How I handled imbalanced text data

Blueprint to tackle one of the most common problems in AI

Yogesh Kothiya
May 15, 2019 · 5 min read ★

Photo by Form on Unsplash

The problem of imbalanced class distribution is prevalent in the field of data science and ML engineers come across it frequently. I am a chatbot developer at IMImoble Pvt Ltd and faced this scenario recently while training intent classification module. Any live business chatbot accessible to real-world users is bound to attract a significant number of out-of-scope queries along with messages pertaining to the task it is designed to perform. Even among the relevant task-oriented messages, imbalances are to be expected as all topics covered by the bot can't be equally popular. For example, in a banking use case, balance inquiries will outnumber home loan applications.

Bot building is not similar to traditional application development. While the latter is relatively stable and is updated less often, the former needs frequent updates to improve user experience and intelligence of the bot. The imbalanced dataset is the problem where data belonging to one class is significantly higher or lower than that belonging to other classes. Most ML/DL classification algorithms aren't equipped to handle imbalanced classes and tend to get biased towards majority classes.

Why accuracy is a sham in the case of an imbalanced dataset

Aiming only for high accuracy for the imbalanced dataset can be counter-productive because standard classifier algorithms like Decision Trees and Logistic Regression do not have the ability to handle imbalanced classes incorporated into them. This leads to a heavy bias towards larger classes and classes with fewer data points are treated as noise and are often ignored. The result is a higher misclassification rate for minority classes compared to the majority classes. Therefore, the accuracy metric is not as relevant when evaluating the performance of a model trained on imbalanced data.

Consider the following case: you have two classes — A and B. Class A is 95% of your dataset and class B is the other 5%. You can reach an accuracy of 95% by simply predicting class A every time, but this provides a useless classifier for your intended use case. Instead, a properly calibrated method may achieve a lower accuracy but would have a substantially higher true positive rate (or recall), which is really the metric you should have been optimizing for.

This article explains several methods to handle imbalanced dataset but most of them don't work well for text data. In this article, I am sharing all the tricks and techniques I have used to balance my dataset along with the code which boosted f1-score by 30%.

Strategies for handling Imbalanced Datasets:

Can you gather more data?

You might think that this is not the solution you're looking for but gathering more meaningful and diverse data is always better than sampling original data or generating artificial data from existing data points.

Removing data redundancy:

1. Removing duplicate data — The dataset I was dealing with contained a lot of similar and even duplicate data points. "Where is my order" and "Where is the order" has the same semantic meaning. Removing such duplicate message will help you reduce the size of your majority class.

2. There were many messages which had the same semantic meaning, for example, consider the following messages, which convey the same meaning. Keeping one or two such utterances and removing others also helps in balancing classes. Well, you can use those messages in the validation set. There are many ways to find text similarity but I have used Jaccard Similarity because it is very easy to implement and it considers only unique sets of words while calculating similarity. You can look at other techniques in this article.

Can I change the delivery time for my delivery?

Can I change time of my delivery?

Can I change delivery time?

3. Merge minority classes — Sometimes multiple classes have overlapping features. It's better to merge those multiple minority classes. This trick helped me improve f1-score by more than 10%.

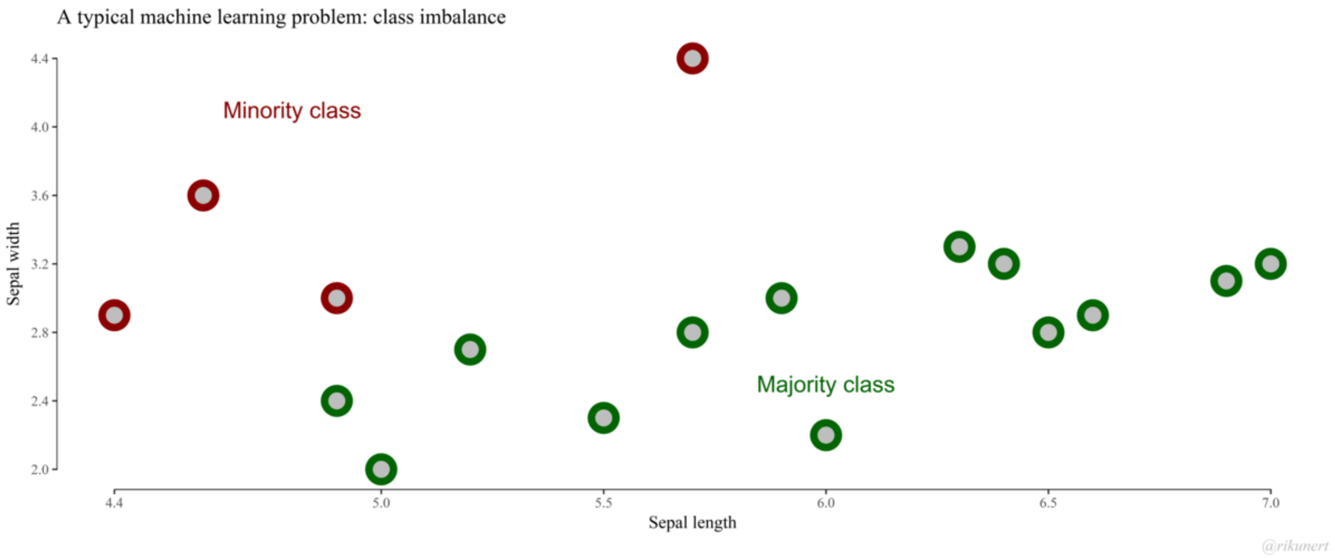
Resample training dataset:

The simplest way to fix imbalanced dataset is simply balancing them by oversampling instances of the minority class or undersampling instances of the majority class. Using advanced techniques like SMOTE(Synthetic Minority Over-sampling Technique) will help you create new synthetic instances from minority class.

1. Undersampling — An effort to eliminate data point from the majority class randomly until the classes are balanced. There is a likelihood of information loss which might lead to poor model training.

2. Oversampling — This is the process to replicate minority class instances randomly. This approach can overfit and lead to inaccurate predictions on test data.

3. SMOTE — SMOTE generates synthetic samples by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors as shown in below GIF. More importantly, this approach effectively forces the decision region of the minority class to become more general. Check this article for an easy explanation. Unfortunately, this technique doesn't work well with text data because the numerical vectors that are created from the text are very high dimensional.



SMOTE Illustration

If it's difficult to gather more data and above tricks do not show promising results then here is the last resort.

Data augmentation:

Data Augmentation is a technique commonly used in computer vision. In image dataset, It involves creating new images by transforming(rotate, translate, scale, add some noise) the ones in the data set. For text, data augmentation can be done by tokenizing document into a sentence, shuffling and rejoining them to generate new texts, or replacing adjectives, verbs etc by its a synonym to generate different text with the same meaning. Any pre-trained word embedding or NLTK's wordnet can be used to find the synonym of a word.

One of the interesting ideas used in Kaggle competition is converting English text to any random language and converting back to English using neural machine translation. This trick helped me to improve f1-score by 17%. Check this GitHub repo to find the code on Data augmentation using language translation, spacy, spacy_wordnet, and word embeddings.

Conclusion:

When dealing with imbalanced dataset there is no one-stop solution to improve evaluation metric. One may need to try several methods to figure out the best-suited technique for the text dataset. Let me know if you ever come across any such problem and how did you tackle it.

Do you want more? Follow me on Medium, LinkedIn, and GitHub.

kothiyayogesh/medium-article-code

Find the code of all my medium article.

Machine Learning Imbalanced Data NLP Smote Naturallanguageprocessing