Topic and Intent Classifier from Scratch

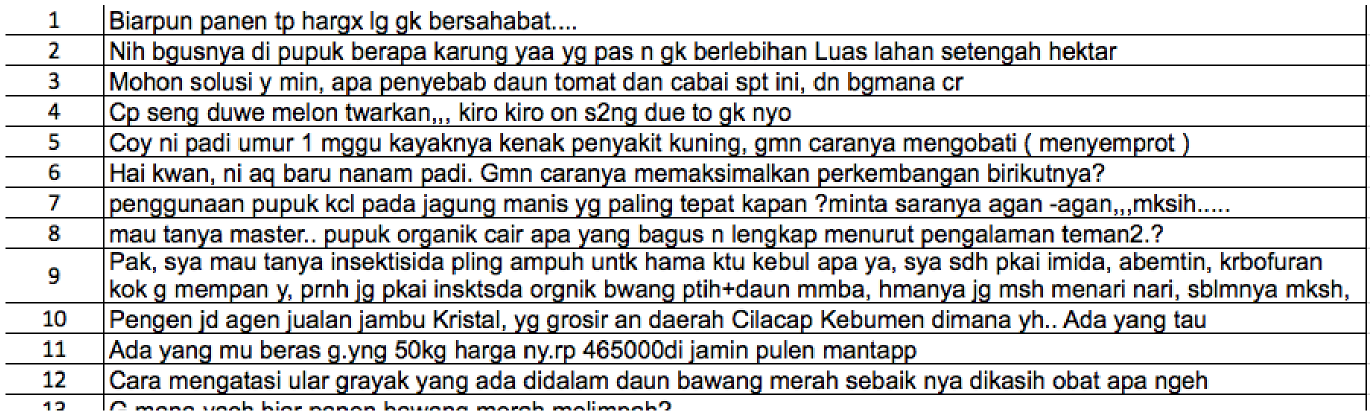
Himang Sharatun Follow
Feb 1, 2018 · 10 min read

Last month my team and I had a research project about machine learning for text analysis which includes sentiment analysis, topic classification, intent classification and named entity recognition. This research is more like a challenge for us due to our knowledge and experience regarding ML that close to zero, but we need to deliver MVP as soon as possible (3 weeks or so). Fortunately, even though it's not perfect, somehow we manage to build an API that fulfills 2 out of 4 features that I mentioned previously, which are topic classification and intent classification. In this article, I would like to share our experience on how someone new to machine learning can build fully functioning text classifier.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

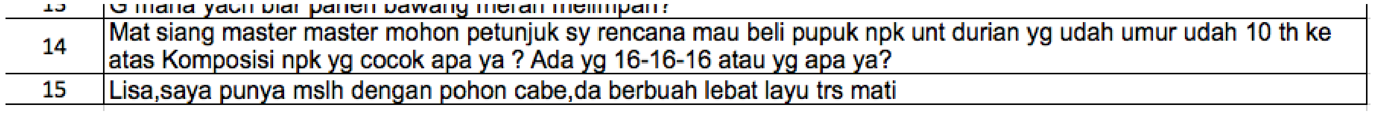
What to prioritize and why I hate our dataset?

The first thing that we do to start this project was deciding which feature is the easiest to work on. We understand that the required feature can be categorized into 2 problems, which are classification and named entity recognition. After 30 minutes discussion and google searching, we decide to work on classifier problem first because in need fewer data to train and the technique required to implement classifier is much simpler. Not to mention most named entity recognition example that we found was language dependent, which means that it can only recognize named entity if the input text is in certain language commonly spoken internationally such as English or Spain. This is so unfortunate because our input would be in Bahasa Indonesia and most of them in informal form. If you are wondering what our data looks like, here it is:



