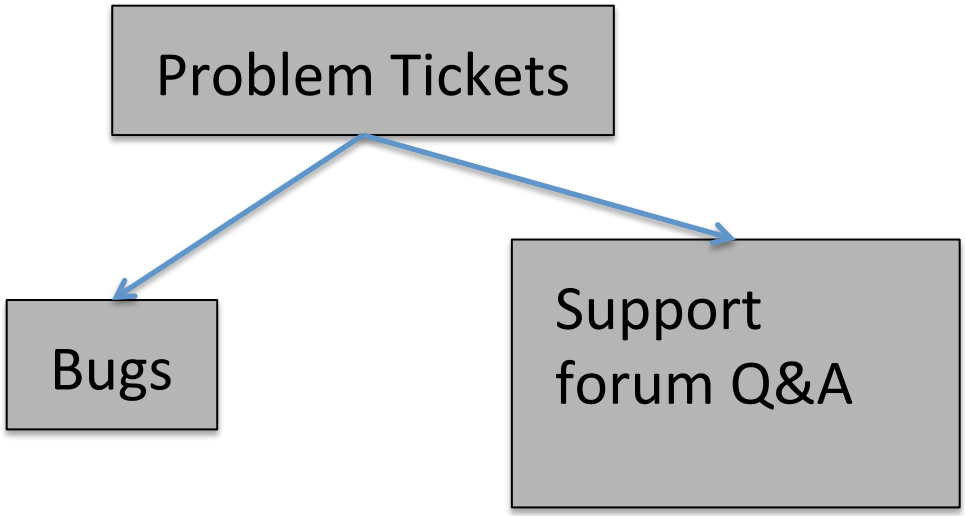
Mining and Summarization of Software Problem Reports

Senthil Mani
Vibha S Sinha
Karthik Sankaranarayanan

IBM Research - India

Objective

Acquaint audience with typical scenarios and techniques where text mining is used on problem tickets



Bug's Life

Bug gets opened

Did this bug exist before ?

1. Detection of Duplicate Defect Reports Using Natural Language Processing. Runeson et al. ICSE 2007.
2. A approach to detecting duplicate bug reports using natural language and execution information. Wang et al. ICSE 2008
3. Duplicate bug report detection with a combination of information retrieval and topic modeling. Nguyen et al. ASE 2012

Bug's Life

Bug gets opened

Who should fix this bug?

1. Automatic bug triage using text categorization. Cubranic. ICSE 2004
2. Who should fix this bug ? Anvik et al. ICSE 2006
3. Improving bug triage with bug tossing graphs. Jeong et al. FSE 2009
4. Fuzzy set and cache-based approach for bug triaging. Tamrawi et al. FSE 2011

Bug's Life

Bug gets opened

How long would it take to fix this bug ? What is it's severity ?

1. How Long Will It Take to Fix This Bug? Weiss et al. MSR 2007
2. Automated severity assessment of software defect reports. Menzies et al. ICSM 2008

Bug's Life

Bug gets assigned

How should I fix this bug ?

1. DebugAdvisor: A Recommender System for Debugging. Ashok et al. FSE 2009
2. A topic-based approach for narrowing the search space of buggy files from a bug report. Nguyen et al. ASE 2011

Bug's Life

Bug gets closed

What was the bug about ?

1. Summarizing Software Artifacts: A Case Study of Bug Reports. Rastkar et al. ICSE 2010
2. AUSUM: approach for unsupervised bug report summarization. Mani et al. FSE 2012

Mining Support Forums

What are people having most trouble with ?

What are some frequently asked questions?

1. Automated Support for Managing Feature Requests in Open Forums. J.Cleland-Huang et al. ACM Communications October 2009
2. Semi-automatically Extracting FAQs to Improve Accessibility of Software Development Knowledge. Stefan et al. ICSE 2012
3. Mining whining in support forums with frictionary. Ko et al. Alt Chi 2012

Use of text mining/learning on non-bug artifacts

Fault Prediction

1. Bug Prediction Based on Fine-Grained Module Histories. Hata et al. ICSE 2012
2. How, and Why Process Metrics are Better. Rahman et al. ICSE 2013

Search on code repository

1. Is text search an effective approach for fault localization: a practitioners perspective. SPLASH Wavefront 2012
2. Retrieval from software libraries for bug localization: a comparative study of generic and composite text models. Rao et al. MSR 2010

Identify topics from code

1. Automated topic naming to support cross-project analysis of software maintenance activities. Hindle et al. MSR 2011

Code summarization

1. Supporting program comprehension with source code summarization. Haiduc et al. ICSE 2012.
2. Towards automatically generating summary comments for Java methods. Sridhara et al. ASE 2010

Session Organization

- Sources and nature of problem data (15 mins)
- Techniques used to analyze (45 mins)
- Examples of tools that can be used (20 mins)
- State of the art papers (30 mins)

DATA SOURCES

What are software repositories?

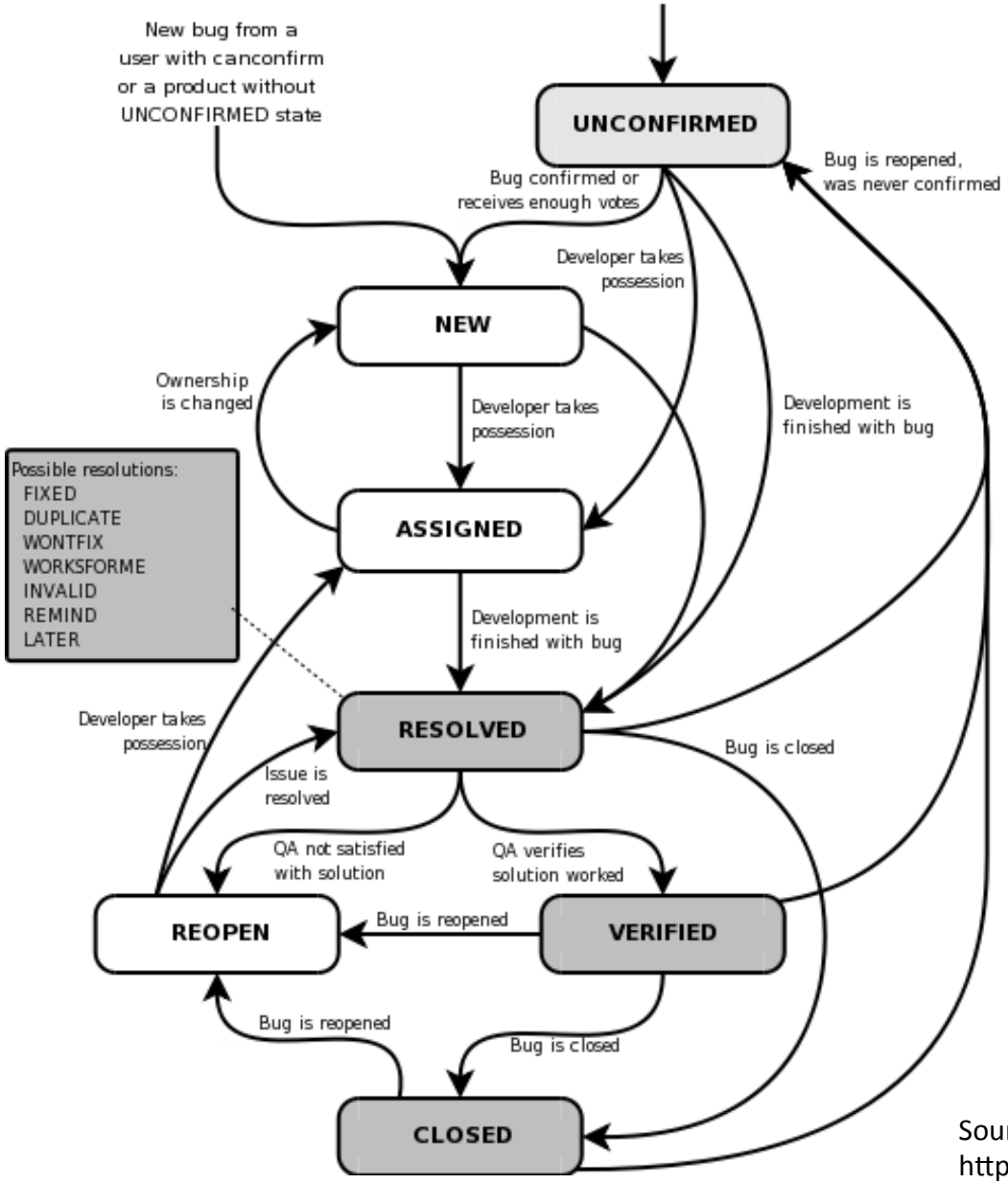
Any repository that holds information about any software artifact

- Code : Code Version History (CVS, SVN etc.)
- Bug / Problem Tickets : Bug / Issue Tracking System (Bugzilla, JIRA, RTC etc.)
- Q&A or Support Forums (StackExchange etc.)

	CVS	Bug Repository	Q&A Forums
CONTEXT	Code evolves as software development progresses. Multiple people might work collaboratively on the same code. Further, they might work from distributed environment	No code is bug free. People using the software need to reports bugs they face. Provide all necessary information about the bug, so that developers can recreate the bug and fix it.	Users need to understand the issues they observe in the product to decide whether they have made some mistakes (or) the issue is indeed a bug. Also, they might need specific information regarding the product.
FEATURES	• Help in tracking changes in the code • Help in synchronizing changes done by multiple people	• Provide ways to report a bug • Provide necessary structure to capture all details related to the bug • Associate a life-cycle (status) to the bug	• Post questions • Post Answers to the questions • Provides ways to tag questions • Search over existing questions • Ways to solicit participation

Focus on Bug Repository and Q&A Forums

Life-cycle of a Bug Report (Bugzilla)



Source
http://www.bugzilla.org/docs/2.18/html/lifecycle.html

Sample Bug Report

Basic fields of a bug report

Durga Deep Tirunagari 2007-01-24 12:47:36 PST Description

```
404: File Not Found

/projects/security/pki/nss/ref/nss/index.htm

We are sorry, the file you requested could not be found.

Referring page:

http://www.mozilla.org/projects/security/pki/nss/ref/ssl/gtstd.html

Please file a bug to let us know that we have a broken link. If you don't have an
account, you will need to register with Bugzilla first (link opens in a new
window), then return here and click the "file a bug" link.
```

David Boswell 2008-09-10 13:57:09 PDT Comment 1

```
Copying Security module owners for input.
```

glen beasley 2008-09-10 14:19:26 PDT Comment 2

```
The NSS documentation needs to be updated. The SSLSample code has been removed in
bug 359302.

I removed the links to the SSLSample code. here is my changes. They will likely
not show up on the site for another day.



--- mozilla-org/html/projects/security/pki/nss/ref/ssl/gtstd.html
+++ mozilla-org/html/projects/security/pki/nss/ref/ssl/gtstd.html
@@ -19,7 +19,7 @@
 <A NAME="1005439">
 Getting Started With SSL</A></H2></CENTER>
 <A NAME="1006945">
-This chapter describes how to set up your environment, including certificate and
key databases, to run the NSS sample code. The sample code and makefiles are
available via LXR in the <A
HREF="http://lxr.mozilla.org/mozilla/source/security/nss/cmd/SSLsample/">SSLSamples</A>
 directory.</A></P>
+This chapter describes how to set up your environment, including certificate and
key databases.</P>
 <A NAME="1013252">
 <A HREF="#1011970">SSL, PKCS #11, and the Default Security Databases</A><br>
```

Information from Bug Report

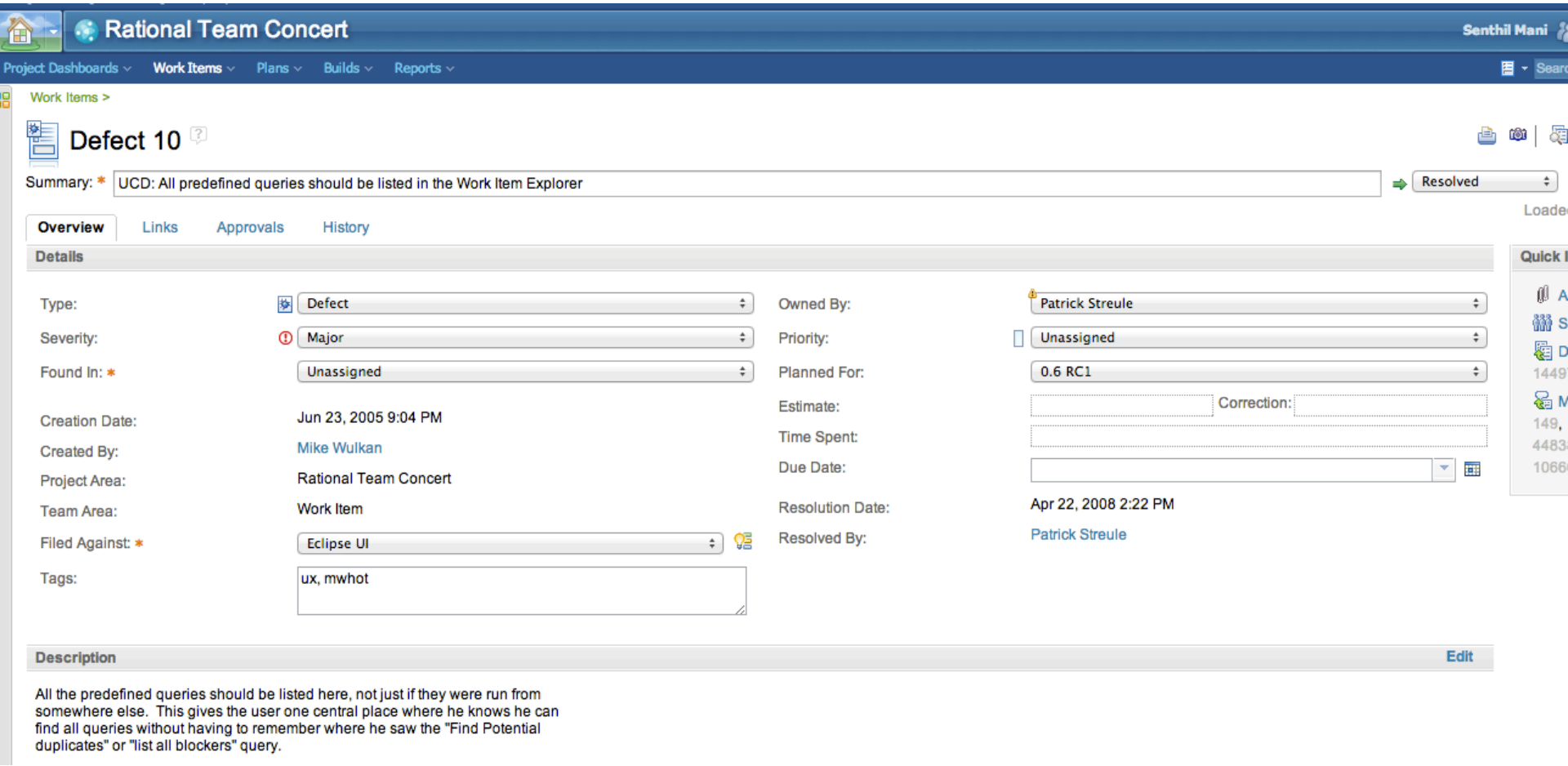
Basic Fields :
Priority, Severity, Module or Application, Created Date, Closed Date

