Generalizability of LDA-RF Language Model for Technical Texts

FNU Vivek
CSC 695 MS Thesis Research, Spring'19,
Computer Science, North Carolina State University
Raleigh, NC
vvivek@ncsu.edu

1 DATASETS FOR VERIFICATION

In the prior versions of this project we have established a language model that can be applied on any github project successfully for commit message classification. The language model is highly localized and can be built on even small datasets. This model is built as a binary classifier which has distinctly two parts:

- Topic model - An LDA based topic model built on the text corpus.
- Classifier - Top N words in the K topics along with the K topic probabilities. These N + K features are fed to a Random Forest Model. (we have K = 2 i.e. 2 topics corresponding to the two classes, 0 and 1).

To establish generality and robustness of our model we evaluate the performance of the model on two different datasets. The datasets have been generously provided by Real-world Artifical Intelligence for Software Engineering at department of Computer Science at NC State. First dataset is the built by extracting data from StackOverflow and second one pertains to technical debt i.e. code comments indicating tech debts in public open source projects.

1.1 Stack Overflow dataset

StackOverflow dataset is collected from stackoverflow.com. The data contains questions and tags associated with them. We specifically focus on 10 classes as shown in Table 1 created by analyzing associated tags. This table shows the amount of data (number of stackoverflow questions) we have in each category. While using this data we will mark one class as 1 and rest as 0 and sample desired amount of rows from this modified dataset.

Table 1: StackOverflow.com dataset

project	Counts
academia	8081
cs	9270
diy	15287
expressionengine	9469
judaism	14393
photo	12733
rpg	11208
scifi	19546
ux	13424
webmasters	17911
total	131322

1.2 Technical Debt Dataset

Technical Debt dataset is extracted from comments in the code of 10 open source projects as shown in Table 2. Each of the comments have been associated with one of the classes namely: defect, design, documentation, implementation, test or without any classification. For the purposes of testing the generality of our model, we classify each of the technical details as 1 and the rows without classification as 0.

2 EVALUATION

Our LDA based model has claims to create a generic and local language model for a specific project. This translated into the fact that our model when modeled as a binary classifier, how efficient is it to capture the features of the natural language. We used the same metrics we have used to evaluate our language model on github commits dataset i.e F1 and Recall. However, we also report accuracy, precision and F2 score here. These are only for references and verifying correctness of the model.

2.1 StackOverflow

We have 10 classes in this dataset as shown in Table 1. For creating the dataset we follow following steps for one class at a time:

- Label all data for a class as 1 and others as 0, and shuffle the data.
- Split the data randomly in 80:20 for train and test, respectively.
- From the training data create a random dataset of 2:1 label counts i.e. one-third of data should be 1s and two-thirds as 0s and drop other data points. Shuffle the training data.
- Split training data in training and validation 80:20.
- Train on training data, validate on validation.
- Test on testing data and record metrics
- Repeat above steps 20 times and report average metrics.

Recall and F1 for StackOverflow dataset is as shown in Fig. 1. We see that on average we have nearly 89% recall and 79.2% F1 score. We find that average precision is less that brings down the F1 score. Depending on the use case we can assign relative importance to recall or not. Therefore, we have included precision and F2 score in Table 1. In particular, it can be seen that this model consistently has better recall than precision which was evident from github commits dataset too. We can see from the Table 4 that the classes with high recall and F1 produce much coherent topics (Topic 1 corresponds to topic related to the category). This model is generic enough to learn features of natural language in texts like stackoverflow, which is loosely SE based texts.

Table 2: Technical Debt Dataset

project	DEFECT	DESIGN	DOCUMENTATION	IMPLEMENTATION	TEST	WITHOUT_CLASSIFICATION
apache-ant-1.7.0	13	95	0	13	10	3967
apache-jmeter-2.10	22	316	3	21	12	7683
argouml	127	801	30	411	44	8039
columba-1.4-src	13	126	16	43	6	6264
emf-2.4.1	8	78	16	2	0	4286
hibernate-distribution-3.3.2.GA	52	355	1	64	0	2496
jEdit-4.2	43	196	0	14	3	10066
jfreechart-1.0.19	9	184	0	15	1	4199
jruby-1.4.0	161	343	2	110	6	4275
sql12	24	209	2	50	1	6929