1	Biarpun panen tp hargx lg gk bersahabat....
2	Nih bgusnya di pupuk berapa karung yaa yg pas n gk berlebihan Luas lahan setengah hektar
3	Mohon solusi y min, apa penyebab daun tomat dan cabai spt ini, dn bgmana cr
4	Cp seng duwe melon twarkan,,, kiro kiro on s2ng due to gk nyo
5	Coy ni padi umur 1 mggu kayaknya kenak penyakit kuning, gmn caranya mengobati (menyemprot)
6	Hai kwan, ni aq baru nanam padi. Gmn caranya memaksimalkan perkembangan birikutnya?
7	penggunaan pupuk kcl pada jagung manis yg paling tepat kapan ?minta saranya agan -agan,,,mksih.....
8	mau tanya master.. pupuk organik cair apa yang bagus n lengkap menurut pengalaman teman2.?
9	Pak, sya mau tanya insektisida pling ampuh untk hama ktu kebul apa ya, sya sdh pkai imida, abemtin, krbofuran kok g mempan y, prnh jg pkai insktsda orgnik bwang ptih+daun mmba, hmanya jg msh menari nari, sblmnya mksh,
10	Pengen jd agen jualan jambu Kristal, yg grosir an daerah Cilacap Kebumen dimana yh.. Ada yang tau
11	Ada yang mu beras g.yng 50kg harga ny.rp 465000di jamin pulen mantapp
12	Cara mengatasi ular grayak yang ada didalam daun bawang merah sebaik nya dikasih obat apa ngeh
13	G mana yaah biar panen bawang merah melimpah?

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



13	G mana yach biar panen bawang merah melimpah?
14	Mat siang master master mohon petunjuk sy rencana mau beli pupuk npk unt durian yg udah umur udah 10 th ke atas Komposisi npk yg cocok apa ya ? Ada yg 16-16-16 atau yg apa ya?
15	Lisa,saya punya mslh dengan pohon cabe,da berbuah lebat layu trs mati

See how messy it is and I assume now you can understand our pain if we decide to work on named entity recognition first.

Since we got the data from an online agriculture forum, most of our data contain informal words, even some sentences contain no formal word at all. This became a problem because all named entity recognition technique that we found need full list of words that might appear in the input and it is impossible for us to list down all the informal word on the internet. Why?. Well, you know how unpredictable people can be when they try to shorten the words they type. For example the word “untuk”, most people will shorten it into “utk” or “u/”, but I found unusual ways to shorten it in the data such as “unt” or just “u”. Even if somehow we, magically, can formalize the data, it still going to be difficult for us to build NER for Bahasa Indonesia due to the difference in grammatical rules with the example we found (English and Spain). We need to formulate grammatical rules for Bahasa Indonesia, which going to takes a lot of time and the technique required is complex. Therefore, we decided to work on classifier first because it is language independent and preprocessing step is optional.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

What are our options?

After we decided to work on classifier first, we need to decide what approach we will use to build it, supervised or unsupervised ML. If we implement the supervised approach, we need to manually give labels to hundreds of data for training purpose which going to be tiring and boring, but if we implement the unsupervised one, there were several critical knowledge gaps that we can’t cover in just 3 weeks especially regarding the design of training process. Therefore, even though we need to manually give labels to our data, we chose to go with the supervised one. This picture below illustrate how text classifier using supervised ML works:

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

At this point, we have several machine learning algorithms that we could choose to implement supervised learning which is Naive Bayesian, LDA, SVM and Neural Network. But before we choose the algorithm, we need to find a method to translate words into an array (word embedding) since all algorithms that I mention previously need input in form of array or at least numbers. There are 2 options that we had to do that, by using one hot encoded bag of words (bow) or word2vec (CBOW). If we had more times, we definitely would choose word2vec to embed the input since the size of array would be significantly smaller compared to BOW, but we had limited time and to implement word2vec we need to use Java (deeplearn4j) or Python(gensim) which no one between us had any experience making an API using these languages. Actually, it is possible for us to create the classifier by using Python but the problem will occur in the process of making an API out of it, especially in the deployment process. To deploy Python in the live server, there are several configurations that need to be done and we don't have the courage to play around with our company server since everyone else is also using it for other projects. So for the sake of familiarity, we decide to use BOW which we manage to find a node package to implement it called mimir.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Why we choose neural network?