Unstructured Fields :
Comments, Description

User Specific:

- Who reported the bug
- Who closed the bug
- Users who commented on the bug
- Users who added attachments

Sample Bug Report from RTC



Some Basic Fields

Source :
https://jazz.net/jazz/web/projects/Rational%20Team%20Concert#action=com.ibm.team.workitem.viewWorkItem&id=10

Defect 10

Summary: * UCD: All predefined queries should be listed in the Work Item Explorer

Overview | Links | Approvals | **History**

History

Change by Patrick Streule (Apr 22, 2008 2:22:54 PM)

Status	Reopened → Resolved
Resolution	Unresolved → Duplicate
Planned For	<Unassigned> → 0.6/0.6 RC1
Comments [#11]	This work item has been marked as a duplicate of work item 14497.
Resolved By	Unassigned → Patrick Streule (archived)
Resolution Date	<Unassigned> → Apr 22, 2008

Change by Mike Wulkan (Dec 13, 2007 11:15:50 PM)

Tags	added: mwhot

Change by Andre Weinand (Mar 7, 2007 2:53:19 PM)

Owned By	Unassigned → Patrick Streule (archived)
Priority	Medium → Unassigned

Change by Marcel Bihr (Feb 23, 2007 7:32:58 PM)

Subscribed By	removed: Marcel Bihr (archived)

Change by Marcel Bihr (Feb 23, 2007 7:32:07 PM)

Subscribed By	added: Marcel Bihr (archived)

Change by Mike Wulkan (Jan 30, 2007 11:40:31 PM)

Severity	Normal → Major
Comments [#10]	Just poking on this one again to see where it stands? I strongley believe that parameterized queries is a fundamental functional requirement and that supporting them in a uniform way throughout the UI as described

Bug Tracking Systems

Bugzilla : http://www.bugzilla.org/
Projects: Mozilla, Linux Kernel, Gnome, KDE, Apache project, Open office, Eclipse,
Companies : NASA and Facebook

Atlassian - JIRA: http://www.atlassian.com/software/jira/overview
Projects : Apache (https://issues.apache.org/jira/secure/Dashboard.jspa), Hibernate, Sonatype, Spring, Codehaus, Flex, JBoss

Rational Team Concert – RTC : https://jazz.net/products/rational-team-concert/
Open Source Project : RSSOwl (http://www.rssowl.org/overview)

Sample Q&A Forum

Questions | Tags | Users | Badges | Unanswered

How can I print line side by side in Java?

0

1

I couldn't find this anywhere; I might finding it with the wrong keywords.

Pictures paint a thousand words, so let me explain.

Supposed we have a set of unknown number of String:

```
String hello = "Hello world\n Welcome\n"
String goodbye = "Goodbye\n See you in the next life\n"
String do = "Do something\n Be part of us\n"
```

I would like a function that produce such result:

```
String hellogoodbyedo = "
  Hello world_____Goodbye_____Do Something\n
  Welcome_____See you in the next life_____Be part of us\n"
```

In which _ means spaces. Is there a smart way of doing such?

java | algorithm | printing | system.out tags

share | improve this question

edited May 5 '12 at 5:33

asked Apr 29 '12 at 18:00
Stella Lie
18 •3

user who asked the question

5 Answers

active oldest votes

votes

4

You can use

```
System.out.printf("%-20s%-s20s%-20s%n", field1, field2, field3);
```

share | improve this answer

edited Apr 29 '12 at 18:15

answered Apr 29 '12 at 18:02

Peter Lawrey
144k ●15 ●99 ●218

answer accepted

user who answered the question

comments

`println()` doesn't take a format string. – Adam Liss Apr 29 '12 at 18:08

1 @AdamLiss True, but printf does. ;) – Peter Lawrey Apr 29 '12 at 18:12

Yes. Much better after the edit, and amazing it was upvoted while it was still incorrect. The perils of crowd-sourcing, I suppose, and probably why government survives. :-) – Adam Liss Apr 29 '12 at 18:13

format string has typos. Try: `System.out.printf("%-20s%-20s%-20s", field1, field2, field3);` – maybeWeCouldStealAVan Apr 29 '12 at 18:14

1 @AdamLiss Indeed. Fixed now. – Peter Lawrey Apr 29 '12 at 18:16

show 4 more comments

Nature of Data

Structured

Any fields or attributes of bug reports of Q&A forums which **does not contain** free text

- Bug Reports : Priority, Severity, Application, Component, Start – End – Modified Dates, People information etc.
- Q&A Forums : Date of post (question or answer), people information (reputation score, badges), number of votes, accepted answer or not, tags etc.

Unstructured

Any filed or attributes of bug reports or Q&A forums which **contains** free text

- Bug Reports : Title, Summary, Description, Comments, Running Notes
- Q&A Forums : Question, Answers and Comments

Nature of Data

Variations in Unstructured Text Field of Bug Reports

- Dumps of Stack Traces
- Dumps of Error Logs

} Should have been provided as Attachments

- Email Snippets
- Chat Transcripts
- People Names
- Manually created vs. Auto Generated

} Typically observed in bug reports in commercial software projects

Data Sources

Mirrors of version archives and bug databases for Mozilla and Firefox and Eclipses
http://msr.uwaterloo.ca/msr2008/challenge/

Repository logs of over 500+ Gnome Projects, XML dump of the bug databases, and the complete SVN repositories of GNOME projects
http://msr.uwaterloo.ca/msr2009/challenge/

Database with information about packages and bug reports of Debian and Ubuntu
http://udd.debian.org/

Eclipse Bug Data for several releases
http://www.st.cs.uni-saarland.de/softevo/bug-data/eclipse/

FLOSSMole: Database of all SourceForge.net projects
http://flossmole.org/

SatckExchange Forum Data Set
http://2013.msrconf.org/challenge.php

Source
http://www.slideshare.net/herraiz/20100618-daniel-uah

Conferences and Journals

International Working Conference Mining Software Repositories (MSR)
http://2013.msrconf.org/

International Conference on Software Maintenance (ICSM)
http://icsm2013.tue.nl/

International Conference on Software Engineering (ICSE)
http://2014.icse-conferences.org/

Foundations of Software Engineering (FSE)
http://esec-fse.inf.ethz.ch/

IEEE Transactions on Software Engineering (TSE)
http://www.computer.org/portal/web/tse

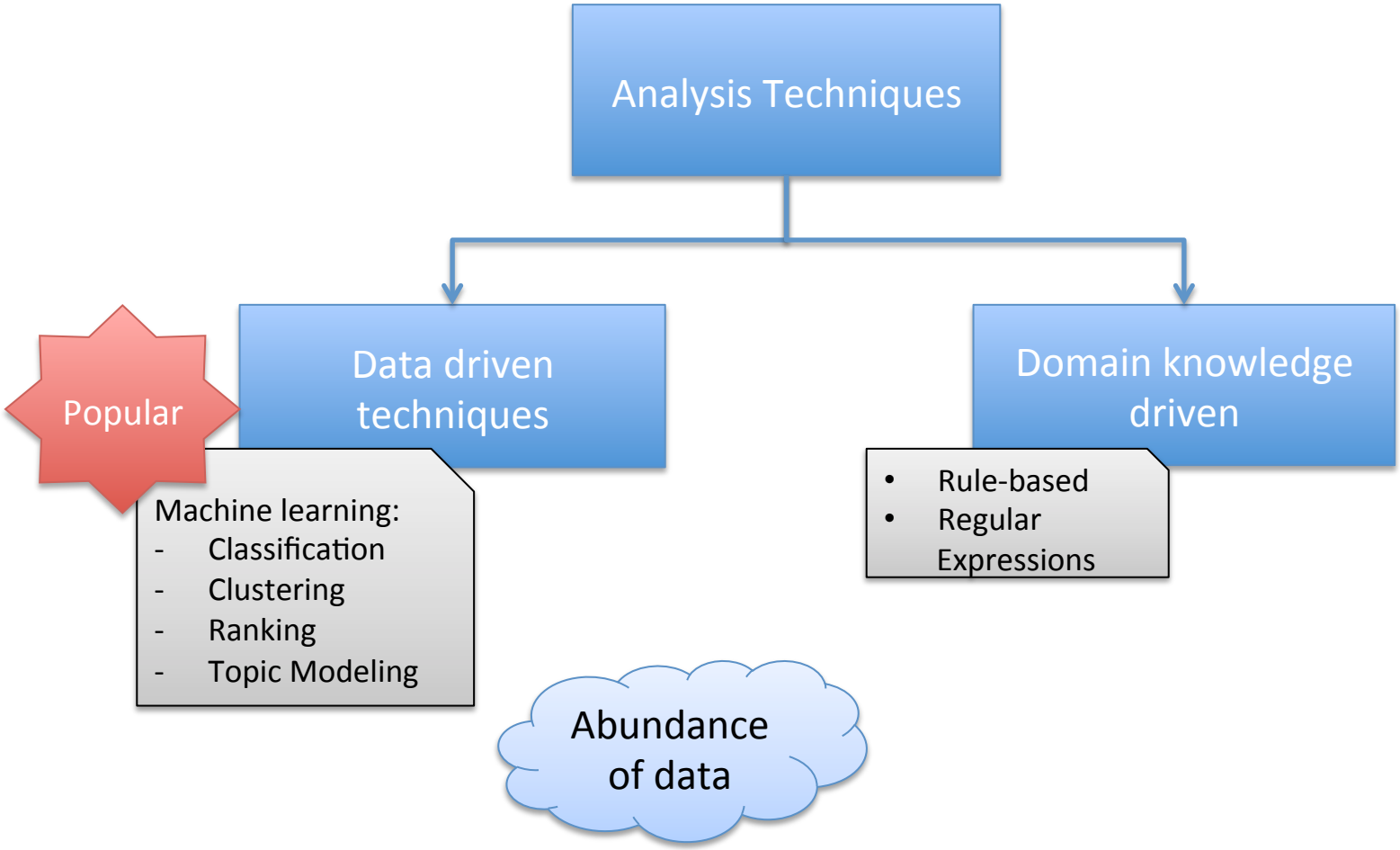
Empirical Software Engineering
http://link.springer.com/journal/10664

ACM Transactions on Software Engineering and Methodology (TOSEM)
http://tosem.acm.org/

Source
http://www.slideshare.net/herraiz/20100618-daniel-uah

TECHNIQUES USED

Techniques Used



Machine Learning Techniques

- Classification or Categorization

  "boiling down" of document content to "pre-defined labels" which "does not lead to discovery of new information" since "presumably the person who wrote the document knew what it was about" - Hearst (1999)

- Clustering

  Group documents into natural categories that arise from statistical, lexical, and semantic analysis rather than the arbitrarily pre-determined categories – M. Sharp (2001)

A *Document* is a bug report, problem ticket, any piece of free text…

How do you represent these documents?

Outline

- Document Representation
  - Similarity measures
- Classification
- Clustering
  - Flat Clustering
  - Hierarchical
- Topic Modeling

Outline

- Document Representation
  - Similarity measures
- Classification
- Clustering
  - Flat Clustering
  - Hierarchical
- Topic Modeling

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

31

Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

32

Bag of words model

- We do not consider the order of words in a document.
- *John is quicker than Mary*

  and

  Mary is quicker than John

  are represented the same way.
- This is called a bag of words model
- Simple, works well.

33

Term frequency tf

- The term frequency $tf_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- Raw term frequency is not what we want because:
  - A document with tf = 10 occurrences of the term is more relevant than a document with tf = 1 occurrence of the term, but not 10 times more relevant.
  - Relevance does not increase proportionally with term frequency.
- Log frequency weighting
  - The log frequency weight of term t in d is defined as follows

$$w_{t,d} = \begin{cases} 1 + \log_{10} tf_{t,d} & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

  - $tf_{t,d} \rightarrow w_{t,d}$:

    0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.

34

Frequency within document vs. frequency in entire collection

- In addition to term frequency (the frequency of the term in the document) . . .

. . .we also want to use the frequency of the term in the collection for weighting and ranking.

- Frequent terms are less informative than rare terms.
- Consider a term that is frequent in the collection (e.g., BUG).

. . . words like BUG, ERROR, TICKET are not discriminatory/important.

- We will use document frequency to factor this into representation

- The document frequency is the number of documents in the collection that the term occurs in.

35

idf weight

- df_t is the document frequency, the number of documents that *t* occurs in.
- df_t is an inverse measure of the informativeness of term *t*.
- We define the idf weight of term t as follows:

$$\text{idf}_t = \log_{10} \frac{N}{\text{df}_t}$$

(*N* is the number of documents in the collection.)

- idf_t is a measure of the informativeness of the term.
- [log N/df_t] instead of [N/df_t] to "dampen" the effect of idf
  - Note that we use the log transformation for both term frequency and document frequency.

36

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- The tf-idf weight . . .
  - . . . increases with the number of occurrences within a document. (term frequency)
  - . . . increases with the rarity of the term in the collection. (inverse document frequency)

- Best known weighting scheme in information retrieval

37

Binary → count → weight matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0
MERCY	1.51	0.0	1.90	0.12	5.25	0.88
WORSER	1.37	0.0	0.11	4.15	0.25	1.95
. . .						

Each document is now represented as a real-valued vector of tf idf weights $\in R^{|V|}$.

38

Documents as vectors

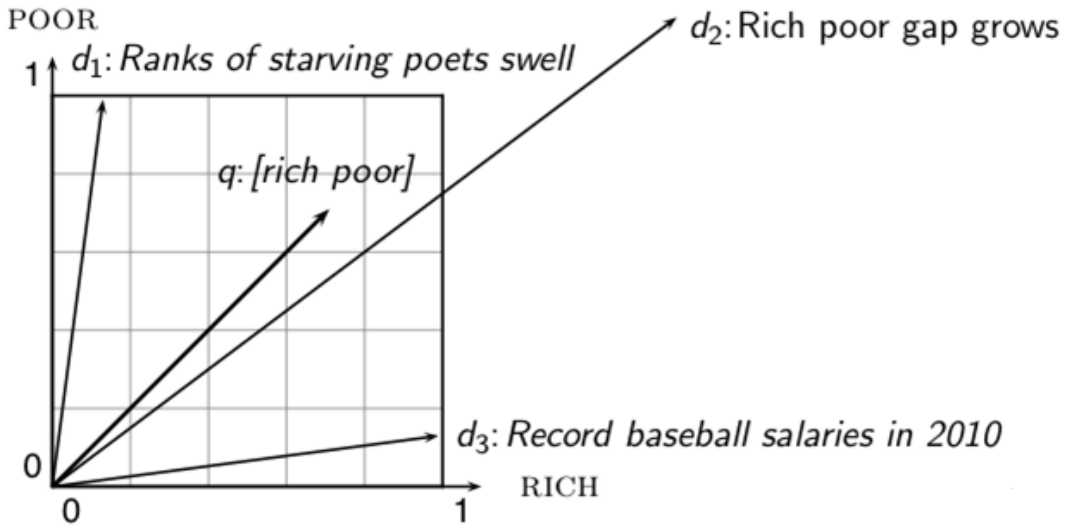
- Each document is now represented as a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$.
- So we have a $|V|$-dimensional real-valued vector space.
- Terms are axes of the space.
- Documents are points or vectors in this space.

