Duplicate Bug Report Detection with a combination of Text Retrieval and Topic Modeling

Anh Tuan Nguyen, Tung Thanh Nguyen, Tien N. Nguyen, David Lo, Chengnian Sun

Presented by Eduardo Jaime

The Problem

Duplicate bug report detection.

Many people are interacting with the system. Devs, testers, final users…

- Common issue in large scale systems: Eclipse, Mozilla, Open source tools…
- But, not so common in small systems: My own calendar app…

Everyone has different backgrounds and interactions with it.

- GUI designers, Developers, Testers, Accountants…

The Problem

Different reports may refer to the same issue while using different technical definitions.

Remember, people use different terminologies or styles, or even describe the same issue using different phenomena:

Execution traces, outputs, screenshots (happened to me a lot)...

A bug is reported and recorded in an issue-tracking database.

Jira, Bugzilla, etc...

Why is detecting duplicates important?

By detecting duplicates we can:

- Save time and effort in triaging bugs.
- Reduce maintenance efforts. Developers are not assigned to same issue.
- Get more information related to the same issue from different points of views.

Automating this process is also critical and many approaches have been proposed:

- IR-based, strict but do not detect topic similarities.
- Topic Modeling, less strict but detect topic similarities.
- NLP, REP, etc...

What did the authors propose?

The authors propose an approach named DBTM, that combines two models:

An IR Based one called BM25F:

  Measures textual similarity.

A Topic Based one called T-Model:

  Their own novelty model.

  Addresses topic similarity between duplicate reports.

The basics of DBTM

A report is considered a textual document describing one or more technical issues/topics in a system.

Duplicate bug reports describe the shared technical issue/topic in addition to their own topics with different phenomena.

The authors apply Ensemble Averaging technique to combine the IR and Topic based methods.

DBTM is trained on historical data with identified duplicate reports.

Topic Modeling with LDA

A system is considered to have K technical aspects/functionality.

Each is considered a topic, represented via certain words/terms.

- File sharing, Versioning, Printing, Report generation....
- Printing related terms could be printer, send, page size.

The bug database is a collection of bug reports.

Each bug report is considered a textual document containing a number of words/terms to report on one or multiple technical issues.

Vocabulary, Topic and Word Selection

The words in all bug reports under consideration are collected into a common vocabulary (Voc) of size V.

To describe a topic, one might use words drawn from that Vocabulary.

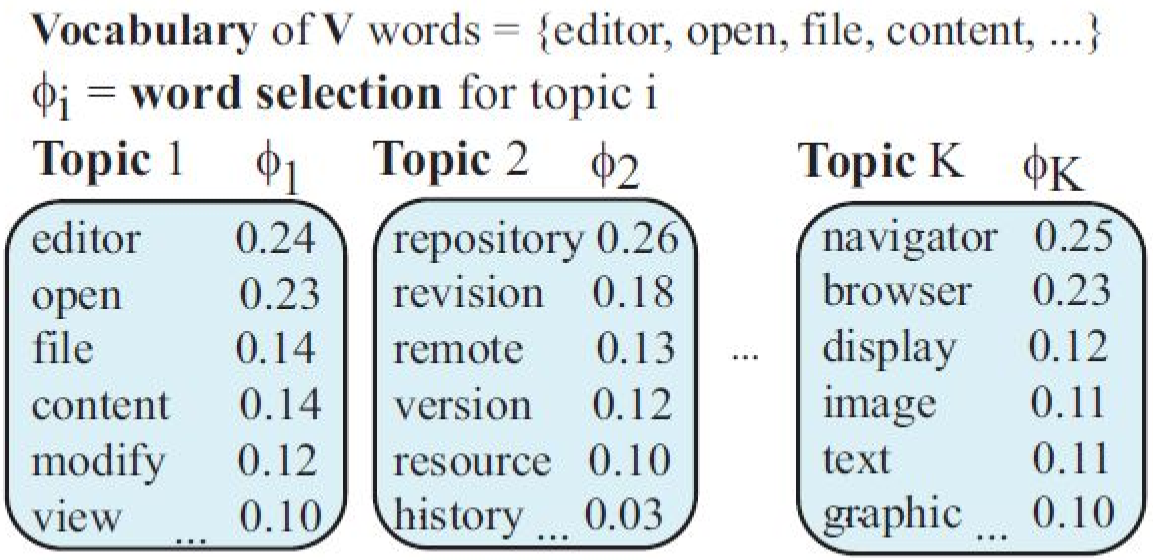
Each word in Voc has different usage frequency in describing a topic k.