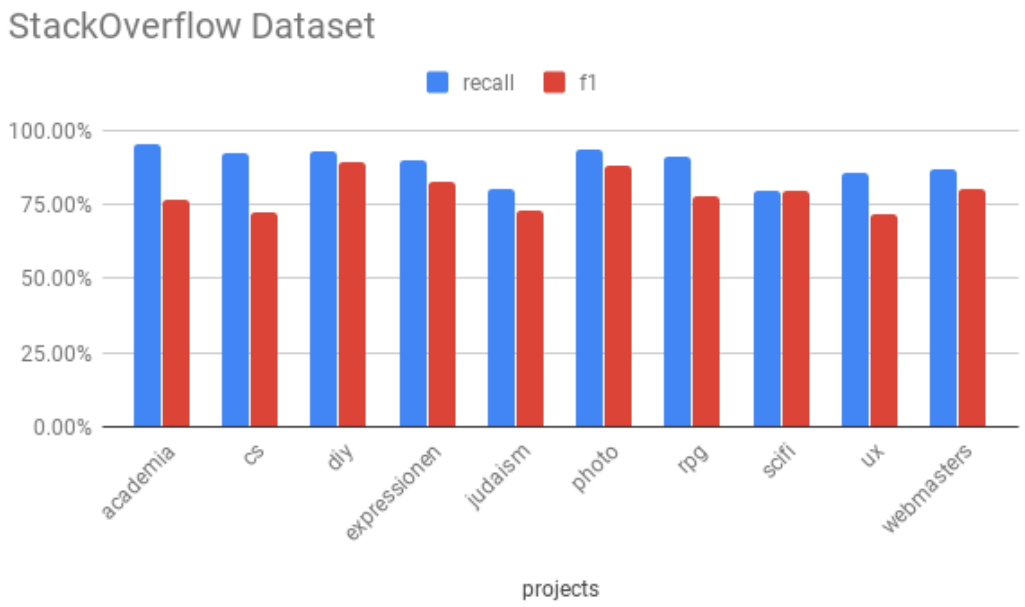

Figure 1: Performance on Stack Overflow dataset

2.2 Tech Debt

Tech Debt data is merely comments from the code-base of the project. We use the same metrics as above i.e. F1 and recall to gauge the performance of our model. The evaluation graph is as shown in Fig. 2. We see that recall is consistently high for all the projects except for 4 classes. F1 score is greater than 50% for 7 classes out of 10. Table 6 shows the words associated each of the topic. In 'jEdit-4.2' project we see a lot of overlap between technical terms. We can attribute this fact for this project to have one of the lower recall and F1 scores. This fact is also evident from the 'emf-2.4.1' and 'apache-ant-1.7.0' project. Also, it is worthy to note that the average recall is 63.5% and F1 is 66.2%. Average recall as established in the baseline model Huang et al [1] is 77.33% and average F1 is 73.7%. Our model does not outperform Huang et al on average. However, in 2 of the projects namely 'columba-1.4-src' and 'jruby-1.4.0' performs better than the baseline model. Although, both our approach and Huang et al approach is a generic approach that generalizes to any project. Huang et al use a Naive Bayes classifier on top of words transformed to a Vector Space Model. Our model uses a probabilistic framework by using LDA built on top of TF-IDF vector and the features from LDA are used to train a Random Forest classifier.

3 CONCLUSION

Our model is a generic model that generalizes on a variety of datasets. We see that our model which is an LDA-based classifier works with github commit messages for change message classification, also with stackoverflow dataset for topic classification and tech debt for comments classification. Though, we should note to achieve this performance on StackOverflow and tech debt dataset, we had to tune some hyperparameters associated with our model.