How do you compute which documents are similar and which are not?

39

Vector space similarity?

- How about: (negative) distance between two points
  - = distance between the end points of the two vectors
- Euclidean distance is a bad idea . . .
  - . . . because Euclidean distance is large for vectors of different lengths.



The Euclidean distance of $\vec{q}$ and $\vec{d}_2$ is large although the distribution of terms in the query *q*
and the distribution of terms in the document d_2 are very similar.

40

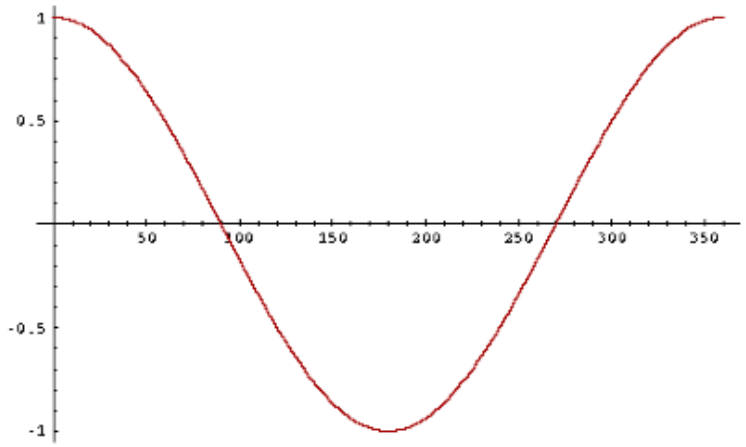
Use angle instead of distance

- Rank documents according to angle with query
- Thought experiment: take a document d and append it to itself. Call this document *d′*. *d′* is twice as long as *d*.
- "Semantically" *d* and *d′* have the same content.
- The angle between the two documents is 0, corresponding to maximal similarity . . .
- . . . even though the Euclidean distance between the two documents can be quite large.

41

From angles to cosines

- The following two notions are equivalent.
  - Rank documents according to the angle between query and document in decreasing order
  - Rank documents according to cosine(query,document) in increasing order
- Cosine is a monotonically decreasing function of the angle for the interval [0°, 180°]



42

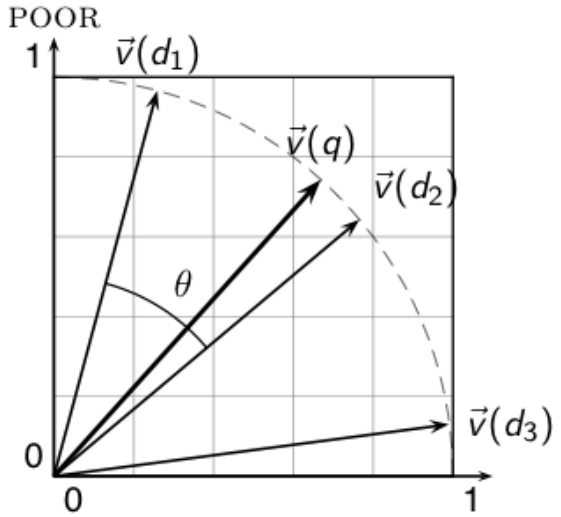
Cosine similarity between query and document

$$\cos(\vec{q},\vec{d}) = \text{SIM}(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- q_i is the tf-idf weight of term *i* in the query.
- d_i is the tf-idf weight of term *i* in the document.
- For normalized vectors, the cosine is equivalent to the dot product or scalar product.

$$\cos(\vec{q},\vec{d}) = \vec{q}\cdot\vec{d} = \sum_i q_i \cdot d_i$$

- (if $\vec{q}$ and $\vec{d}$ are length-normalized).



Outline

- Document Representation
  - Similarity measures
- Classification
- Clustering
  - Flat Clustering
  - Hierarchical
- Topic Modeling

Classification: Training

Given:

- A document space X
  - Documents are represented in this space – typically some type of high-dimensional space.
- A fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
  - The classes are human-defined for the needs of an application (e.g., relevant vs. nonrelevant).
- A training set D of labeled documents with each labeled document $\langle d, c \rangle \in X \times C$

Using a learning method or learning algorithm, we then wish to learn a classifier Υ that maps documents to classes:

$$\Upsilon : X \rightarrow C$$

Classification: Application or Testing

Given: a description $d \in X$ of a document

Determine: $\Upsilon(d) \in C$,

that is, the class that is most appropriate for d

Linear classifiers

- Linear classifiers compute a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
- Classification decision: $\sum_i w_i x_i > \theta?$

- Geometrically, the equation $\sum_i w_i x_i = \theta$ defines a line (2D), a plane (3D) or a hyperplane (higher dimensionalities).
- Assumption: The classes are linearly separable.
- Methods for finding a linear separator: Perceptron, Naive Bayes, linear support vector machines….

Let's look at Naïve Bayes Classifier…

Very popular

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document *d* being in a class *c* as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document. (number of tokens)
- $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class *c*
- $P(t_k|c)$ as a measure of how much evidence t_k contributes that *c* is the correct class.
- $P(c)$ is the prior probability of *c*.
- If a document's terms do not provide clear evidence for one class vs. another, we choose the c with highest $P(c)$.

Maximum a posteriori class

- Our goal in Naive Bayes classification is to find the "best" class.
- The best class is the most likely or maximum a posteriori (MAP) class c_{map}:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)$$

Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- Since $\log(xy) = \log(x) + \log(y)$, we can sum log probabilities instead of multiplying probabilities.
- Since log is a monotonic function, the class with the highest score does not change.
- So what we usually compute in practice is:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \left[\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c) \right]$$

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

- Simple interpretation:
  - Each conditional parameter log $\hat{P}(t_k|c)$ is a weight that indicates how good an indicator t_k is for c.
  - The prior log $\hat{P}(c)$ is a weight that indicates the relative frequency of c.
  - The sum of log prior and term weights is then a measure of how much evidence there is for the document being in the class.
  - We select the class with the most evidence.

Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from train data: How?
- Prior:

$$\hat{P}(c) = \frac{N_c}{N}$$

- N_c : number of docs in class *c*; *N*: total number of docs
- Conditional probabilities:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- T_{ct} is the number of tokens of *t* in training documents from class *c* (includes multiple occurrences)
- We've made a Naive Bayes independence assumption here: $\hat{P}(t_{k_1}|c) = \hat{P}(t_{k_2}|c)$

Naive Bayes: Training

TRAINMULTINOMIALNB($\mathbb{C}, \mathbb{D}$)

1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
5 $prior[c] \leftarrow N_c/N$
6 $text_c \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS($\mathbb{D}, c$)
7 **for each** $t \in V$
8 **do** $T_{ct} \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 **return** $V, prior, condprob$

Naive Bayes: Testing

```
APPLYMULTINOMIALNB(ℂ, V, prior, condprob, d)
1  W ← EXTRACTTOKENSFROMDOC(V, d)
2  for each c ∈ ℂ
3  do score[c] ← log prior[c]
4     for each t ∈ W
5     do score[c] += log condprob[t][c]
6  return arg max_{c∈ℂ} score[c]
```

Time complexity of Naive Bayes

mode	time complexity
training	$\Theta(\|\mathbb{D}\|L_{ave} + \|\mathbb{C}\||V|)$
testing	$\Theta(L_a + \|\mathbb{C}\|M_a) = \Theta(\|\mathbb{C}\|M_a)$

- L_{ave}: average length of a training doc, L_a: length of the test doc, M_a: number of distinct terms in the test doc, $\mathbb{D}$: training set, V : vocabulary, $\mathbb{C}$: set of classes
- $\Theta(|\mathbb{D}|L_{ave})$ is the time it takes to compute all counts.
- $\Theta(|\mathbb{C}||V|)$ is the time it takes to compute the parameters from the counts.
- Generally: $|\mathbb{C}||V| < |\mathbb{D}|L_{ave}$
- Test time is also linear (in the length of the test document).
- Thus: Naive Bayes **is linear** in the size of the training set (training) and the test document (testing). This is optimal.

Naive Bayes is not so naive

- Naive Bayes has won some bakeoffs (e.g., KDD-CUP)
- More robust to non-relevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- Better than methods like decision trees when we have many equally important features
- A good dependable baseline for text classification (but not the best)
- Optimal if independence assumptions hold (never true for text, but true for some domains)
- Very fast
- Low storage requirements

Evaluating classification

- Evaluation must be done on test data that are independent of the training data (usually a disjoint set of instances).
- It's easy to get good performance on a test set that was available to the learner during training (e.g., just memorize the test set).
- Measures: Precision, recall, F_1, classification accuracy

Precision *P* and recall *R*

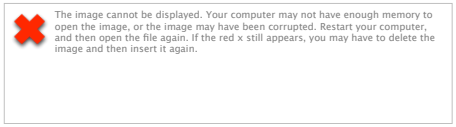
	in the class	not in the class
predicted to be in the class	true positives (TP)	false positives (FP)
predicted to not be in the class	false negatives (FN)	true negatives (TN)

$$P = TP / (TP + FP)$$

$$R = TP / (TP + FN)$$

A combined measure: *F*

- F_1 allows us to trade off precision against recall.
- $$F_1 = \frac{1}{\frac{1}{2}\frac{1}{P} + \frac{1}{2}\frac{1}{R}} = \frac{2PR}{P+R}$$
- This is the harmonic mean of *P* and *R*:



Outline

- Document Representation
  - Similarity measures
- Classification
- Clustering
  - Flat Clustering
  - Hierarchical
- Topic Modeling

Clustering: Definition

- (Document) clustering is the process of grouping a set of documents into clusters of similar documents.
- Documents within a cluster should be similar.
- Documents from different clusters should be dissimilar.
- Clustering is the most common form of unsupervised learning.
- Unsupervised = there are no labeled or annotated data.

Classification vs. Clustering

- Classification: **supervised** learning
- Clustering: **unsupervised** learning
- Classification: Classes are human-defined and part of the input to the learning algorithm.
- Clustering: Clusters are inferred from the data without human input.
  - However, there are many ways of influencing the outcome of clustering: number of clusters, similarity measure, representation of documents, . . .

High-level Comparison

Classification

- **Advantage**:
  - Classification “rules” are identified automatically, which is useful for large document sets.
  - You know what each output means.
- **Disadvantages:**
  - You must assign documents to categories before generating the rules.
  - Rules may not be as specific or accurate as the ones you would write yourself.

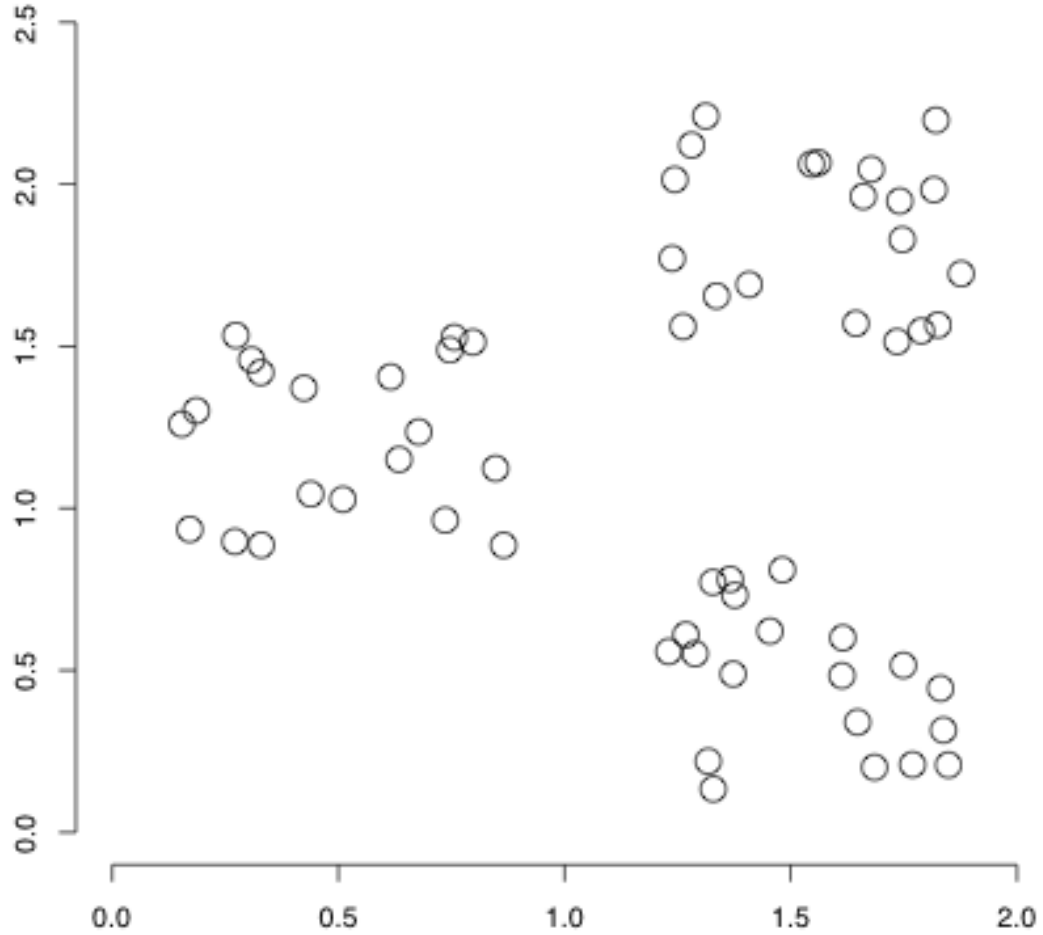
Clustering

- **Advantages:**
  - You don't need to provide either the classification rules or the sample documents as a training set.
  - Helps to discover patterns and content similarities in your document set that you might overlook.
- **Disadvantages:**
  - Clustering might result in unexpected groupings, since the clustering operation is not user-defined, but based on an internal algorithm.
  - You do not see the rules that create the clusters.

Two ways to combine these depending on scenarios:

1. When you do not have a clear idea of rules or classifications: Use unsupervised classification to provide an initial set of categories, and to subsequently build on these through supervised classification.
2. When you have a lot of data but only a small subset of it is labeled, and the rest is unlabeled. Employ formulations which perform clustering keeping in mind the labeled examples which *guide* which documents should be grouped together and which should not. This is called *semi-supervised learning.*

Data set with clear cluster structure



Finding the cluster structure in this example

Goals for clustering

- General goal: put related docs in the same cluster, put unrelated docs in different clusters.
  - How do we formalize this?
- The number of clusters should be appropriate for the data set we are clustering.
  - Assume the number of clusters *K* is given.
  - *Semi*-automatic methods for determining *K*
- Secondary goals in clustering
  - Avoid very small and very large clusters
  - Define clusters that are easy to explain to the user
  - Many others . . .

