*

Class	Precision	Recall	F1-Score	Support
Period	0.87	0.98	0.92	1932
QMark	0.79	0.34	0.47	198
ExPoint	0.68	0.26	0.37	240
Macro	0.78	0.53	0.59	2370

Table 12: *Random Forest Dev Confusion Matrix*

	Period	QMark	ExPoint
Period	1909	11	22
QMark	133	56	9
ExPoint	170	11	62

Table 13: *Random Forest Dev Metrics*

Class	Precision	Recall	F1-Score	Support
Period	0.86	0.98	0.92	1942
QMark	0.72	0.28	0.41	198
ExPoint	0.62	0.22	0.32	230
Macro	0.73	0.49	0.55	2370

5.4 Support Vector Machines

Each of the support vector machines (SVMs) were trained on bag-of-words vectors. We started by training two non-linear kernel support vector classifiers (SVCs), a radial basis function (RBF) and a polynomial kernel. Both of these models took so long to train that we trained them on a random subsample of the train set, but they then predicted all periods. With more compute, we could have tweaked the hyper-parameters on these models, including outputting probabilities instead of hard classification, to achieve better metrics, but given their poor results and the time they took to train, we decided our time would be better spend experimenting with other models. We also tested a stochastic gradient descent (SGD) classifier with a linear kernel. We found that this last model performed best-in-class for SVMs (Tables 14, 15).

Table 14: *SVM SGD Test Confusion matrix*

	Period	QMark	ExPoint
Period	1903	17	12
QMark	128	69	1
ExPoint	182	16	42

Table 15: *SVM SGD Test Metrics*

Class	Precision	Recall	F1-Score	Support
Period	0.86	0.98	0.92	1932
QMark	0.68	0.35	0.46	198
ExPoint	0.76	0.17	0.28	240
Macro	0.77	0.50	0.55	2370

Table 16: *SVM SGD Dev Confusion matrix*

	Period	QMark	ExPoint
Period	1913	18	11
QMark	138	56	4
ExPoint	196	11	23

Table 17: *SVM SGD Dev Metrics*

Class	Precision	Recall	F1-Score	Support
Period	0.85	0.99	0.91	1942
QMark	0.66	0.28	0.40	198
ExPoint	0.61	0.10	0.18	230
Macro	0.71	0.46	0.49	2370

6 Analysis

Though the random forest classifier out-performed the other statistical models on the test set, it did so by only a few percentage points of macro-average F1-score. This is not significant enough for us to conclude that a random forest is the optimal model for this problem.

We found, not surprisingly, that all of our models had stronger performance on QMARK sentences than EXPOINT sentences. This matches our intuition, questions can be identified by the presence of question words (ex: who, when, will) while exclamatory sentences are more easily confused with declarative sentences. We see this in the confusion matrix for our oracle (Table 4) which demonstrates that even humans do better with QMARK sentences than EXPOINT sentences.

We also found that logistic regression and naïve bayes seemed to overpredict PERIOD while random forests and SVMs were more willing to predict QMARKs and EXPOINTs.

7 Conclusions and Future Work

All of our models outperformed our baseline but were not able to outperform our oracle, demonstrating that machine learning methods provide a reasonable solution to this problem, but can still be improved. Still, the poor absolute performance (precision of 70% on questions and exclamations) means that none of our models would be appropriate to include on a keyboard and dictation applications.

We strongly anticipate that incorporating knowledge about the length of the sequences, the order of the words, and the related context can greatly improve our performance. In particular, we believe that order can improve performance on questions which start clauses with common ’question words (who, is, when, how). Additionally sentences such as "is it now" (question) and "it is now" (declaration or exclamation) are vectorized identically by our models, so incorporating order will allow our feature vectors to more accurately represent our input phrases. We believe that sentence length and context can help with exclamations which are often shorter that PERIOD sentences and grouped together.

One way we could do this is by appending a series of binary vectors to our feature vector representing a one-hot of the first word, a one-hot of the last word, a binary vector of words that begin clauses, and a 150 binary vector where the nth element is 1 if there at least n tokens in our sentence. Another approach we would like to try is a true sequence-based model such as an RNN. To do this we would likely need to find a word embedding to reduce our feature space. To incorporate context we would also consider using a bidirectional-RNN.

Finally, we envision these models having applications on keyboard and dictation apps on mobile devices, so we would like to train our models over a more representative corpus of text.

Contributions

Eric worked on the Naïve Bayes, Random Forest, Naïve Bayes, and Oracle models. He also set up the development environment and preprocessing scripts. Jonathan worked on the Logistic Regression and Baseline models. He also worked on the vectorizer and tokenizer.

Acknowledgments

We would like to acknowledge Professor Ng and the CS 229 course staff for teaching us the material necessary to create these models and for looking over / providing feedback on our work. We would also like to acknowledge Project Gutenberg for providing us with a completely free-to-use corpus of text and Google for providing the free-to-use trillion word corpus.

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* 15(7):5249-5262.

[4] Che, X., Wang, C., Yang, H., Meinel, C. (2016, May). Punctuation Prediction for Unsegmented Transcript Based on Word Vector. In LREC.

[5] Frank, E., Bouckaert, R. R. (2006, September). Naive bayes for text classification with unbalanced classes. In European Conference on Principles of Data Mining and Knowledge Discovery (pp. 503-510). Springer, Berlin, Heidelberg.

[6] Salloum, W. Finley, G. Edwards, E. Miller, M. Suendermann-Oeft, D (2017, August). Deep Learning for Punctuation Restoration in Medical Reports (pp. 159-164). Association for Computer Linguistics.

Code and Data

Our code can be found at github.com/ericmarkmartin/cs229-autumn-2018-project.
The list of 20k words were taken from github.com/first20hours/google-10000-english.
The texts used in this project were taken from www.gutenberg.org/browse/scores/top.