Table 3: StackOverflow Dataset Metrics

project	accuracy	precision	recall	f1	f2
academia	96.35%	64.30%	95.44%	76.71%	86.89%
cs	94.95%	59.38%	92.81%	72.27%	83.26%
diy	97.40%	85.56%	93.41%	89.31%	91.72%
expressionengine	97.27%	76.64%	89.88%	82.72%	86.87%
judaism	93.42%	67.01%	80.24%	73.01%	77.17%
photo	97.54%	83.32%	93.77%	88.23%	91.47%
rpg	95.53%	67.49%	91.62%	77.72%	85.50%
scifi	93.85%	79.36%	79.96%	79.65%	79.84%
ux	93.11%	61.60%	85.65%	71.65%	79.44%
webmasters	94.24%	74.92%	87.00%	80.50%	84.28%

Table 4: StackOverflow Topics - Top words

project	topic 0	topic 1
academia	use, page, site, like, user, would, need, way, set, one, websit, googl, look, imag	paper, research, one, student, would, work, univers, year, phd, question, time, know, ask, get, book
cs	use, page, site, user, like, would, com, work, imag, want, get, look, googl, websit, name	problem, one, time, would, algorithm, languag, number, question, find, know, say
diy	use, one, would, like, page, know, site, user, time, question, say, name, exampl, com, differ	water, wall, light, wire, hous, use, would, need, floor, instal, one, like, switch, replac
expressionengine	one, would, use, like, know, time, question, make, say, think, look, could	entri, page, site, field, channel, user, use, form, exp, file, categori, name
judaism	use, page, site, user, like, want, work, would, set, get, com, look, need	one, say, would, know, sourc, time, question, peopl, day, answer, make, torah, read
photo	page, use, site, user, name, com, file, one, would, websit, like, work, exampl, googl, question	camera, len, use, photo, would, one, like, get, light, take, canon, look, imag, pictur
rpg	use, page, site, user, name, would, like, work, imag, websit, com, googl, one, know, problem	would, charact, one, use, level, player, like, attack, spell, game, make, get, rule, power, book
scifi	use, page, user, site, would, like, need, work, want, problem, get, com, way, imag, websit	one, book, would, time, know, read, say, like, year, seem, stori, rememb, peopl, question, power, charact
ux	one, would, use, know, time, say, like, get, question, make, seem, read, take, look	user, page, use, site, design, button, websit, like, search, form, list, com, field, exampl
webmasters	one, would, use, time, like, know, question, say, make, get, seem, look, could, take, think	site, page, com, googl, websit, user, use, domain, link, http, search, file, content, imag, url

We have kept number of features as 1000 and top N words from topics to be 100. In change message classification model, these values were 100 and 25 respectively. We find that a small amount of tuning is required to created project specific model. Another point to note is that, we used TFIDFVectorizer in these experiments, however we have noted that CountVectorizer gives similar performance. TFIDF tends to give more coherent and distinct topics. Our earlier analysis of using CountVectorizer to be the choice needs a thorough analysis.

Since, the performance in each of these datasets our model performs well within the range of the performance of baseline models and also outperforming in certain cases. It can be established that LDA-RF i.e. LDA based Random Forest classifier is very generic and robust to datasets. The property of this model to generalize and localize to a project is effective in extracting useful features from natural language SE texts. The projects that generally have lower F1 and recall scores have more correlation between the two topics. For our case, as a binary classifier, we have kept number of topics to be 2. It is our conscious decision that the two topics should correspond to the two classes in order to learn features of different classes. However, in case of high correlation between the topics we need to figure out a way to incorporate more feature extraction technique. We can either let LDA learn for a larger number of iteration rather than 20 that is default for our model. We could also explore creating more topics and understanding topic perplexity to incorporate more granular features.