Flat vs. Hierarchical clustering

- Flat algorithms
  - Usually start with a random (partial) partitioning of docs into groups
  - Refine iteratively
  - Main algorithm: *K*-means
- Hierarchical algorithms
  - Create a hierarchy
  - Bottom-up, agglomerative
  - Top-down, divisive

Hard vs. Soft clustering

- Hard clustering: Each document belongs to exactly one cluster.
  - More common and easier to do

- Soft clustering: A document can belong to more than one cluster.
  - Makes more sense for applications like creating browsable hierarchies
  - You may want to put sneakers in two clusters:
    - sports apparel
    - shoes
  - You can only do that with a soft clustering approach.

Outline

- Document Representation
  - Similarity measures
- Classification
- Clustering
  - Flat Clustering
  - Hierarchical
- Topic Modeling

Flat algorithms

- Flat algorithms compute a partition of *N* documents into a set of *K* clusters.
- Given: a set of documents and the number *K*
- Find: a partition into *K* clusters that optimizes the chosen partitioning criterion
- Global optimization: exhaustively enumerate partitions, pick optimal one
  - Not tractable
- Effective heuristic method: *K*-means algorithm

K-means

- Perhaps the best known clustering algorithm
- Simple, works well in many cases

- Each cluster in *K*-means is defined by a centroid.
- Objective/partitioning criterion: minimize the average squared difference from the centroid

$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x}$$

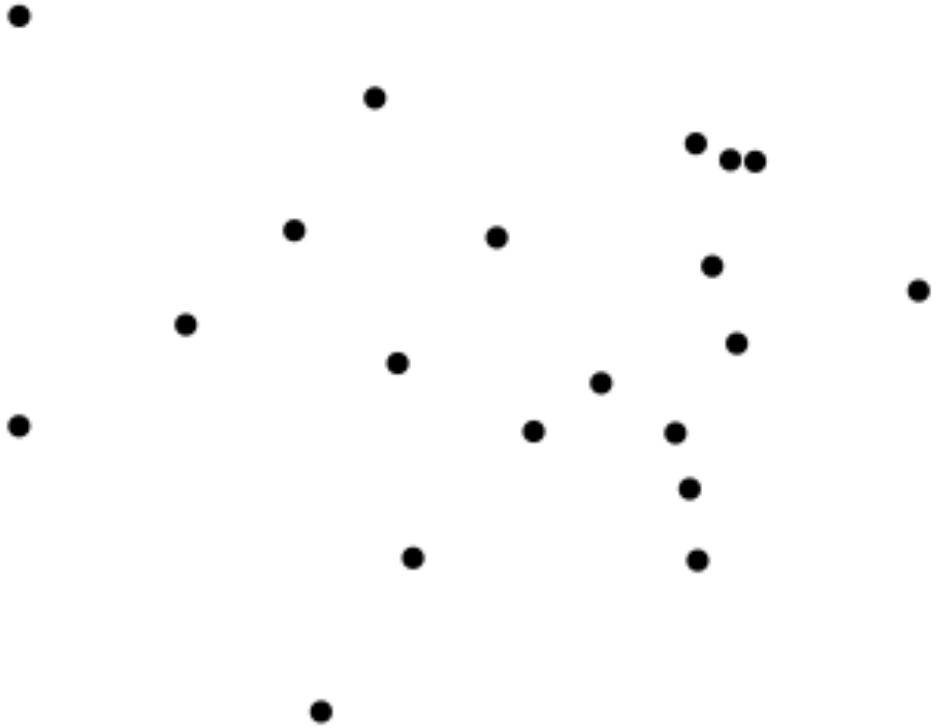
where we use ω to denote a cluster.

- We try to find the minimum average squared difference by iterating two steps:
  - reassignment: assign each vector to its closest centroid
  - recomputation: recompute each centroid as the average of the vectors that were assigned to it in reassignment

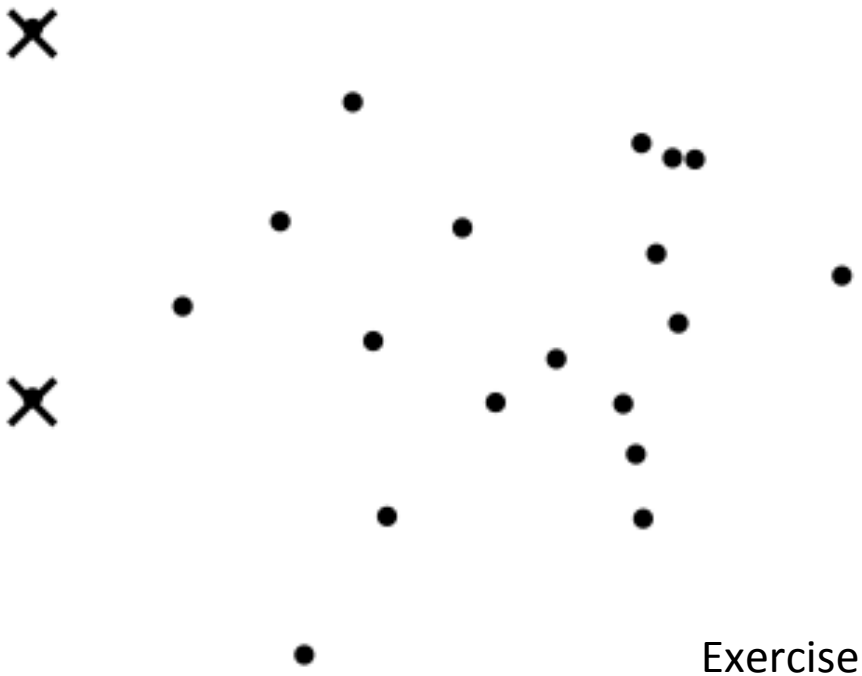
K-means algorithm

K-MEANS$(\{\vec{x}_1, \ldots, \vec{x}_N\}, K)$
1 $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS$(\{\vec{x}_1, \ldots, \vec{x}_N\}, K)$
2 **for** $k \leftarrow 1$ **to** K
3 **do** $\vec{\mu}_k \leftarrow \vec{s}_k$
4 **while** stopping criterion has not been met
5 **do for** $k \leftarrow 1$ **to** K
6 **do** $\omega_k \leftarrow \{\}$
7 **for** $n \leftarrow 1$ **to** N
8 **do** $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$
9 $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ *(reassignment of vectors)*
10 **for** $k \leftarrow 1$ **to** K
11 **do** $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ *(recomputation of centroids)*
12 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$

Worked Example: Set of to be clustered

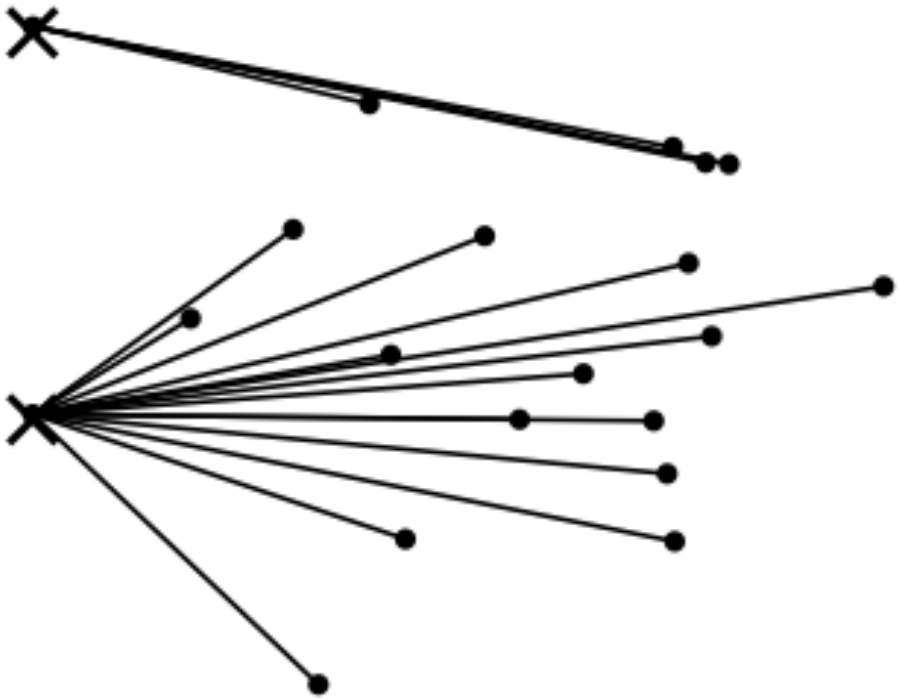


Worked Example: Random selection of initial centroids

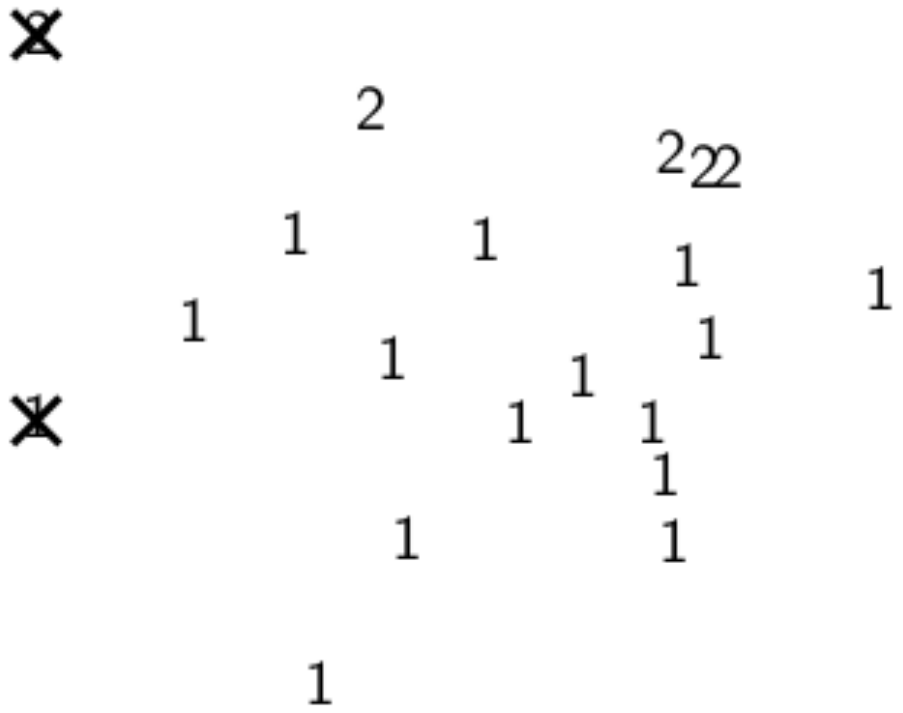


Exercise: (i) Guess what the optimal clustering into two clusters is in this case; (ii) compute the centroids of the clusters

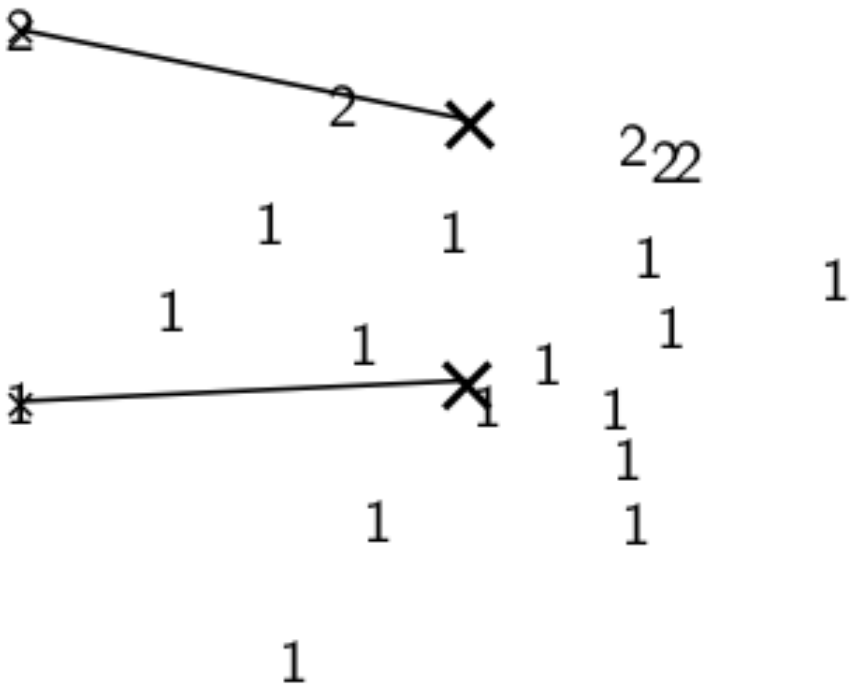
Worked Example: Assign points to closest center