Some of you might think that we do a lot of simplification in this project because we want to take an easy way, but the truth is, we just want to explore more about ML technique to ensure optimize the MVP. We need to understand that there are a lot of ML techniques out there and each technique can be combined depending on resources and dataset provided. So, basically in this project, we spent most of our time by experimenting and evaluating any ML technique that we can think of under 3 criteria, which are:

1. Training Time
2. Convergence Error
3. Classifier Accuracy

For this trial and error process, we randomly picked 100 (80 training data + 20 test data) out of 2000+ data we provided and give 4 possible labels to each data to represent intent, which is question, advice, promotion, and others. There are 2 algorithms out of 4 options that we actually try and evaluate which are Naive Bayesian (using natural) and Neural Network

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

(BrainJS). Even though LDA has principal difference with Naive Bayesian (NB), but since both algorithms using the same approach, which is words probability, we conclude that the performance would be similar with NB with slight difference in accuracy. SVM is a unique technique that actually more suitable for small data like ours unfortunately, SVM is not suitable for multiclass classifier problem due to its nature that only capable to separate 2 classes (binary classification). Even though we found some intuition on how using SVM for multiclass classification, it requires us to train multiple numbers of SVM models depending on the number of class. For sentiment classifier with 3 classes (positive, neutral and negative) we need to train 2 SVM models, for intent classifier with 4 classes (question, advice, promotion, and others) we need to train 3 SVM models and for topic classifier with 8 classes, we need to train 7 SVM models. So instead of training 12 SVM models which going to takes times, we prioritize experiment by using Neural Network instead because we only need to train 3 NN models (1 for each problem). Therefore in our experiment, we prioritize experiment by using Naive Bayesian and Neural Network first over LDA and SVM.

After training and evaluation process finish, we found out that Bayesian takes less time (20 minutes) and resource to train compared with NN (30

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

minutes) but unfortunately in term of accuracy NN perform better with approximately 71% accuracy compared to NB with less than 60% accuracy. The errors most likely happen when the data should be classified as promotion, but both algorithms classify such data as question or advice. This happens because of imbalance label in our training data. In our training data, there is small number of data that manually classified as promotion while majority of them are classified into question and advice. To improve such condition, we need to increase more data that could be classified into promotion, but unfortunately even in the 2000+ provided data, there are small number of promotion data, so for now we decide to ignore this solution and conclude that neural network is the most suitable algorithm for our problem.

How we optimize the training process?

Theoretically, for bigger data, NN accuracy will improve more significantly compared to NB that will only slightly increase its accuracy. To prove that theory, we repeat the NN training process by using more data (250 training data and 50 test data). Unfortunately, somehow the training process takes unexpectedly 2 days to finish. At this moment there are 2 possible reasons

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

that the training process takes significantly more time than the previous one. The first possibility is that indeed, it is impossible to use JavaScript to train such a big data since JavaScript runs only in single thread CPU. Our last options would be to change our programming language to Python, but first, we try to find another alternative solution due to previously discussed problem if we use Python to create the API. Therefore, we decide to look for node package or library that enabling our training process to run on GPU which significantly faster than CPU. But, before we finish searching for GPU JavaScript library we decide to try to change our machine that runs the training process. We suspected that the problem is in the machine hardware spec since we run the training in core i5 machine. Therefore we try to run our training process on another machine with core i7 processor, and it works. The training process only takes less than an hour to finish and we discover that the accuracy of our model increase to around 75%.

An hour training process for 250 data is not that bad, but we have 2000+ data that potentially could be used in training process. Therefore, we need to find a solution to make the training process faster and one of them is by reducing BOW matrix. As we know, BOW will create a list of every word found in the training text. If in our training data, we have 10k unique words, then we will have 10k index in BOW matrix, which is going to take