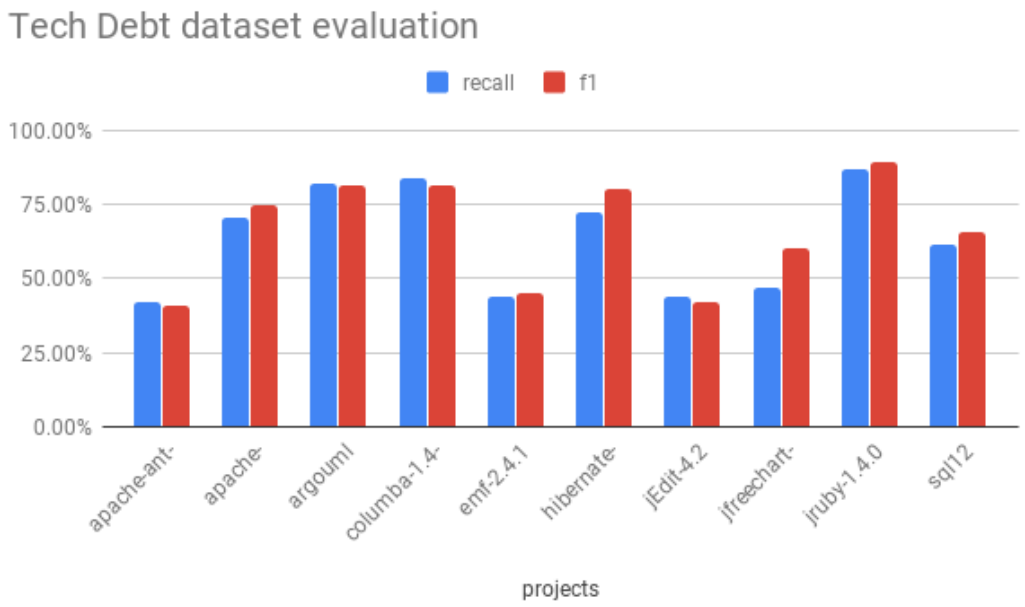

Figure 2: Performance on Technical Debt datasets

Table 5: Technical Debt Dataset Metrics

project	accuracy	precision	recall	f1	f2
apache-ant-1.7.0	96.14%	42.34%	42.29%	41.01%	41.42%
apache-jmeter-2.10	97.80%	79.65%	70.88%	74.72%	72.31%
argouml	94.51%	81.53%	82.48%	81.69%	82.10%
columba-1.4-src	98.71%	80.30%	84.26%	81.57%	82.93%
emf-2.4.1	97.41%	47.79%	43.87%	44.97%	44.14%
hibernate-distribution-3.3.2.GA	94.34%	90.81%	72.26%	80.40%	75.30%
jEdit-4.2	97.02%	41.33%	43.79%	42.22%	43.07%
jfreechart-1.0.19	97.11%	87.41%	46.83%	60.47%	51.42%
jruby-1.4.0	97.38%	91.74%	86.95%	89.26%	87.85%
sql12	97.44%	71.65%	61.27%	65.71%	62.91%

REFERENCES

[1] Qiao Huang, Emad Shihab, Xin Xia, David Lo, and Shanping Li. 2018. Identifying Self-admitted Technical Debt in Open Source Projects Using Text Mining. *Empirical Softw. Engg.* 23, 1 (Feb. 2018), 418–451. https://doi.org/10.1007/s10664-017-9522-4

Table 6: Technical Debt Topic - Top words

project	topic 0	topic 1
apache-ant-1.7.0	file, ignore, attribute, property, set, directory, specified, create, need, name	checkstyle, ignore, visibilitymodifier, class, default, method, end, get, test
apache-jmeter-2.10	todo, used, set, test, file, check, default, use, name	nls, non nls, noop, panel, main, org, apache, see, apache jmeter, main panel
argouml	todo, class, end, set, needed, name, element, model, add, nothing	see, org, argouml, uml, object, java, lang, lang object,
columba-1.4-src	nls, non nls, message, folder, add, todo, create, get, new, author	columba, text, org, non javadoc, javadoc see, event, update, tooltip,
emf-2.4.1	value, byte, fill, object, non nls, nls, copied, non, interface, javadoc copied, javadoc	ignore, create, feature, nothing, command, class, add, file, new, content
hibernate-distribution-3.3.2.GA	todo, cache, check, add, column, alias, new, sql, object, use, one, node, key, note, method	collection, type, return, null, element, name, join, support, property, table, first, handle, select, array
jEdit-4.2	constructor, variable, type, line, case, instance, value, buffer, need, check, object, note, instance variable	method, class, member, private, private member, create, init, init method, find, package, workaround, inner class, inner
jfreechart-1.0.19	draw, argument, defer, defer argument, checking, argument checking, test, nothing, todo, paint	check, series, add, item, data, point, value, case, try, plot, independence, set, null, axis, fixme, check independence
jruby-1.4.0	line, fixme, switch, set, thread, constant, new, add, receiver, module, ignore, true, create, error, node, exception	method, value, args, class, block, scope, ruby, string, mri, array, name, variable, case, java, call
sql12	sql, error, todo, see, null, session, non, class, row, user, squirrel, oracle, nothing, javadoc, test, non javadoc	table, file, column, data, name, type, use, new, object, set, need, method, value, add, create, graph, text, expected, read