A topic k can be described via one or multiple words. Thus, represented as a set of words with their probabilities. (word-selection vectors ϕk)

Each vector is meaningful for the entire collection of reports.

Representation of a Topic k

All the vectors for K topics from the per-topic word distribution matrix: (ϕ)



Vocabulary of V words = {editor, open, file, content, ...}

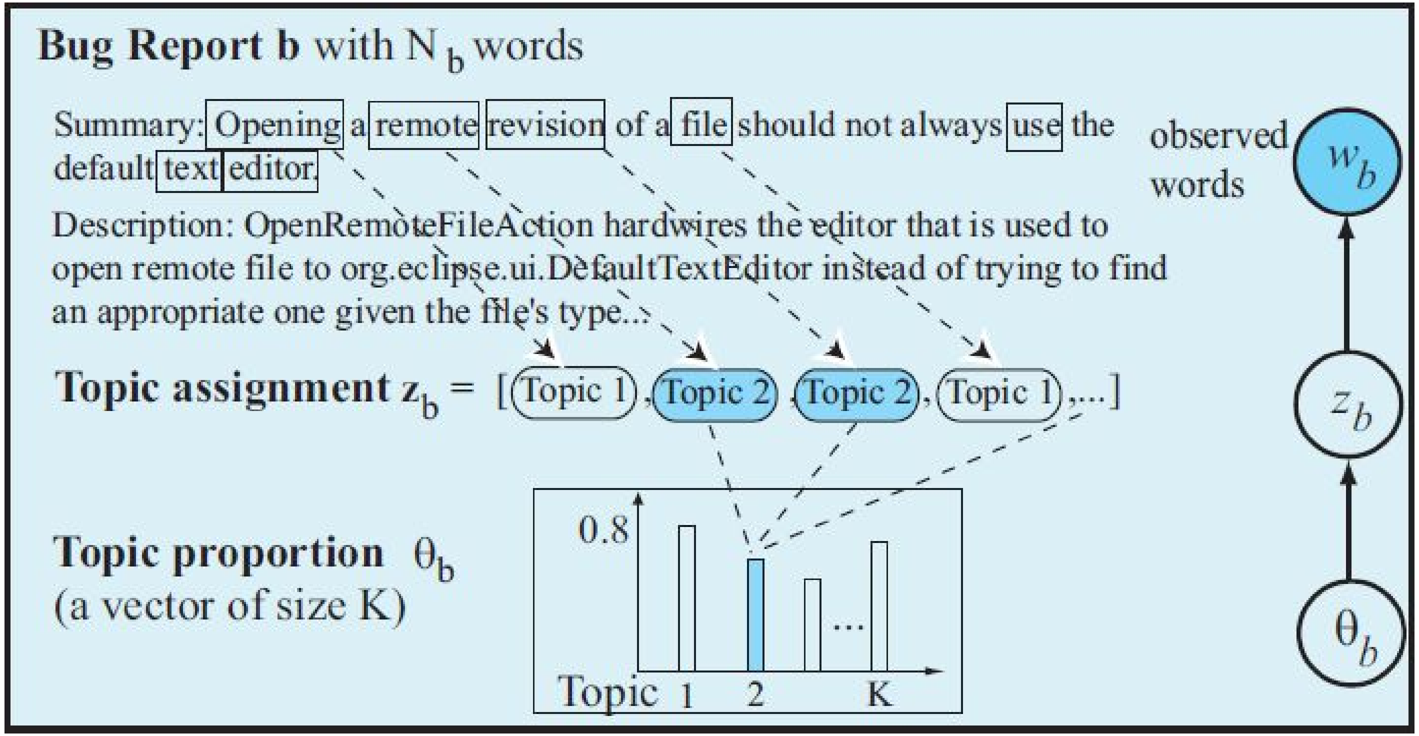
ϕ_i = **word selection** for topic i

Topic 1	ϕ_1	**Topic** 2	ϕ_2		**Topic** K	ϕ_K
editor	0.24	repository	0.26		navigator	0.25
open	0.23	revision	0.18		browser	0.23
file	0.14	remote	0.13	...	display	0.12
content	0.14	version	0.12		image	0.11
modify	0.12	resource	0.10		text	0.11
view ...	0.10	history ...	0.03		graphic ...	0.10

Figure 3: Topic and Word Selection [6]

Bug Report Modeling

Text from description and summary fields in a report are extracted:


Bug Report b with N_b words
Summary: Opening a remote revision of a file should not always use the default text editor.
Description: OpenRemoteFileAction hardwires the editor that is used to open remote file to org.eclipse.ui.DefaultTextEditor instead of trying to find an appropriate one given the file's type...
observed words
w_b
Topic assignment z_b = [Topic 1, Topic 2, Topic 2, Topic 1,...]
z_b
Topic proportion θ_b
(a vector of size K)
0.8
...
Topic 1 2 K
θ_b

Bug Report Modeling

Document b contains Nb words, each document has two parameters associated:

Topic Assignment Vector zb. Each Nb position in b describes one topic. zb for b has the length of Nb.

Topic Proportion θb. b can describe multiple topics. LDA associates a topic proportion to each document b to represent the significance of all topics in b.

This is a vector with K elements, with a value [0-1]. As an example:

if θb = [0.20, 0.24, 0.12, ...], 20% of all words in b are about file editing, 24% are about file versioning, and so on totalling 100%.

T-Model for Duplicate Bug Reports

Novel topic model developed by the authors.

Each bug report b is modeled by a LDA, represented with 3 variables:

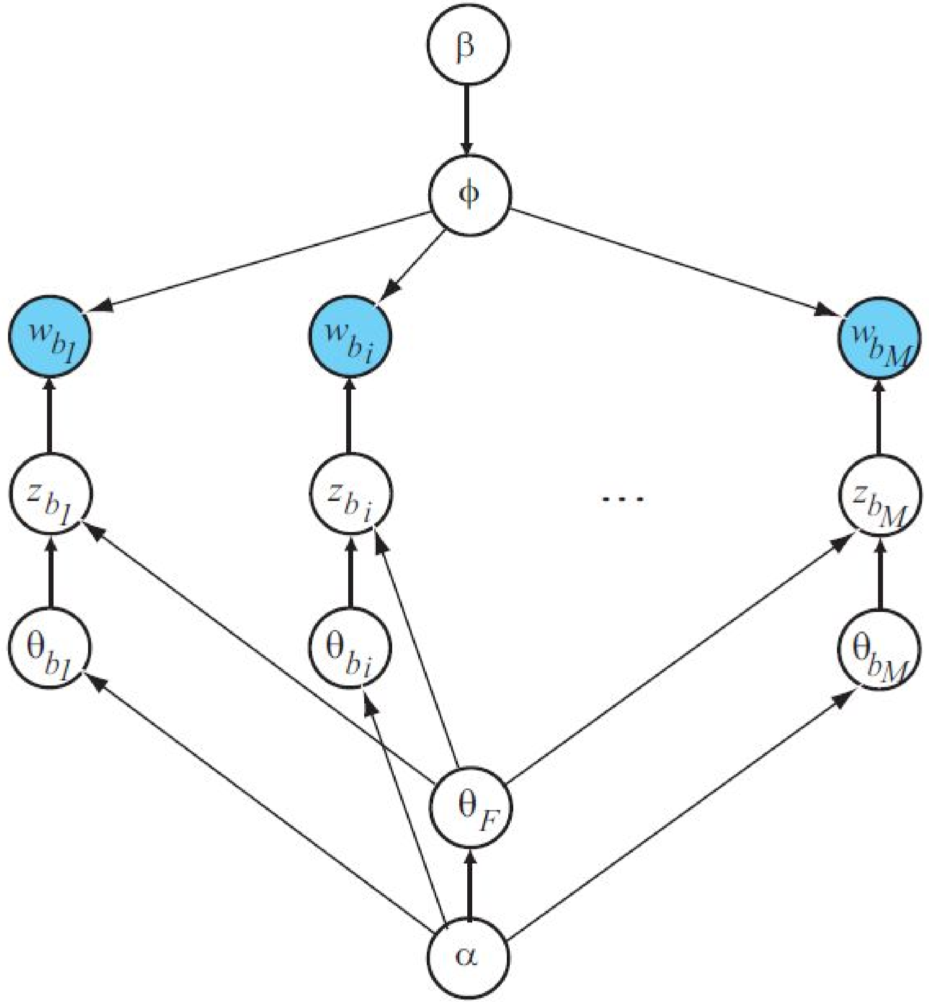
Topic proportion θb, topic assignment (zbi), and word selection (ϕ).

zbi is generated based on θb, then a word wb is generated based on topic assignment (zbi), and word selection (ϕ) for that corresponding topic.

Shared technical issues F are considered as topics and its topic proportion/distribution denoted by θF.

T-Model Example

- b1,..., bM denote M duplicate bug reports for the shared technical issue F.
- Those M reports describe that technical topic as well as their own local topics.
- Local topics for each report bi are modeled by the topic proportion θbi

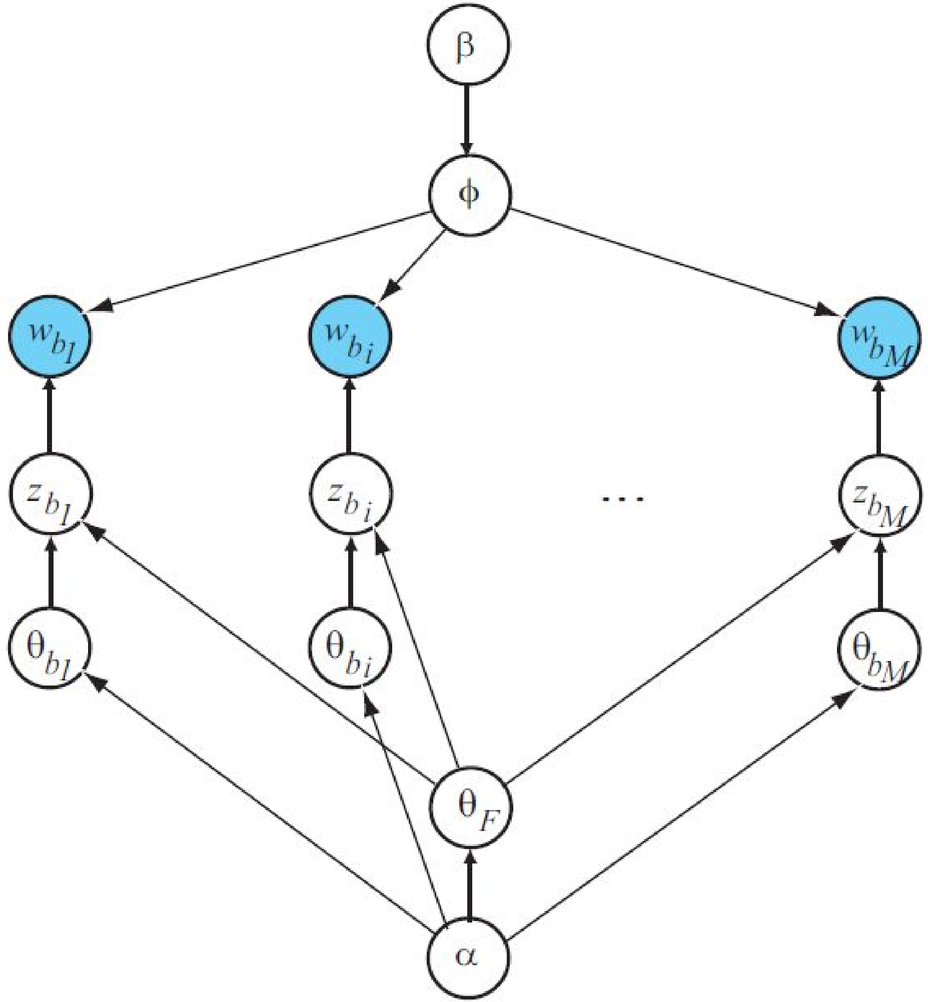


T-Model Example

- Combined topic proportion θ*bi for a bug report bi is a combination of its local topic proportion θbi and θF

  θbi = θbi × θF (Hadamard product)

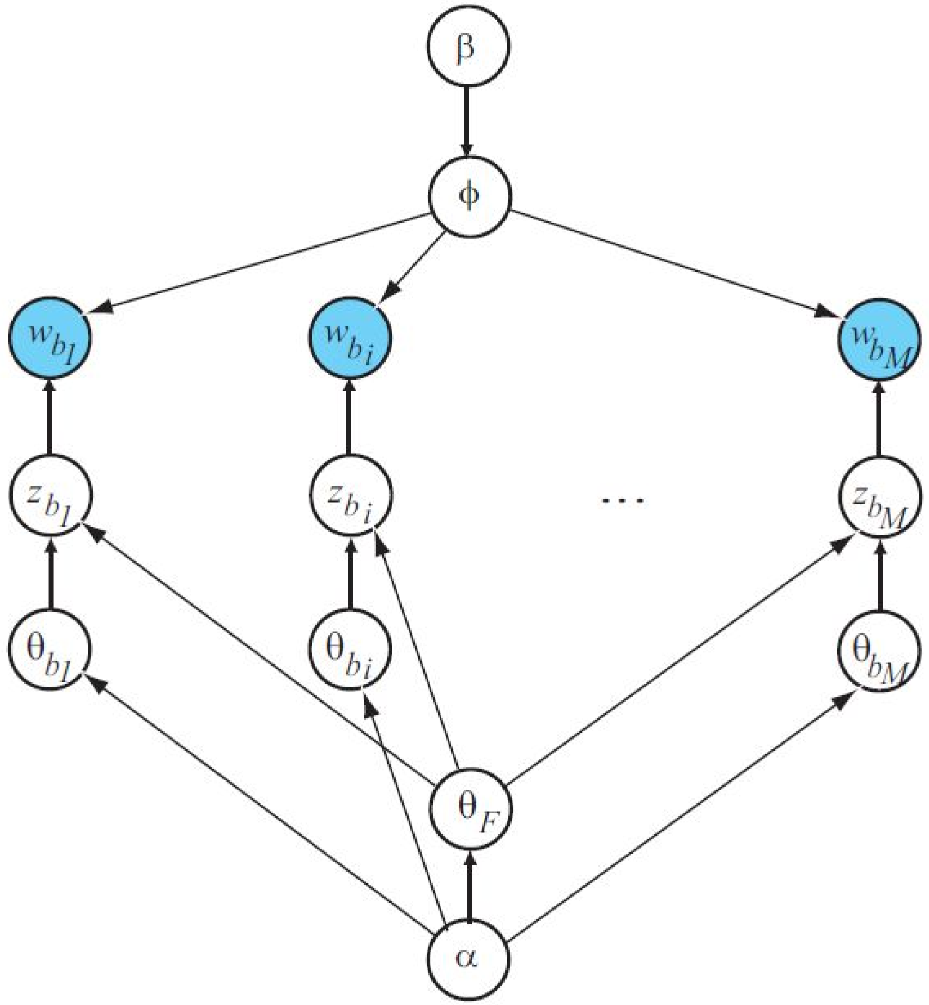
- If a topic k has high proportion in both θbi and θF , it also has a high proportion in θ*bi .



T-Model Example

- Hyper parameters: α (uniform Dirichlet prior on topic distributions θbi and θF) and β (uniform Dirichlet prior on the per-topic word selection distribution ϕ)

- The parameters of the T-Model can be learned and then used to estimate the topics of bug reports and to detect the duplicate ones.



BM25F for Textual Similarity Measure

BM25F is an advanced document similarity function based on weighted word vectors.

Retrieves structured documents composed of several fields.

Each field correspond to a word vector (**bag** of words).

Each field could be assigned different degrees of importance in the retrieval process.

What does it do?

Given a set D of N documents, every document d has F fields.

Bug reports are composed by two fields: summary and description.

BM25F computes the similarity between a query and a document based on the common words.

Two importance factors: global, measured by computing the inverse document frequency (IDF) and, local measured by computing the term frequency (TFD)

Based on these factors, BM25F computes a score for a document d and a query q.

Importance Factors Formulas

To compute TDF of a word t (local)

$$TF_D(d,t) = \sum_{f=1}^{F} \frac{w_f \times occurrences(d[f],t)}{1 - b_f + \frac{b_f \times length_f}{average\_length_f}}$$

To compute IDF of a word t (global)

$$IDF(t) = log\frac{N}{N_d}$$

Note: wf and bf automatically determined.

The magic formula!

$$BM25F(d,q) = \sum_{t \in d \cap q} IDF(t) \times \frac{TF_D(d,t)}{c + TF_D(d,t)} \qquad (3)$$

t is the word that appears in both d and q

c (c ≥ 0) is a parameter that controls the relative contribution of TFD(d, t) to the final score.

Parameters to be specified: wf and bf for each field f, and c. These can be automatically determined based on a set of training data.

Combining the models into DBTM

Two prediction experts, y1 (T-Model), and y2 (BM25F).

The two experts have different advantages in the prediction of duplicate bug reports.

y2 is stricter and better in the detection of duplicate bug reports written with the same textual tokens. However, it does not work well with the bug reports that describe the same issue with different terms.

y1 can detect the topic similarity between two bug reports even when they are not similar in texts. This is less strict in comparison.

Combining the models into DBTM

The authors applied a machine learning technique called Ensemble Averaging.

The combined expert is a linear combination of the two experts:

$y = \alpha_1 \; y_1 + \alpha_2 \; y_2$

α_1 and α_2 are the parameters to control the significance of experts in estimating duplicate bug reports, satisfying $\alpha_1 + \alpha_2 = 1$.

Combining the models into DBTM

If $\alpha1 = 1$ and $\alpha2 = 0$, only topic-based expert is used.

If $\alpha1 = 0$ and $\alpha2 = 1$, only text-based one is used.

The optimal values of $\alpha1$ and $\alpha2$ are learned from the training set

Since $\alpha1 + \alpha2 = 1$ by definition, DBTM has to learn $\alpha1$ only. $\alpha1$ can be learned from the training set.

Training

DBTM will be trained from historical data including bug reports and their duplication information.

The observed words of bug reports and duplicate relations among them will be used to estimate:

-Topic assignment vectors of all bug reports
-Topic proportion of the shared technical issue(s)
-Local topic proportions of the bug reports.

The variables will be trained to make the model fit most with both the report contents and the duplicate relations.

Training

The parameters of trained models are used for estimating text similarity levels and topic similarity levels of a test bug report and a duplicate report group.

Those similarity levels are combined into sim(Btest, Ctest) via a varying weight α1 with the step of 0.01.

The combined similarity values are used to rank the links between bug reports and duplicate report groups.

The higher the combined similarity, the more likely bnew is a duplicate of the reports in the test group..

Prediction

Apply DBTM to a new bug report bnew.

It uses the trained parameters to estimate the topic proportion θbnew of bnew. θbnew is used to find groups of duplicate bug reports which could share technical issue(s)

- i.e having high topic proportion similarity, and therefore are potentially duplicate of bnew.

Finally, all duplicate report groups are ranked, and the top-k most similar groups are shown to bug triagers to check for potential duplications for bnew.

Evaluation

DBTM’s detection accuracy was compared to REP.

Same data sets and evaluation settings as REP: reports sorted in chronological order.

First, sensitivity of DBTM’s accuracy with respect to different numbers of topics K.

When the number of topics increases, accuracy increases as well and becomes stable at some ranges.

However, as K is larger (K>380), accuracy decreases, nuanced topics appear and topics may begin to overlap semantically.

Evaluation

Second, accuracy comparison using the parameter K selected after fine-tuning for best results in the previous step.

DBTM achieves very high accuracy in detecting bug reports.

For a new bug report, it can detect the duplication with a single recommended bug report in 57% of the detection cases.

Within a list of top-5 resulting reports, it detects duplication in 76% of the cases. With a list of 10 reports, detects duplication in 82% of the cases.

It can relatively improve REP by up to 20% in accuracy.

Evaluation

Efficiency results:

DBTM is highly efficient. Evaluated using OpenOffice, Eclipse, and Mozilla.

For a large project like Mozilla, it took about 5 minutes for training (run in background).

For predicting, on average, prediction time for one bug report are just 0.031s, 0.035 s, and 0.041s for OpenOffice, Eclipse, and Mozilla, respectively.

Conclusion

Scalable and efficient model.

It was only evaluated in Eclipse, Mozilla and OpenOffice.

The author’s would want to validate their method against commercial software.

Although, those three projects are sufficient for their analysis.

DBTM can learn sets of terms describing the same issue.

DBTM can improve other approaches by 20% in accuracy.

Questions

In your experience, have you been exposed to duplicate bug reports? If so, what has been done to address this issue in your case?

Combining two approaches, as the authors did, is apparently a good idea as showed in their results. After this presentation, what’s your stance on combining IR based methods with Topic Modeling features?

The authors used Open Source projects for their evaluation. Do you think that evaluating these models using commercial software will provide different results? Think about quality/availability of bug reports.