Worked Example: Assignment

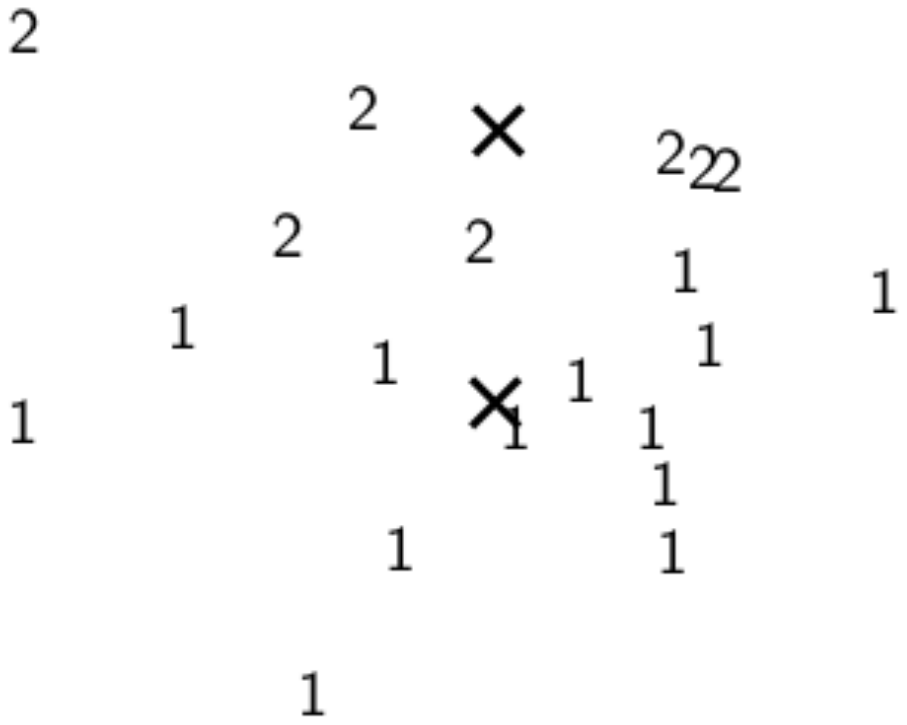


Worked Example: Recompute cluster centroids

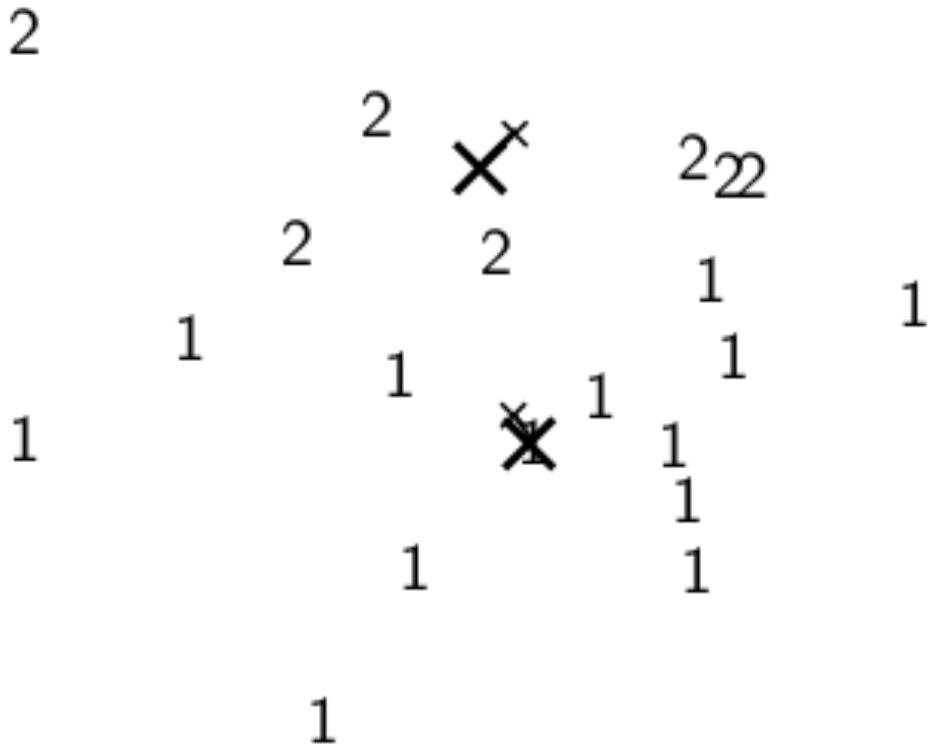


Worked Example: Assign points to closest centroid

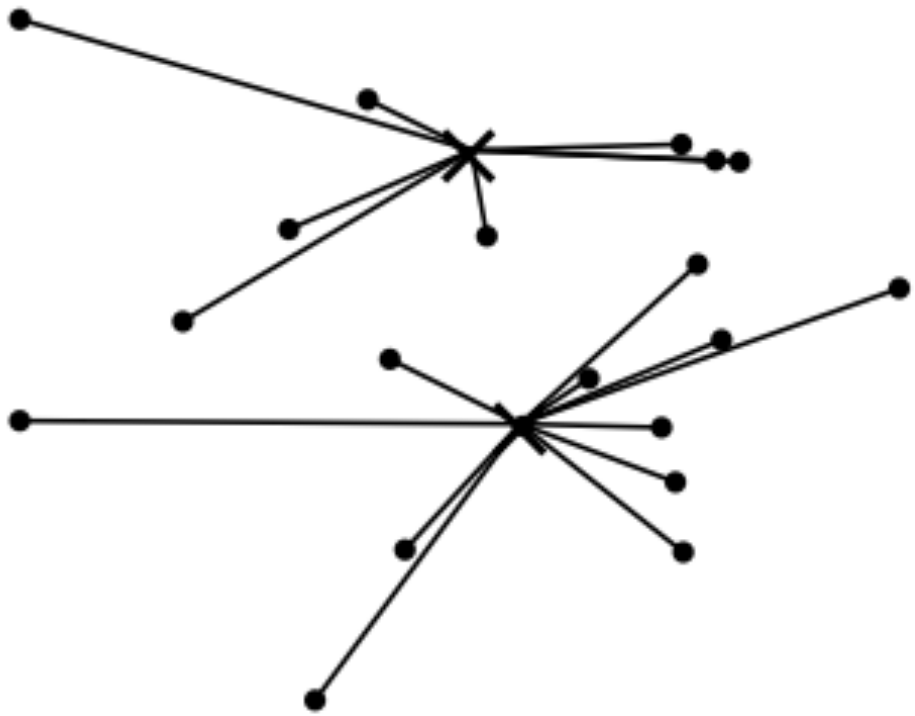
Worked Example: Assignment



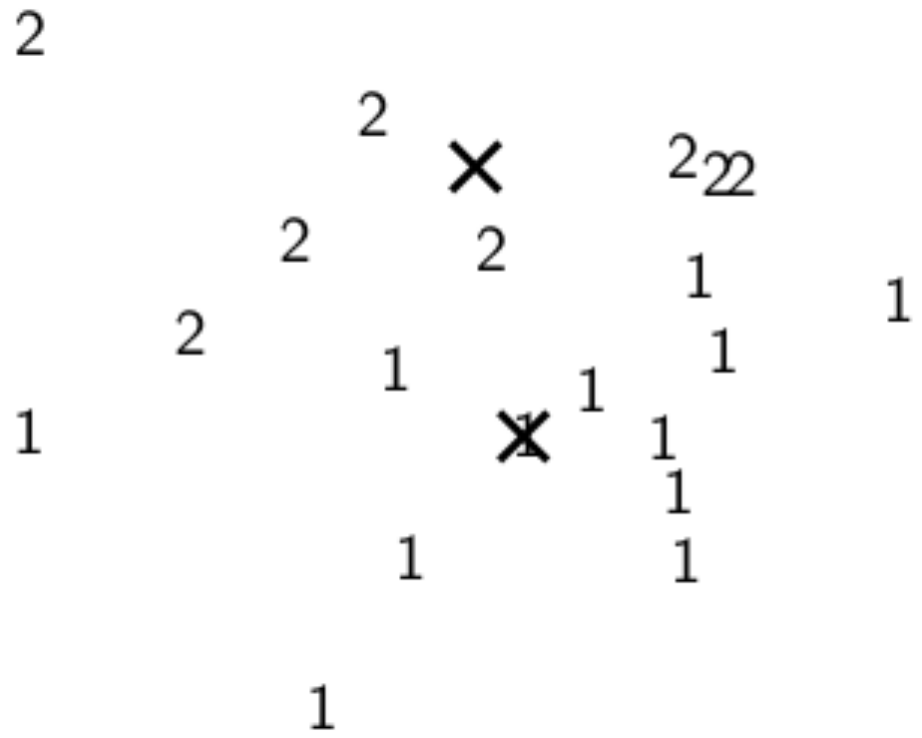
Worked Example: Recompute cluster centroids



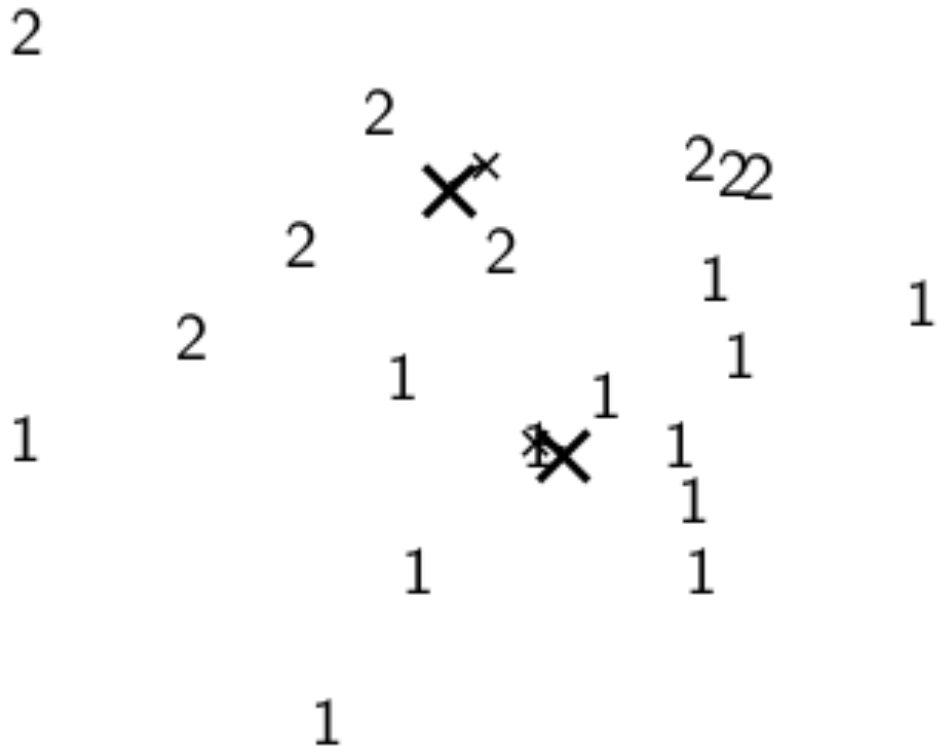
Worked Example: Assign points to closest centroid



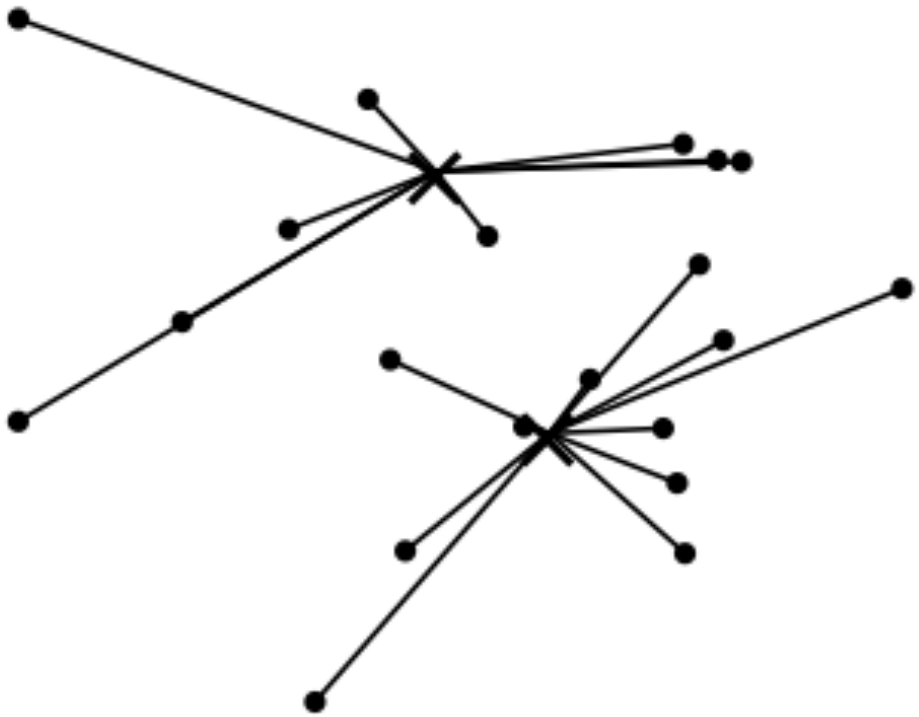
Worked Example: Assignment



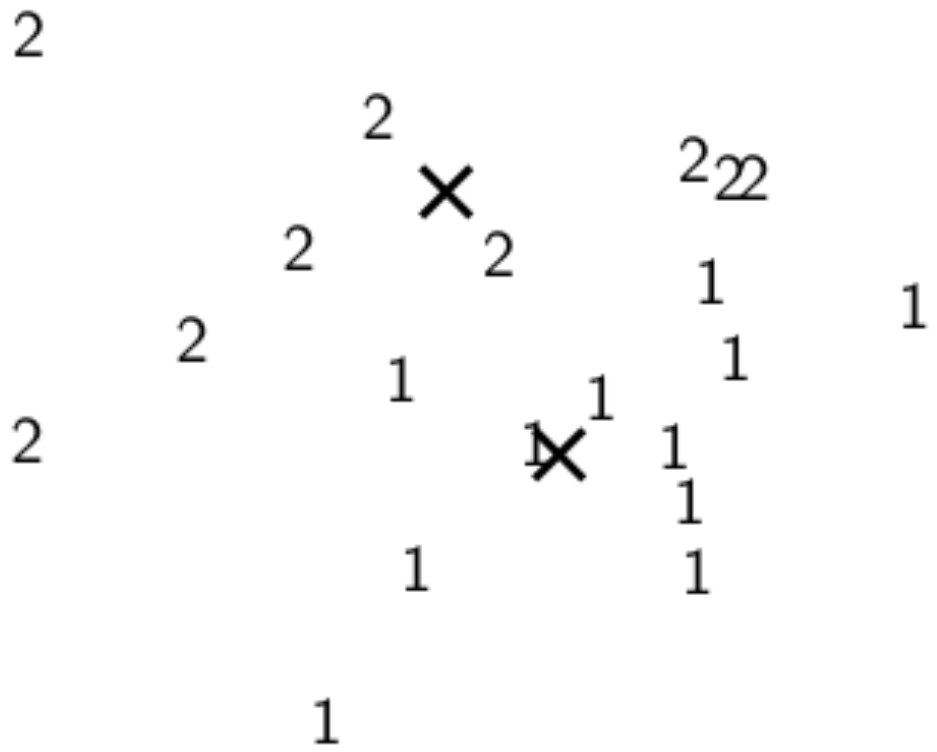
Worked Example: Recompute cluster centroids



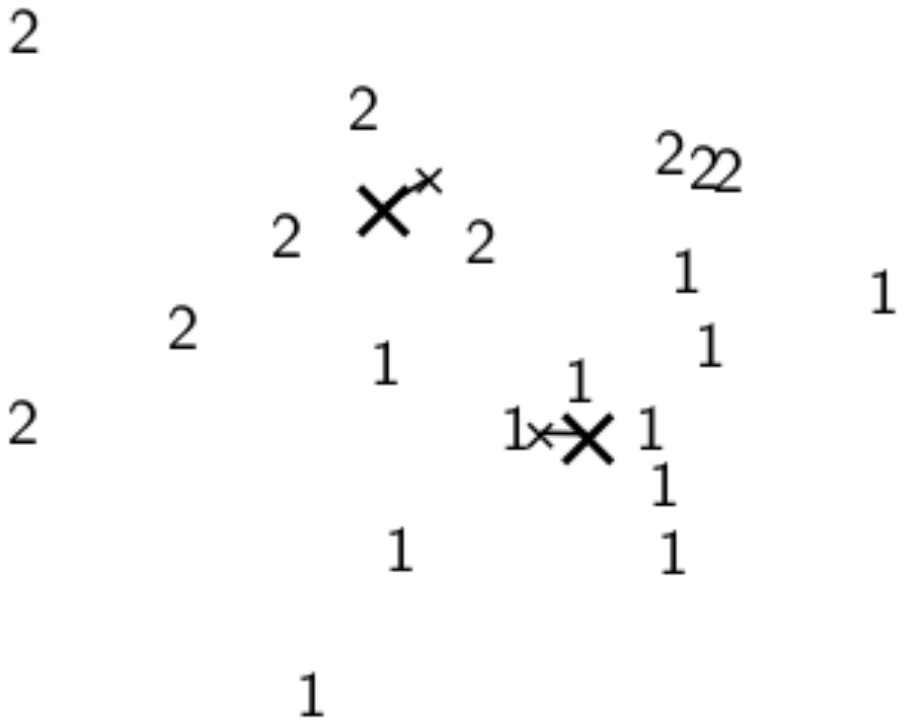
Worked Example: Assign points to closest centroid