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

times and resources to perform a calculation on such a huge matrix. Therefore, we tried to reduce the word count by using stop words removal and stemming (preprocessing). We decided to remove stop words such as “yang”, “di”, “dari”, “ke” etc because we believed that stop words can be found in many sentence thus make it less important for the classifier to include stop words in the classification process. We used nalapa to determine whether a word is a stop word or not and if it is indeed a stop word, we remove it from the sentence. After that, we remove prefix and suffix from the remaining words also by using nalapa , so words such as “mempunyai”, “dipunyai” and “punyaku” will only count as 1 word (“punya”) instead of 4. By doing so, hypothetically, we could reduce BOW matrix to speed the training process up without affecting the accuracy. This picture illustrate our new training process.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Unfortunately, our hypothesis was wrong. Yes, the training process became faster but not only it is not significant (40 minutes) the accuracy drops to only 45%. We can't pinpoint the exact cause of this behavior, but we suspect that the stop word removal in nalapa also remove words that potentially become discriminant between classes and/or the stemming process couldn't handle informal Bahasa Indonesia. It is impossible for us to manually exempt potentially discriminant words since we don't know which words are important for classifier and which one is not, so we couldn't prove our first hypothesis through experiment. For the second hypothesis, we already tried to use another node package to stem the sentences (akarata), but the accuracy drops further to 43%. It came into our consideration to create our own stemming library, but to do that we going to need more times and there is no certainty that it will improve the accuracy. So, we decide to stick to the previous technique which didn't involve any preprocessing in the training process.

The fruit of our labor

In the last week of research, we focus on training NN models using as much as possible data and creating the API. As expected, when we train the NN

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

model using 1800 data, it takes more time than previous experiment even though we run the training process on the server. Unfortunately, when the training process finish, we failed to save the training model. This happens because there is limitation on number of String that JavaScript capable to process. Because the deadline is getting closer, we decide to use the previous model, the one that we train using 250 data. Even though the accuracy is only 75%, it is still considered as acceptable model, not to mentions we had no other model that could predict more accurate than this model. After we manage to properly deploy the intent classifier, we continue to train another model for topic classifier using 300 data. We couldn’t train the sentiment classifier yet because the data that we provided only contain small number of data that could be classified as positive and negative, too much neutral data. That’s why, our API only capable to classify intent and topic from the sentence for now. Here is the example of our API response:

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

While this picture explain the classification process for intent classification that we use in the MVP

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

What we could improve?

From the MVP that we create on this project, there are several things that could be done to improve it. First, change the word embedding technique to technique that capable to represent word in smaller array such as word2vec, fasttext or GloVe. This is not only to speed up the training process but to capture the relationship between words in order to increase classification accuracy. Second, change the programming language to Python, Java or R, since JavaScript is not suitable for heavy computational operation. Third, explore about SVM and clustering or unsupervised learning.

That's the end of our story for this project, we hope that you can learn something new from our experience. If you have further question, don't hesitate to ask me at himang@skyshi.io or you can come to our office at PT Skyshi Digital Indonesia. See you on the next article.

Machine Learning JavaScript

261 claps

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

WRITTEN BY

Himang Sharatun

Follow

from "hello world" to change the world

SkyshiDigital

Follow

We help scale up your startup to the next level

See responses (5)

More From Medium

Continuous control with A2C and Gaussian Policies — MuJoCo , PyTorch and C++

Vittorio la Barbera

Facial Recognition: You Cannot Hide From The Machine

Prof Bill Buchanan OBE

How to Use Machine Learning in Five Minutes

Econsult Solutions in Econsult Solutions

Machine Learning Operations (MLOps) Pipeline using Google Cloud Composer

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Engineering@ZenOfAI in ZenOf.AI

Facial Recognition — a visual step by step

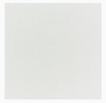
Patrick Ryan in The Startup

Attention Please: Document Classification

Jon-Ross Presta in Towards AI — Multidisciplinary Science Journal

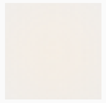
Introduction to Reinforcement Learning

Code Heroku in Code Heroku



TensorFlow Lite Now Faster with Mobile GPUs (Developer Preview)

TensorFlow in TensorFlow



Discover Medium

Welcome to a place where words matter. On Medium, smart voices and original ideas take center stage - with no ads in sight. Watch

Make Medium yours

Follow all the topics you care about, and we'll deliver the best stories for you to your homepage and inbox. Explore

Become a member

Get unlimited access to the best stories on Medium — and support writers while you're at it. Just $5/month. Upgrade

Medium

About Help Legal

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD