Automatic Reviewer: Predicting Whether a Paper will be Accepted or Rejected

Hong Wang
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607, USA
hwang207@uic.edu

Shuyang Lin
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607, USA
slin38@uic.edu

Abstract

//TODO

1 Introduction

After submitted a paper to a conference, you, as the author of the paper, usually are most interested in whether your paper will be accepted or not. However, it is the decision made by the reviewers of conference. A question arises here is that can we predict the result? or more specifically, can we find the criteria of a certain conference? In this paper, we introduce Automatic Reviewer, which may give you the answers, even before you submit the paper to the conference.

This paper studies the following two problems: 1. What statistical features of papers are correlated to the decision of the conference? 2. To what extent can an automatic program predict whether a paper will be accepted or not?

We extract several features, which range from paper metadata to paper topics. We use supervised learning algorithm (SVM) to learn our system model, and use cross-fold validation to evaluate our method.

2 Related Work

Predicting has been studied in Machine Learning/Data Mining area for many years [1]. Great successes have been made in areas range from e-commerce to economics by applying such predicting techniques. With the growing popularity of social networking, researchers try to predict the trend by analyzing the topics of posts in the social networking [2].

3 Data Set

Currently, we focus on ACL conference. The full papers accepted by ACL are used as positive samples. As we do not have access to the real rejected papers, we use the papers in the joint workshops with ACL as the negative samples. We made an assumption that the workshop papers are the papers rejected by the main conference. The assumption is obviously not true for every ACL workshop paper. But it still make sense for two reasons: (1) Usually, workshop paper are not of as good quality as the main conference paper. (2) If a paper is rejected by the main conference, the authors will be suggested to submit the paper to the workshops come along with the conference.

We collect the papers which were published in ACL conference and joint workshops in year 2007, 2010, and 2012. They are the latest three ACL conferences that are not a joint conference. We do not want to consider joint conference, because we want to focus on ACL conference alone. After collecting the data, we filter out the workshop paper that is less than 8 pages, since the full papers accpected by ACL conference usually have 9 or 10 pages. The statistics of dataset is listed in Table 4.4

4 Approaches

4.1 Outline

In this study, we first downloaded all the papers from ACL web-site. Then we send them to Automatic Re-

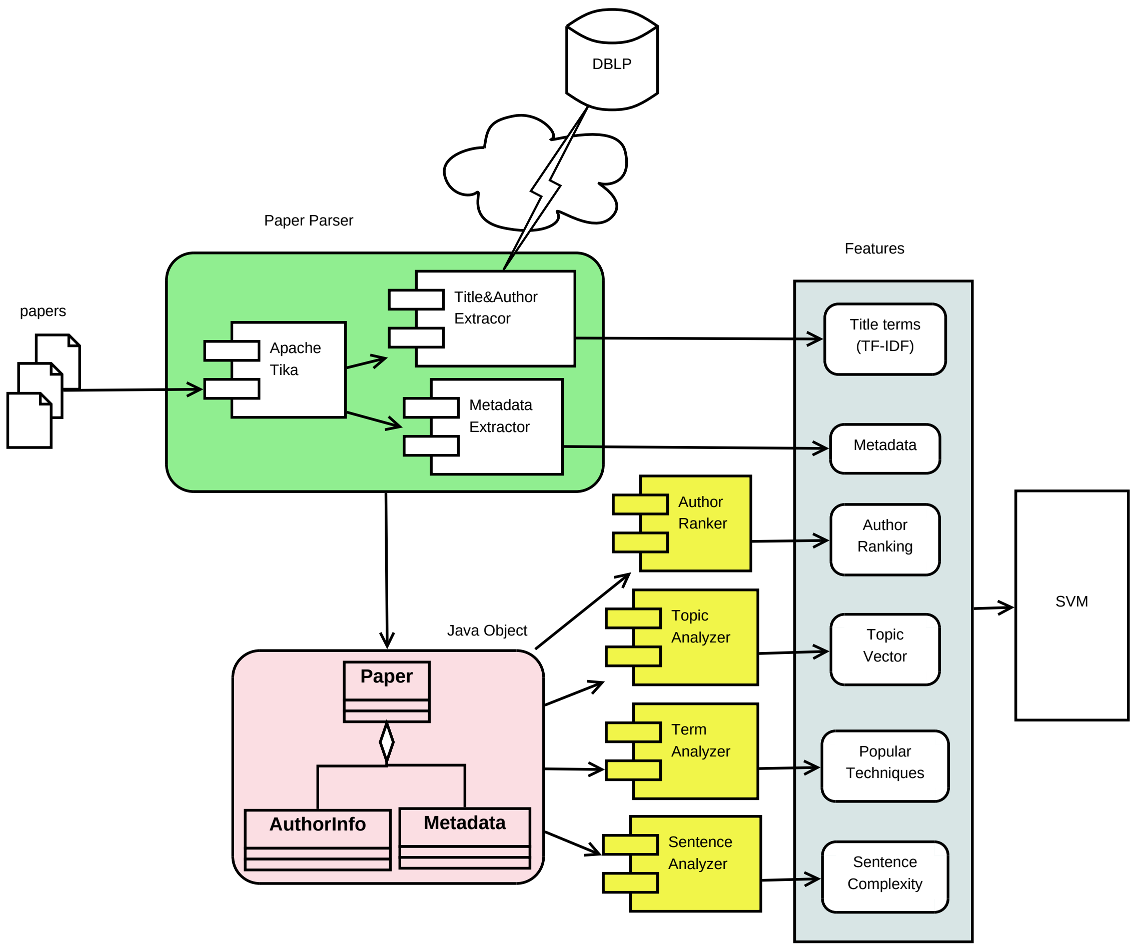


Figure 1: The architecture of the proposed approach

Year	Positive	Negative	Total
2007	99	179	278
2010	160	144	304
2012	111	79	190
Total	370	402	772

Table 1: The Statistics of Dataset

viewer. There are three main components in Automatic Reviewer: 1. paper parser; 2. feature extractors; and 3. model and evaluation component. The architecture is showed in Figure 1.

Papers are first parsed into structured data (Java objects). Metadata from the paper are used directly as features. Then the parsed papers are processed by several feature extractors. Finally, the feature vectors are sent to Support Vector Machines (SVM) for model learning and evaluation.

4.2 Features

We use several features: some of them are directly extracted from the paper; the others are calculated by using more sophisticated techniques.

- **Metadata** Metadata are those information extracted from paper itself, like the number of pages, total numbers of tables/formulas/figures, number of tables/formulas/figures per page, max number of tables/formulas/figures per page. In our current experiment, we only use total number of tables/formulas/figures as features.

- **Author Ranking** We extracted top 2000 authors in "Natural Language & Speech" area from Microsoft Academic Search , and use this extracted list as the authors ranking. If author doesnt appear in the list, the rank will be zero.

- **Popular techniques mentioned in the paper** We also manually created a list of popular techniques which may be used in the NLP research. The terms of techniques and its synonyms, abbreviations are group together. We check each technique in the paper, to see if it has been mentioned or not.
- **Words in the Title (TF-IDF Score)** Based on the assumption that novel ideas are favored in the conference, and papers with novel ideas may have titles that contain infrequent terms which are more attractive to the reviewers, we count the TF-IDF score for the title terms, and use score as one of features.
- **Topic LDA (Latent Dirichlet Allocation)** LDA[??] is used to extract topics from all the papers. In our experiment, top 20 topics are extracted, and the probability distribution vector of topics is used as one feature.
- **Sentence complexity** All the content sentences within papers are parsed into phrase structure trees by using Stanford Parser [??]. The depth of tree is used as complexity of the corresponding sentence. For each paper, we count the number of sentences with each possible value of complexity, and use the frequency of all the possible values of complexity as a feature vector.

4.3 Paper parser

There are three sub-components in the paper parser: Apache Tika , Metadata Extractor, and Title & Author Extractor.

The original papers which downloaded from ACL web-site are first sent to Apache Tika which is a PDF file parser. It can parse PDF file into HTML-like structured data. The paper is parsed into several ¡page¿ tags which contain ¡p¿ tags that denote the raw paragraphs in the original paper.

Because the parsing is not very accurate, the raw paragraphs contain all the texts in the paper, including the page foot, like "Proceedings of ", "BioNLP 20...". After all these noise paragraphs are discarded by regular expression matching, the first raw paragraph is considered as the candidate title. The paragraph starts with "abstract" is the abstract paragraph. If theres no paragraph starts with "abstract", the first paragraph which contains more than 300 English letters is considered as the abstract paragraph. We consider all the paragraphs between title and abstract as raw author info which contains author names, affiliations, and email addresses. All paragraphs after the key-word "reference" or "references" are considered as paper references. The other paragraphs between abstract and references are considered as content paragraphs.

Content paragraphs are then sent to Metadata Extractor. Metadata Extractor uses regular expression `"\^(fig\.|figure)\s*\d+"` and `"\^(tab\.|table)\s*\d+"` to find the figures and tables in each page. Continuous paragraphs which contain more than 30% non- alphabet English letters are considered as one formula. These approaches may not be very accurate. However, since both training and testing papers are parsed based on this same rule, we think the inaccuracy is acceptable. Using the methods above, all the metadata are generated.

Candidate title and raw author information are processed by Title & Author Extractor. This extractor utilizes DBLPs web service API . This API supports fuzzy query, so we send the raw title to the API and retrieve the most reliable one as the real title for the paper. If a paper does not exist in DBLP, the raw title is accepted. Together with the title, author names are also returned from DBLP web API. The extractor uses regular expression to match the email addresses within raw author info, the text between the author name and matched email addresses are considered as the authors affiliation. Currently, only title and author names are used in our experiment.

Through the paper parser, the PDF paper is parsed into structured Java "Paper" object. "Paper" contains one "Metadata", several "Author" objects, and all sentences in each paragraph. The corresponding UML is attached in the Appendix.

4.4 Feature Extractors

Besides the metadata feature which directly comes from the parsed Metadata object, other features are generated from the following 4 feature extractors.

- **Author ranker** Author ranker extracts top 2000 authors in "Natural Language & Speech" area from Microsoft Academic Search. Then the author names, affiliations and ranking are stored in a Lucene index. Because abbreviation in names may appear in the paper, we send fuzzy query of author name to the Lucene index to retrieve the ranking information. Also an empirical threshold matching score is set to filter out those unlikely matching.

- **Term analyzer** Term analyzer tries to catch technical concepts from paper at term-level. Right now this analyzer mainly focuses on the popular technique terms mentioned in the paper. Unlike topics of paper, these technique terms only concentrate on technique aspects of the paper content. We manually created a list of technique terms and their synonyms and abbreviations. Term analyzer uses this term list to generate a Boolean feature vector. Each value in the vector corresponds to the existence of each technique term.

- **Topic analyzer** We believe that during a period of time, certain topics are more interested in certain research area, and those papers studied such topics have more chances been accepted by the conference in that research area. In order to find out what topics are more favored in the conference, we use LDA (Latent Dirichlet allocation) to extract 20 topics from all the training data. The probability distribution of the topics of paper is used as a feature vector.

- **Sentence analyzer** Because there is a limit in the number of pages of conference paper, we think the sentence would be more complicated if the paper contains more information, and such paper are more valuable than others. Based on this assumption, we use sentence analyzer to analyze the complexities of sentences in each paper. The paragraphs in paper are split into sentences first. And then these sentences are parsed by Stanford Parser. The entire parsing process is extremely slow. It costs more than 27 hours in our experiment machine (Intel Core i5, 2.27GHz, 6G RAM for JVM) with 5 working threads in parallel. After we got the parsed phrase structure tree from Stanford Parser, the depth of each sentence tree is used

Term	Forms
LDA	LDA; Latent Dirichlet allocation
HMM	HMM; HMMs; Hidden Markov model
MaxEnt	MaxEnt; maximum entropy
MEMM	MEMM; maximum-entropy Markov model; maximum entropy Markov model
CRF	CRF; CRFs; Conditional random fields
NER	NER; named-entity recognition; named entity recognition; entity identification; entity extraction
SVM	SVM; SVMs; support vector machine; support vector machine; support vector network; support vector networks
LogisticRegression	logistic regression
LinearRegression	linear regression
LSI	LSI; latent semantic indexing; SVD; singular value decomposition; LSA; latent semantic analysis
KLdivergence	KL-divergence; KLdivergence; Kullback-Leibler divergence; Kullback Leibler divergence; information gain
MutualInformation	mutual information;

Table 2: The Statistics of Dataset

as the measurement of sentence complexity.

5 Results and Discussion

We use the SVM with 10-fold cross-validation to evaluate our Automatic Reviewer system. The result of all positive predictions is used as the baseline.

5.1 Evaluation of Each Feature

As first part of our experiment, we evaluate the classifiers that learned using each single feature. For the evaluation, We conduct 10-fold cross-validation on the ACL 2012 data set, and report the precision, recall and F-score for the classier using each single feature.

As shown in the Table 5.1, LDA Topic feature and sentence complexity feature are two best predictors for this problem. Both of they can construct a classifier that is better that the all-positive baseline. Each of the metadata feature (number of pictures/formulas/tables on each page), when is used alone, is a weak predictor. But when we consider all the features from metadata together, we get a classifier that is significantly outperforms the all-positive baseline. Terms in title and popular techniques are two other weak predictors, but they can still make alone a classifier with higher precision than the baseline, which means when they are used together with other features, they can contribute to a better classifier. The author ranking feature is the weakest predictor among all the features we consider.

The result shows that: (1) The full papers accepted by ACL 2012 are statistically different from the workshop papers in the complexity of sentences and topic of the paper. (2) The numbers of pictures/formulas/tables in the full papers accepted by ACL are larger comparing with those of workshop papers. (3) The ranking of authors of a paper does not have influence on whether the paper is an ACL conference full papers or not, which may result from the double-blind reviewing of ACL2012.

5.2 Evaluation on Different Data Sets

We also evaluate the proposed approach on different data sets. We collect the ACL conference and workshop papers on the years 2012, 2010 and 2007, and evaluate our approach by conducting cross-validation on each of the three data sets.

	Precision	**Recall**	**F-score**
#Pictures	63.6	68.5	65.8
#Formulas	74.5	68.5	71.4
#Tables	70.0	80.2	74.8
Metadata	74.8	82.9	78.6
Title	65.2	91.0	76.0
LDA Topic	69.9	91.9	79.4
Popular Techniques	67.0	47.7	55.8
Author Ranking	57.0	76.6	65.4
Sentence Complexity	71.4	90.1	79.7
All Positive	58.4	100.0	73.7

Table 3: Evaluation of Classifiers Using Each Single Feature

	Precision	**Recall**	**F-score**
Metadata	74.8	82.9	78.6
LDA Topic	69.9	91.9	79.4
Sentence Complexity	71.4	90.1	79.7
Metadata &LDA& Complexity	79.8	92.8	**85.8**
All features	80.5	85.6	83.0
All Positive	58.4	100.0	73.7

Table 4: Evaluation on ACL 2012 Data Set

For each dataset we compare the evaluation result of the classifier learned by all the features with the all-positive basline. Since metadata feature, LDA topic feature and sentence complexity feature are three most useful features, as discussed in Section 5.1 , we also list the result of the classifer learned from each of the three features as well as the one learned from three of them. Table 5.2 5.2 and 5.2 show the resulf of ACL 2012, 2010 and 2007 datasets respectively.

As shown in these tables, the proposed approach performs best on the ACL 2012 data set, and can get slightly better performance than the baseline on the ACL 2010 and 2007 data set.

On the ACL 2012 data set, the feautres "Metadata", "LDA topic" and "Sentence Complexity" all

	Precision	**Recall**	**F-score**
Metadata	70.1	68.8	69.4
LDA Topic	65.4	66.3	65.8
Sentence Complexity	65.3	60.0	62.5
Metadata &LDA& Complexity	69.3	72.9	**70.6**
All features	67.2	69.4	68.3
All Positive	52.7	100.0	68.9

Table 5: Evaluation on ACL 2010 Data Set

	Precision	**Recall**	**F-score**
Metadata	35.2	31.3	33.2
LDA Topic	59.2	58.6	58.9
Sentence Complexity	65.3	60.0	**62.5**
Metadata &LDA& Complexity	42.7	41.4	42.1
All features	60.5	49.5	54.5
All Positive	35.6	100.0	52.5

Table 6: Evaluation on ACL 2007 Data Set

have better F-score than the baseline. When three of them considered together, the F-score is significantly better then the baseline. The F-score of the classifier using all features is slightly lower than the three-best-features classifier, but still outperforms the baseline with 10%.

On the ACL 2010 data set, each of the three features is not as strong predictor as it is on the ACL 2012 data set. But when we create a classifier use all of the three of them, the result is better than the baseline. But if we cansider all the features together, the result classifier is not better than the baseline.

On the ACL 2007 data set, the sentence complexity feature alone is a good predictor. It can make a classifer than is 10% outperforms the all-positive baseline. Other features are not as good that the sentence complexity feature, when consider all the features together, the result is not as good as the one using only the sentence complexity feature, but it is still better than the baseline.

As we have discussed, the result of evaluation is different on each dataset, because features may work well on one dataset, but not so well on another dataset. It is still not clear to us why each feature may have different performance on different data set. It may because of that each year the ACL conference has different set of workshops comes alone with it.

5.3 Conclusion and Future Work

From the result of evaluation, we can see that the proposed model can predict whether a paper is accepted or not with good accuracy. The metadata, sentence complexity and the topic distribution learned by LDA model are the best features. This result proved our assumption that certain topics are more favored by certain conference. Also, tables number is the best of all three metadata, which indicates that the reviewers of paper are more willing to see the well structured result of research rather than plain text.

Features like popular techniques and author ranking do not work very good. This phenomena indicates that the quality of ACL conference is very high, the papers are thoroughly studied before accepting/rejecting, and the conference cares more about the content of paper rather than whether the techniques used are popular or not. Because the reviewing is blind, the authors ranking cannot influence the decision of the reviewers.

In the future, we want to study how the topic/style of papers changes over the time. And we also want to extend our research to other conferences in other area. We want to find out the main difference between the accepted papers in different conferences, and different areas. Currently, the negative samples are workshop papers which may not be representative. We are considering using papers come from lower-level conferences within the same research area as the negative samples.

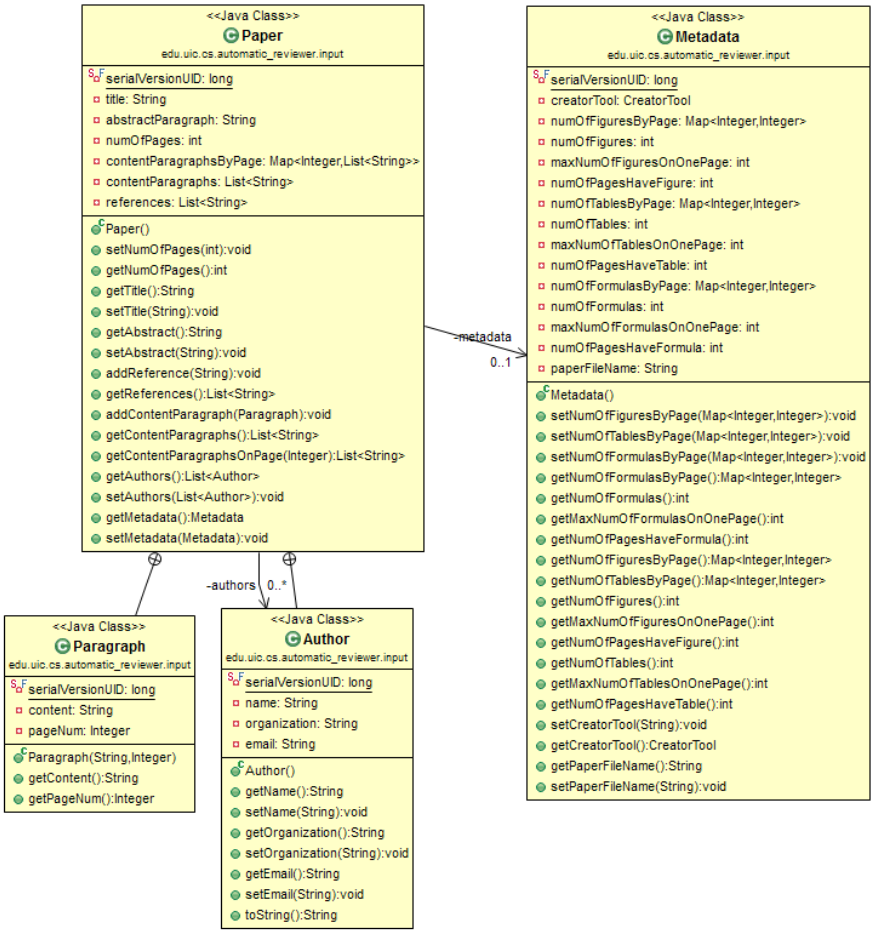
A Additional materials

See the Figure 2 for the parsed Jave Paper object.

B Project Workload

See the Figure 3.

- The blue parts (includs paper parser, major part of feature extracting) are done by Hong Wang.
- The red parts (includs data collecting, minor part of feature extrating, feature generating and classifier) are done by Shuyang Lin.
- This project report and previous presentation slides are done by both of them.



<<Java Class>>
Paper
edu.uic.cs.automatic_reviewer.input
serialVersionUID: long
title: String
abstractParagraph: String
numOfPages: int
contentParagraphsByPage: Map<Integer,List<String>>
contentParagraphs: List<String>
references: List<String>
Paper()
setNumOfPages(int):void
getNumOfPages():int
getTitle():String
setTitle(String):void
getAbstract():String
setAbstract(String):void
addReference(String):void
getReferences():List<String>
addContentParagraph(Paragraph):void
getContentParagraphs():List<String>
getContentParagraphsOnPage(Integer):List<String>
getAuthors():List<Author>
setAuthors(List<Author>):void
getMetadata():Metadata
setMetadata(Metadata):void
-metadata
0..1
<<Java Class>>
Metadata
edu.uic.cs.automatic_reviewer.input
serialVersionUID: long
creatorTool: CreatorTool
numOfFiguresByPage: Map<Integer,Integer>
numOfFigures: int
maxNumOfFiguresOnOnePage: int
numOfPagesHaveFigure: int
numOfTablesByPage: Map<Integer,Integer>
numOfTables: int
maxNumOfTablesOnOnePage: int
numOfPagesHaveTable: int
numOfFormulasByPage: Map<Integer,Integer>
numOfFormulas: int
maxNumOfFormulasOnOnePage: int
numOfPagesHaveFormula: int
paperFileName: String
Metadata()
setNumOfFiguresByPage(Map<Integer,Integer>):void
setNumOfTablesByPage(Map<Integer,Integer>):void
setNumOfFormulasByPage(Map<Integer,Integer>):void
getNumOfFormulasByPage():Map<Integer,Integer>
getNumOfFormulas():int
getMaxNumOfFormulasOnOnePage():int
getNumOfPagesHaveFormula():int
getNumOfFiguresByPage():Map<Integer,Integer>
getNumOfTablesByPage():Map<Integer,Integer>
getNumOfFigures():int
getMaxNumOfFiguresOnOnePage():int
getNumOfPagesHaveFigure():int
getNumOfTables():int
getMaxNumOfTablesOnOnePage():int
getNumOfPagesHaveTable():int
setCreatorTool(String):void
getCreatorTool():CreatorTool
getPaperFileName():String
setPaperFileName(String):void
-authors
0..*
<<Java Class>>
Paragraph
edu.uic.cs.automatic_reviewer.input
serialVersionUID: long
content: String
pageNum: Integer
Paragraph(String,Integer)
getContent():String
getPageNum():Integer
<<Java Class>>
Author
edu.uic.cs.automatic_reviewer.input
serialVersionUID: long
name: String
organization: String
email: String
Author()
getName():String
setName(String):void
getOrganization():String
setOrganization(String):void
getEmail():String
setEmail(String):void
toString():String

Figure 2: UML for the parsed Java Paper object

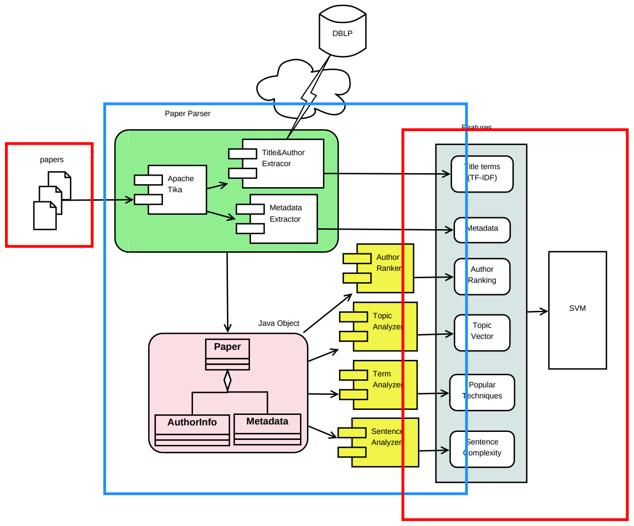


Figure 3: Wordload