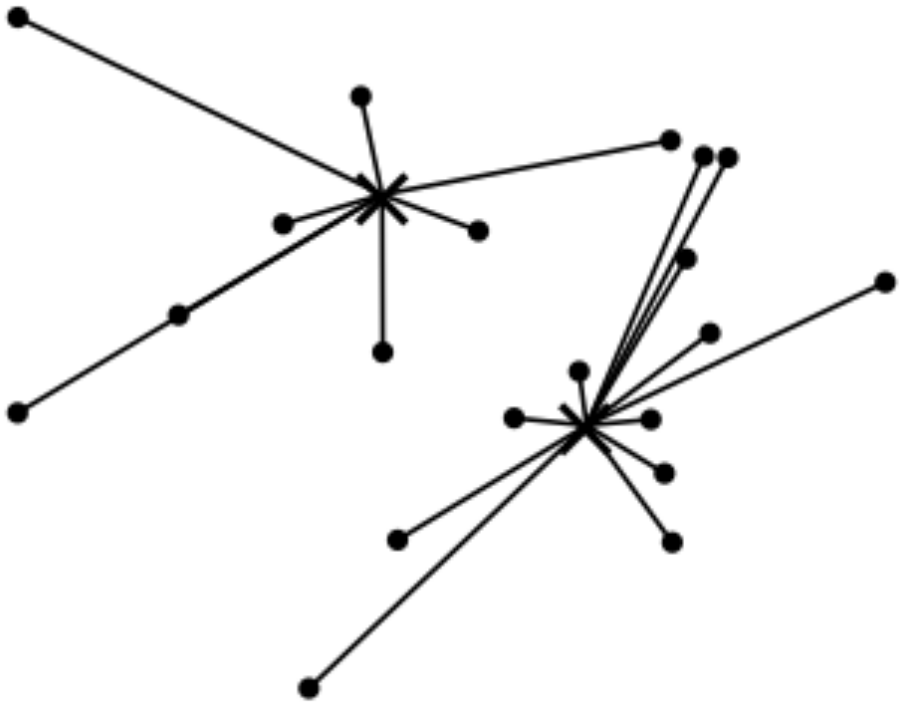
Worked Example: Assignment



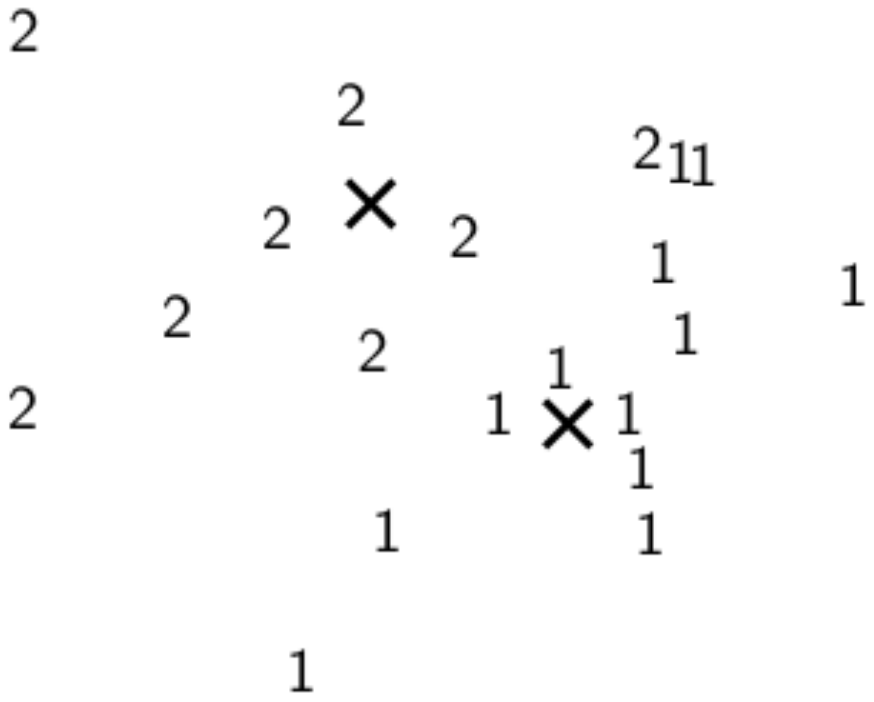
Worked Example: Recompute cluster centroids



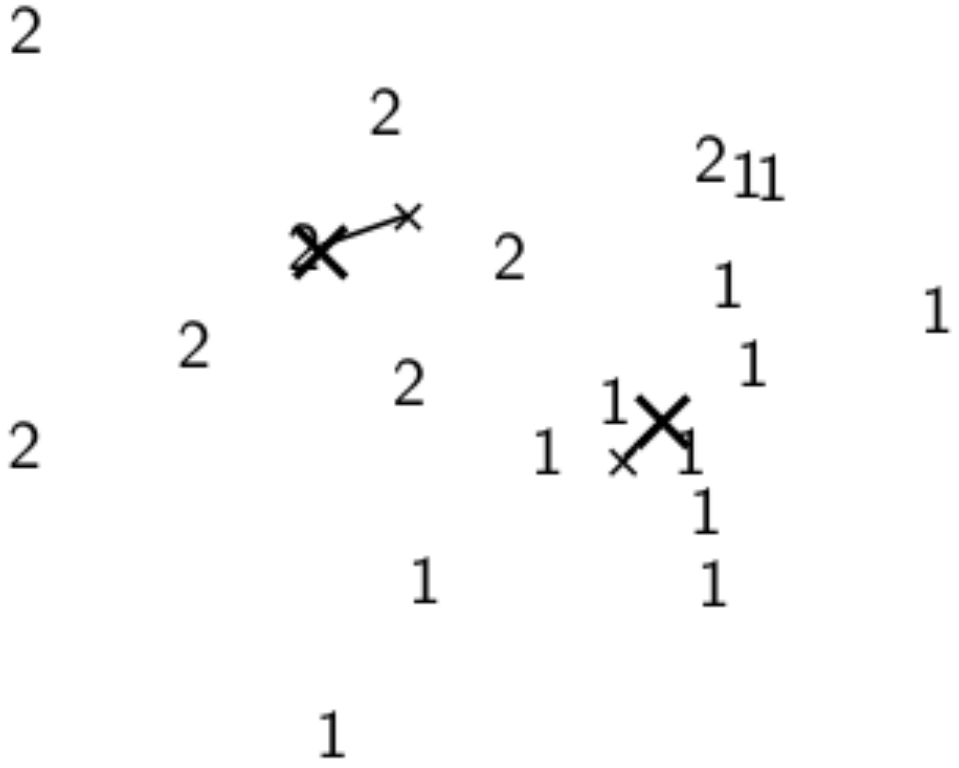
Worked Example: Assign points to closest centroid



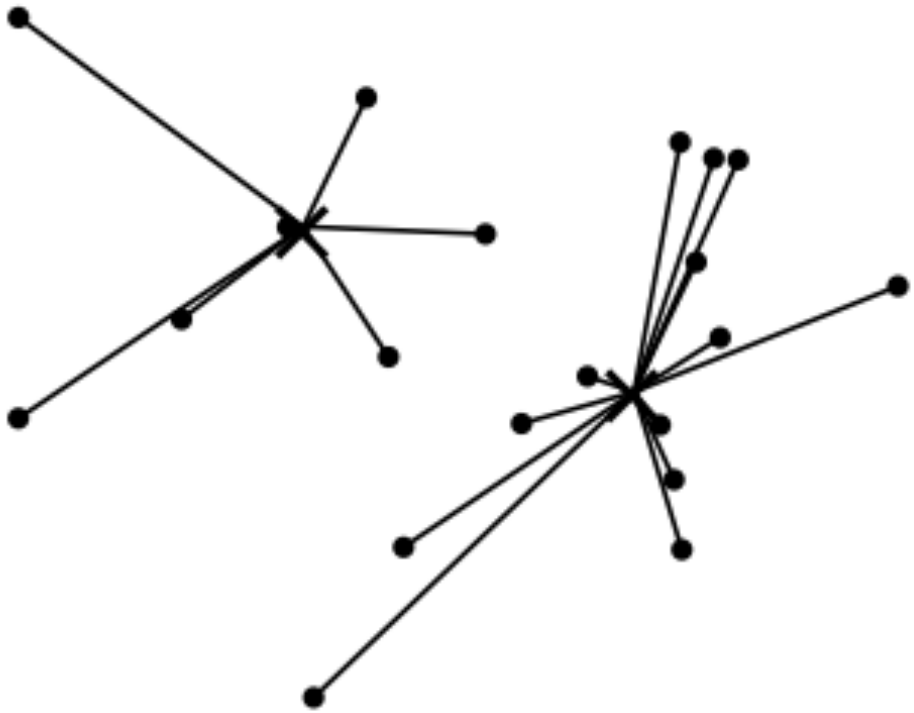
Worked Example: Assignment



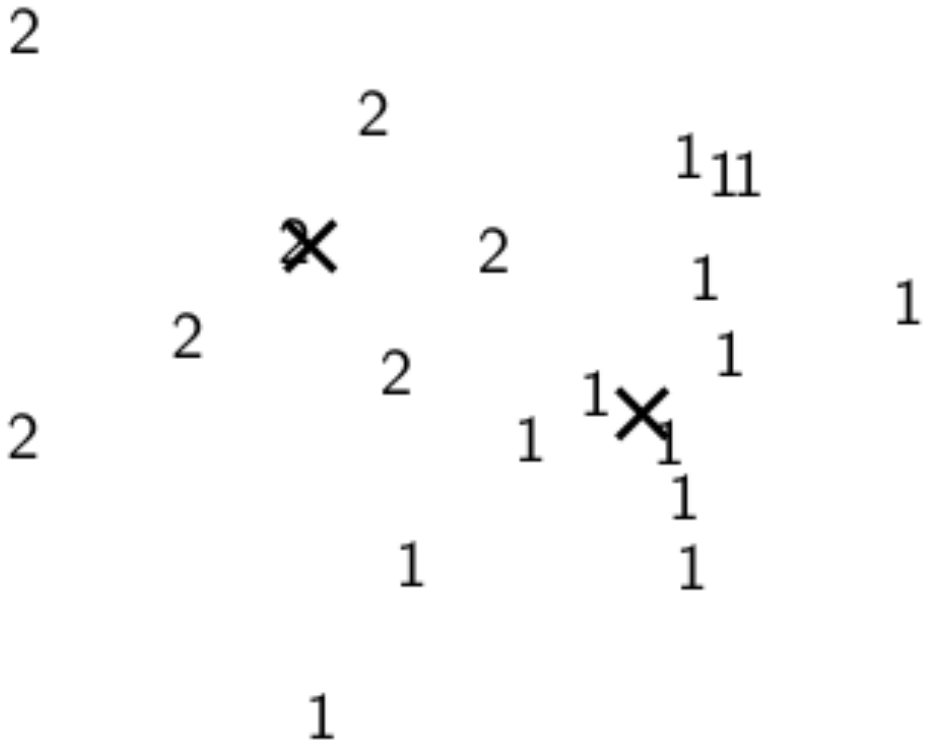
Worked Example: Recompute cluster centroids



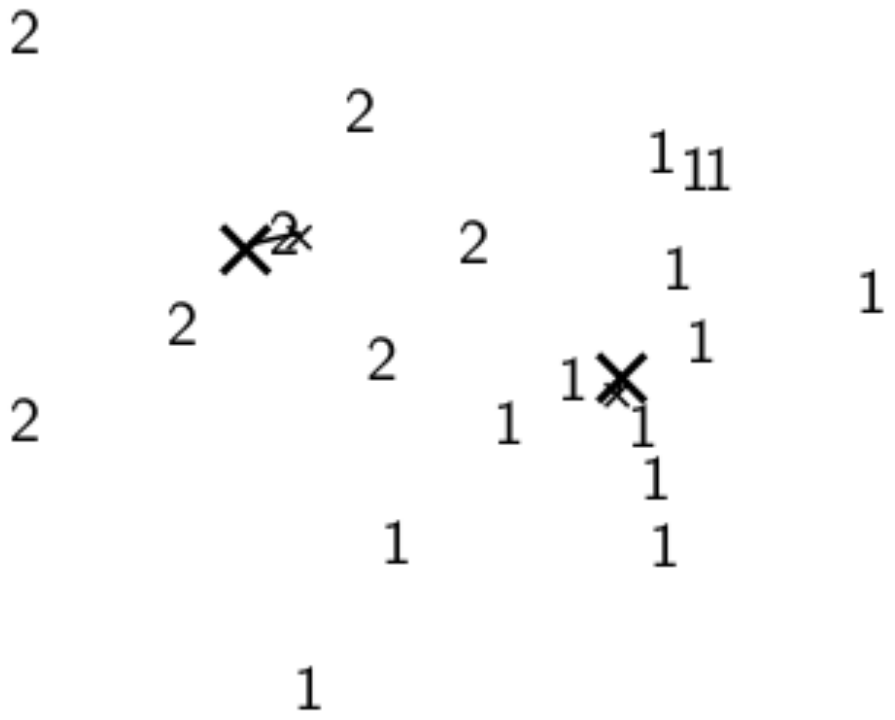
Worked Example: Assign points to closest centroid



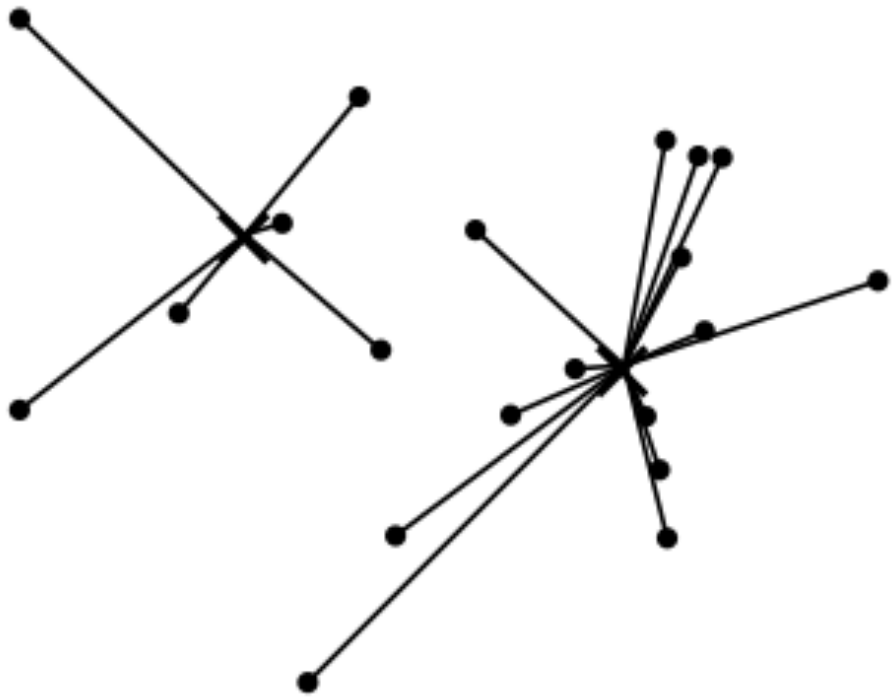
Worked Example: Assignment



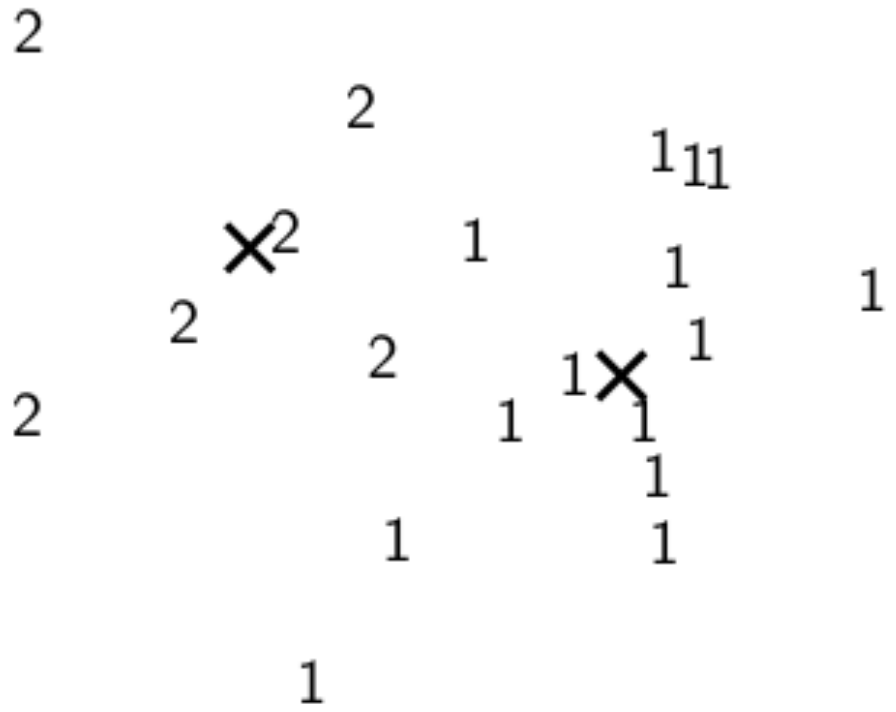
Worked Example: Recompute cluster centroids



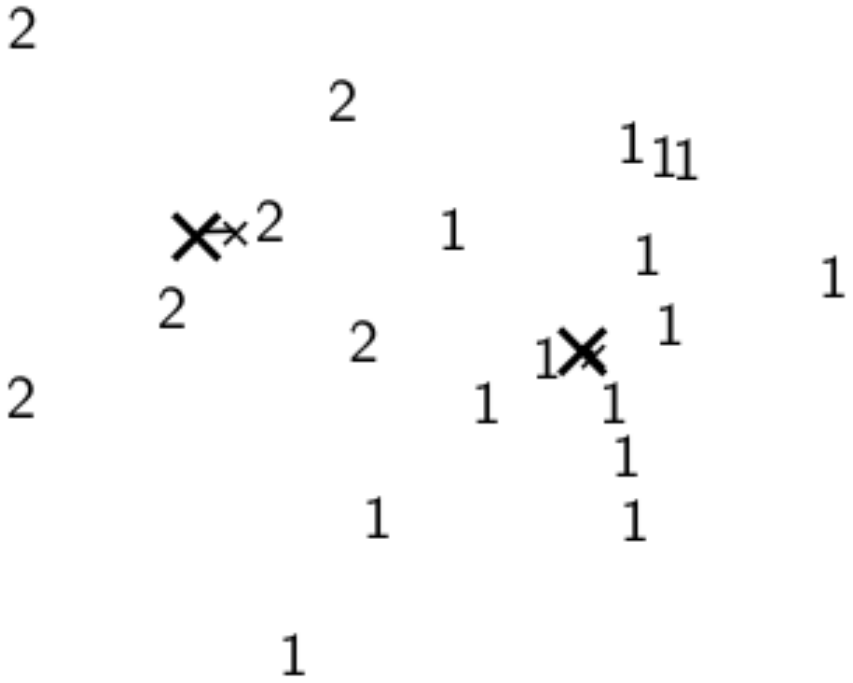
Worked Example: Assign points to closest centroid



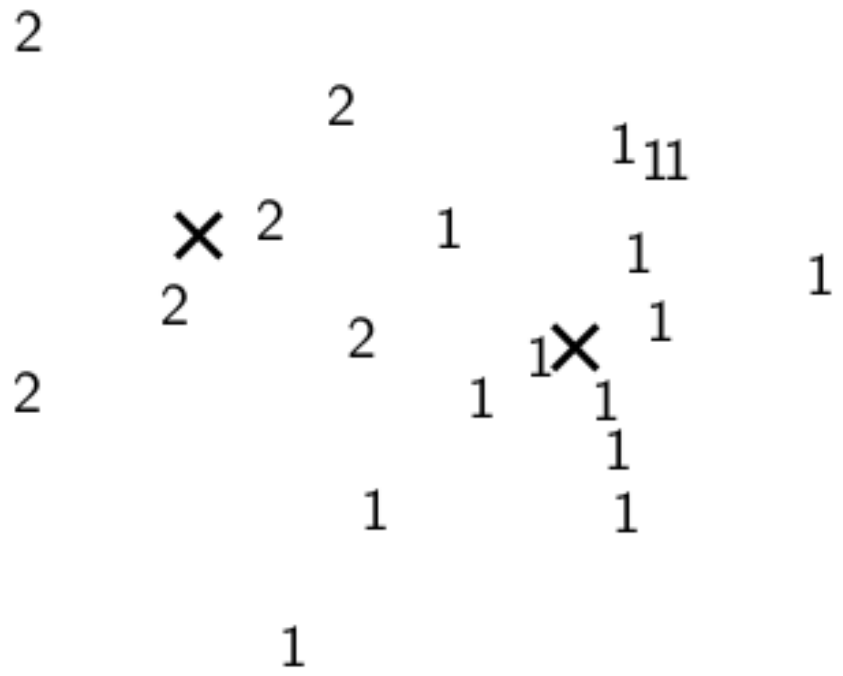
Worked Example: Assignment



Worked Example: Recompute cluster caentroids



Worked Ex.: Centroids and assignments after convergence



Initialization of *K*-means

- Random seed selection is just one of many ways K-means can be initialized.
- Random seed selection is not very robust: It's easy to get a suboptimal clustering.
- Better ways of computing initial centroids:
  - Select seeds not randomly, but using some heuristic (e.g., filter out outliers or find a set of seeds that has "good coverage" of the document space)
  - Use hierarchical clustering to find good seeds
  - Select *i* (e.g., *i* = 10) different random sets of seeds, do a *K*-means clustering for each, select the clustering with lowest RSS

Time complexity of *K*-means

- Computing one distance of two vectors is $O(M)$.
- Reassignment step: $O(KNM)$ (we need to compute KN document-centroid distances)
- Recomputation step: $O(NM)$ (we need to add each of the document's $< M$ values to one of the centroids)
- Assume number of iterations bounded by I
- Overall complexity: $O(IKNM)$ – linear in all important dimensions
- However: This is not a real worst-case analysis.
- In pathological cases, complexity can be worse than linear.

What is a good clustering?

- Internal criteria
  - Example of an internal criterion: RSS in *K*-means
- But an internal criterion often does not evaluate the actual utility of a clustering in the application.
- Alternative: External criteria
  - Evaluate with respect to a human-defined classification

External criteria for clustering quality

- Based on a gold standard data set, e.g., the class labels we would use for the evaluation of classification
- Goal: Clustering should reproduce the classes in the gold standard
- First measure for how well we were able to reproduce the classes: purity

External criterion: Purity

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

- $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ is the set of clusters and $C = \{c_1, c_2, \ldots, c_J\}$ is the set of classes.
- For each cluster ω_k : find class c_j with most members n_{kj} in ω_k
- Sum all n_{kj} and divide by total number of points

Two other external evaluation measures

- Two other measures
- Normalized mutual information (NMI)
  - How much information does the clustering contain about the classification?
  - Singleton clusters (number of clusters = number of docs) have maximum MI
  - Therefore: normalize by entropy of clusters and classes
- F measure
  - "precision" and "recall" can be weighted

How many clusters?

- Number of clusters *K* is given in many applications.
  - There may be an external constraint on *K*.

- What if there is no external constraint? Is there a "right" number of clusters?
- One way to go: define an optimization criterion
  - Given docs, find *K* for which the optimum is reached.
  - What optimization criterion can we use?
  - We can't use RSS or average squared distance from centroid as criterion: always chooses $K = N$ clusters.

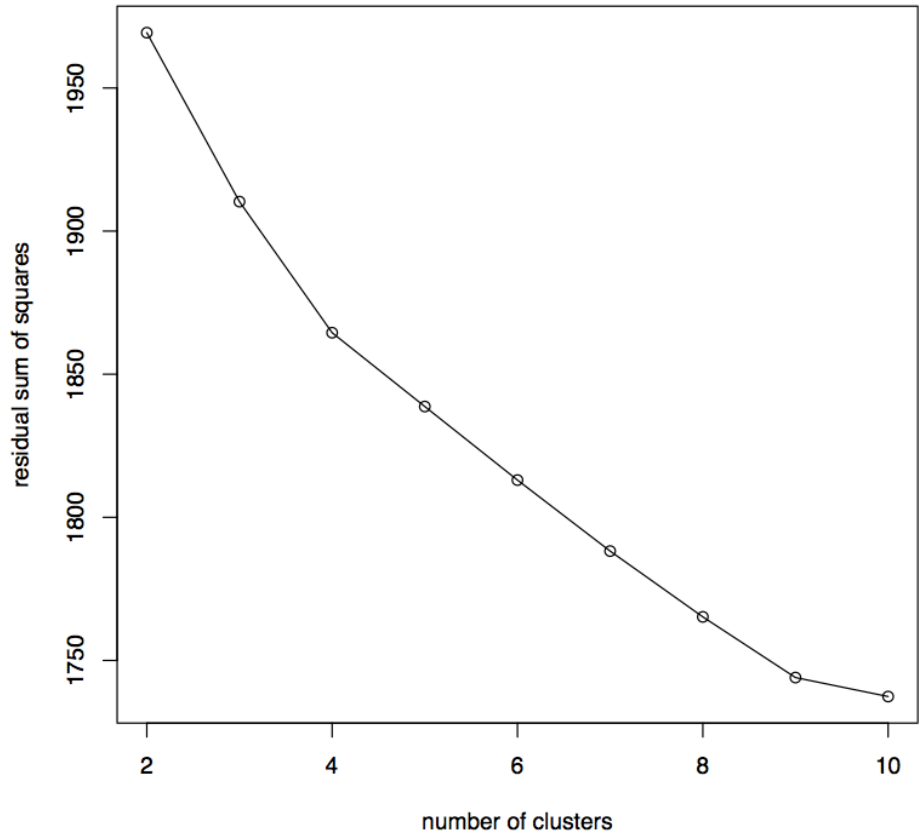
Simple objective function for *K* (1)

- Basic idea:
  - Start with 1 cluster (*K* = 1)
  - Keep adding clusters (= keep increasing *K*)
  - Add a penalty for each new cluster
- Trade off cluster penalties against average squared distance from centroid
- Choose the value of *K* with the best tradeoff

Simple objective function for *K* (2)

- Given a clustering, define the cost for a document as (squared) distance to centroid
- Define total distortion RSS(K) as sum of all individual document costs (corresponds to average distance)
- Then: penalize each cluster with a cost λ
- Thus for a clustering with *K* clusters, total cluster penalty is *K*λ
- Define the total cost of a clustering as distortion plus total cluster penalty: RSS(K) + *K*λ
- Select K that minimizes (RSS(K) + *K*λ)
- Still need to determine good value for λ . . .

Finding the “knee” in the curve



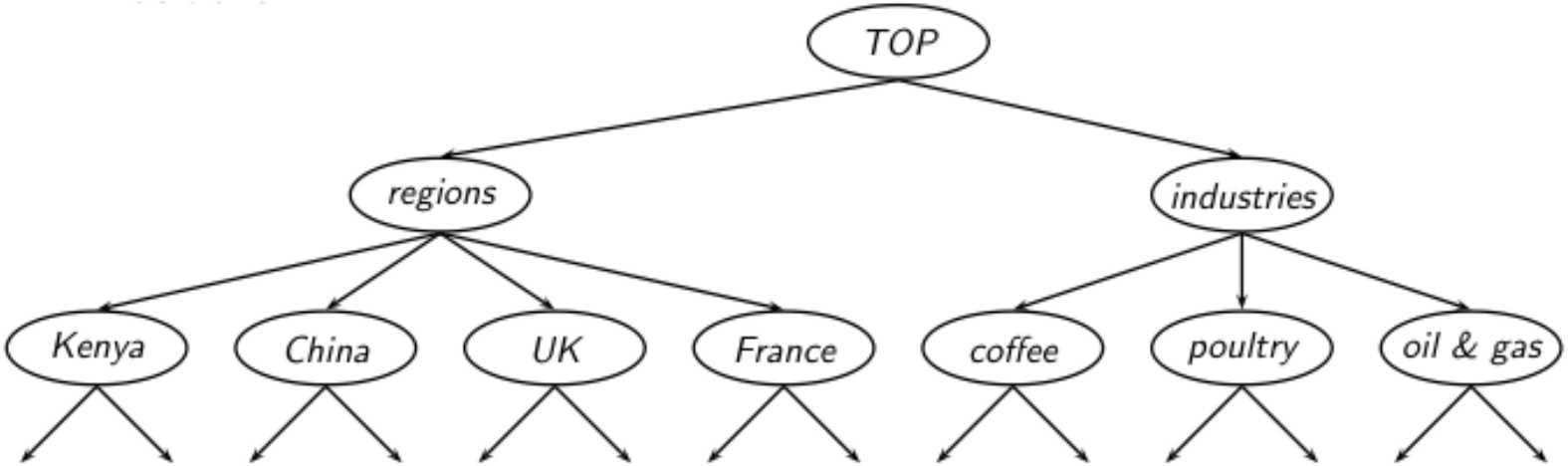
Pick the number of clusters where curve “flattens”. Here: 4 or 9.

Outline

- Document Representation
  - Similarity measures
- Classification
- Clustering
  - Flat Clustering
  - Hierarchical
- Topic Modeling

Hierarchical clustering

Our goal in hierarchical clustering is to create a hierarchy something like:



We want to create this hierarchy automatically. We can do this either top-down or bottom-up. A well known bottom-up method is hierarchical agglomerative clustering (HAC).

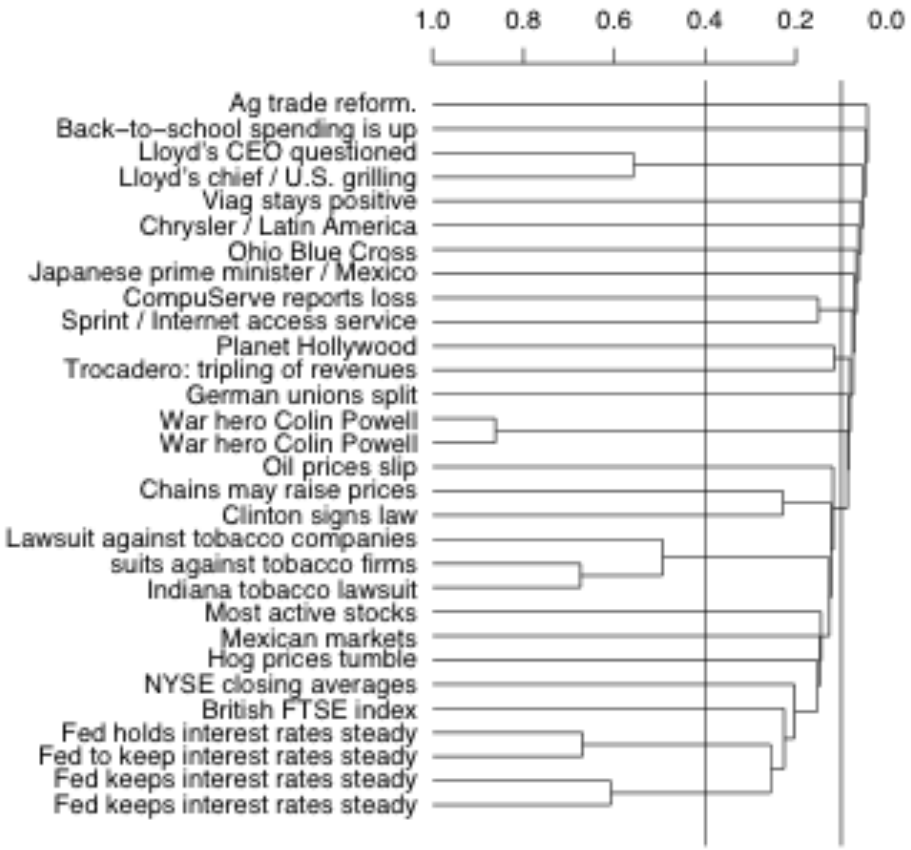
Hierarchical agglomerative clustering (HAC)

- HAC creates a hierachy in the form of a binary tree.
- Assumes a similarity measure for determining the similarity of two clusters.
- Up to now, our similarity measures were for documents.
- We will look at four different cluster similarity measures.

Hierarchical agglomerative clustering (HAC)

- Start with each document in a separate cluster
- Then repeatedly merge the two clusters that are most similar
- Until there is only one cluster
- The history of merging is a hierarchy in the form of a binary tree.
- The standard way of depicting this history is a dendrogram.

A dendogram



- The history of mergers can be read off from bottom to top.
- The horizontal line of each merger tells us what the similarity of the merger was.
- We can cut the dendrogram at a particular point (e.g., at 0.1 or 0.4) to get a flat clustering.

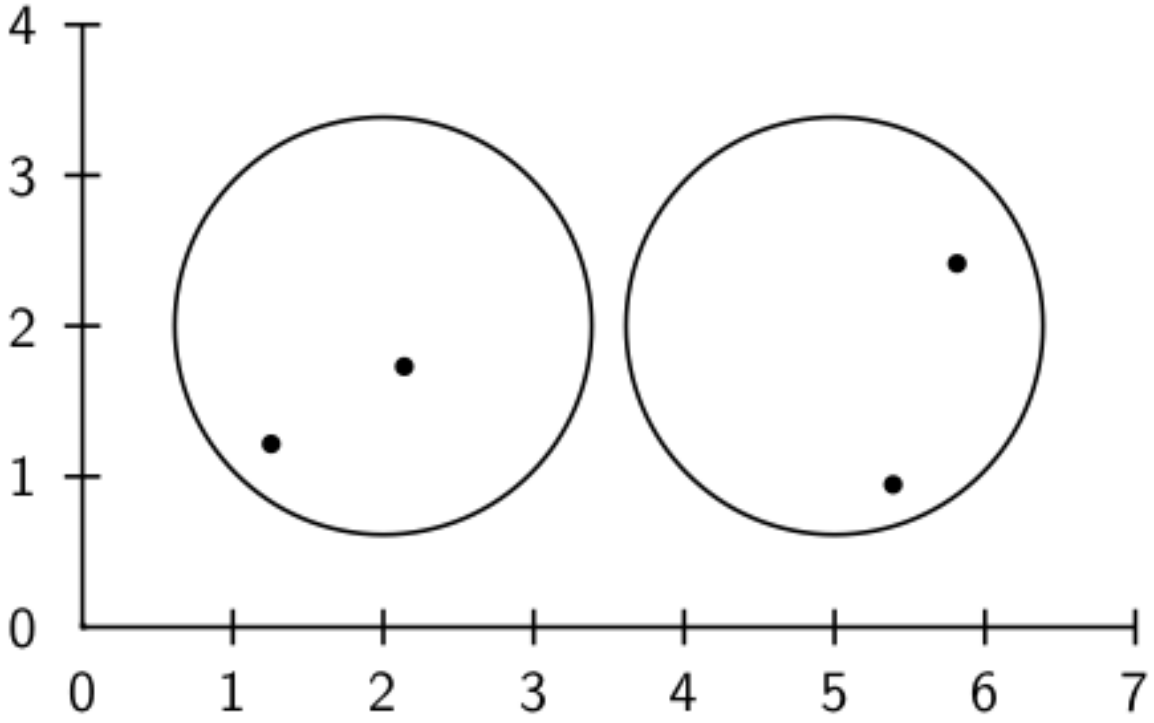
Computational complexity of the naive algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.
- Then, in each of N iterations:
  - We scan the $O(N \times N)$ similarities to find the maximum similarity.
  - We merge the two clusters with maximum similarity.
  - We compute the similarity of the new cluster with all other (surviving) clusters.
- There are $O(N)$ iterations, each performing a $O(N \times N)$ "scan" operation.
- Overall complexity is $O(N^3)$.

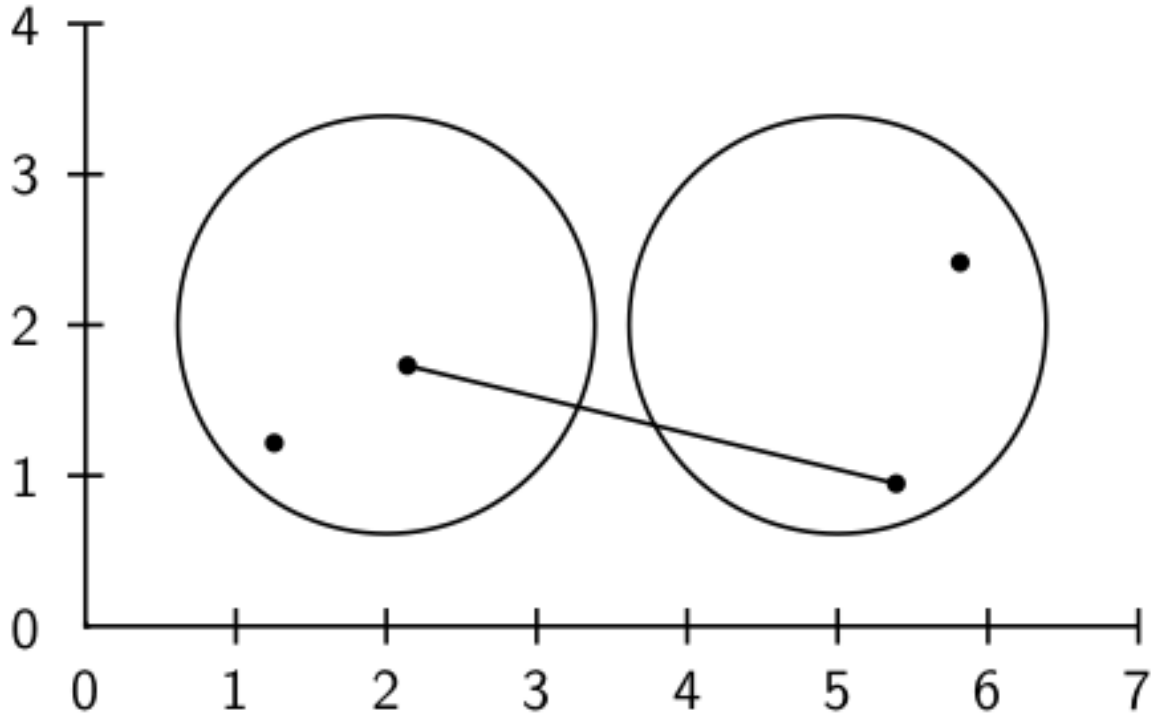
Key question: How to define cluster similarity

- Single-link: Maximum similarity
  - Maximum similarity of any two documents
- Complete-link: Minimum similarity
  - Minimum similarity of any two documents
- Centroid: Average "intersimilarity"
  - Average similarity of all document pairs (but excluding pairs of docs in the same cluster)
  - This is equivalent to the similarity of the centroids.
- Group-average: Average "intrasimilarity"
  - Average similary of all document pairs, including pairs of docs in the same cluster

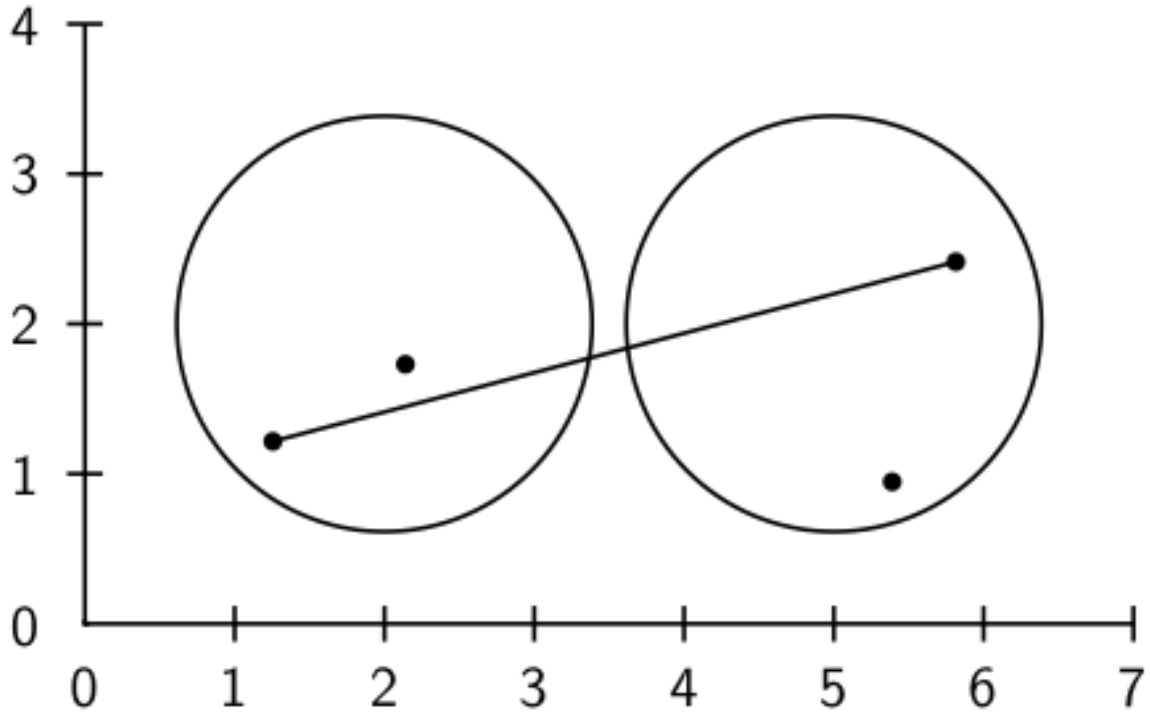
Cluster similarity: Example



Single-link: Maximum similarity

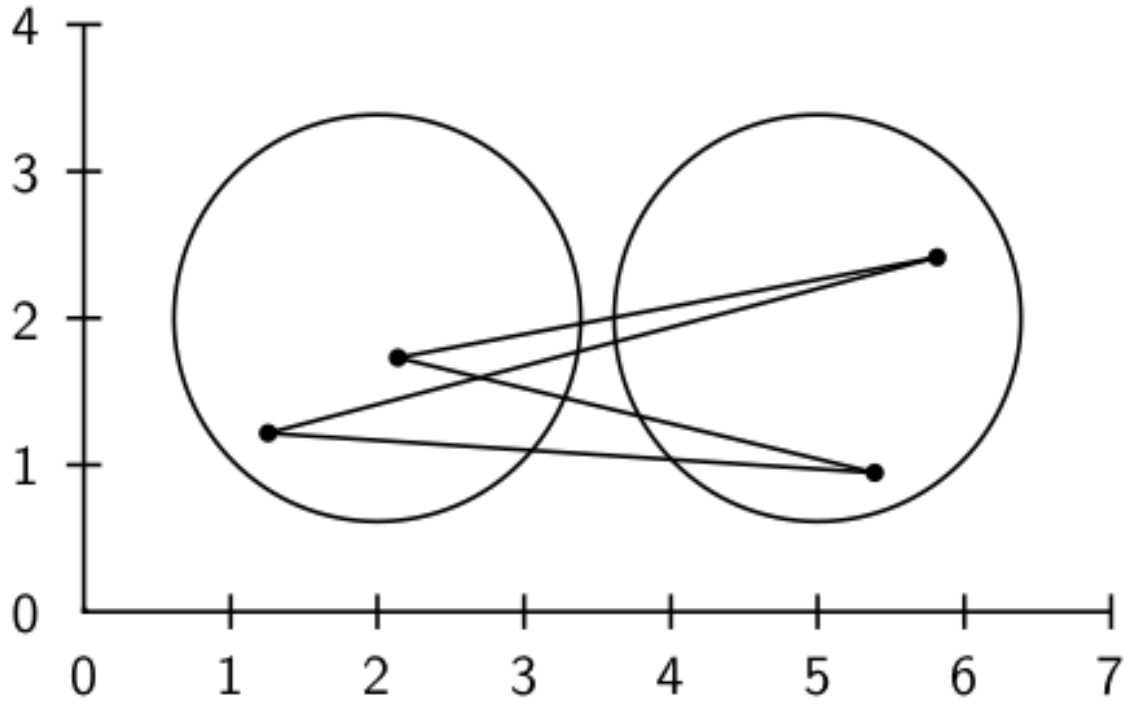


Complete-link: Minimum similarity



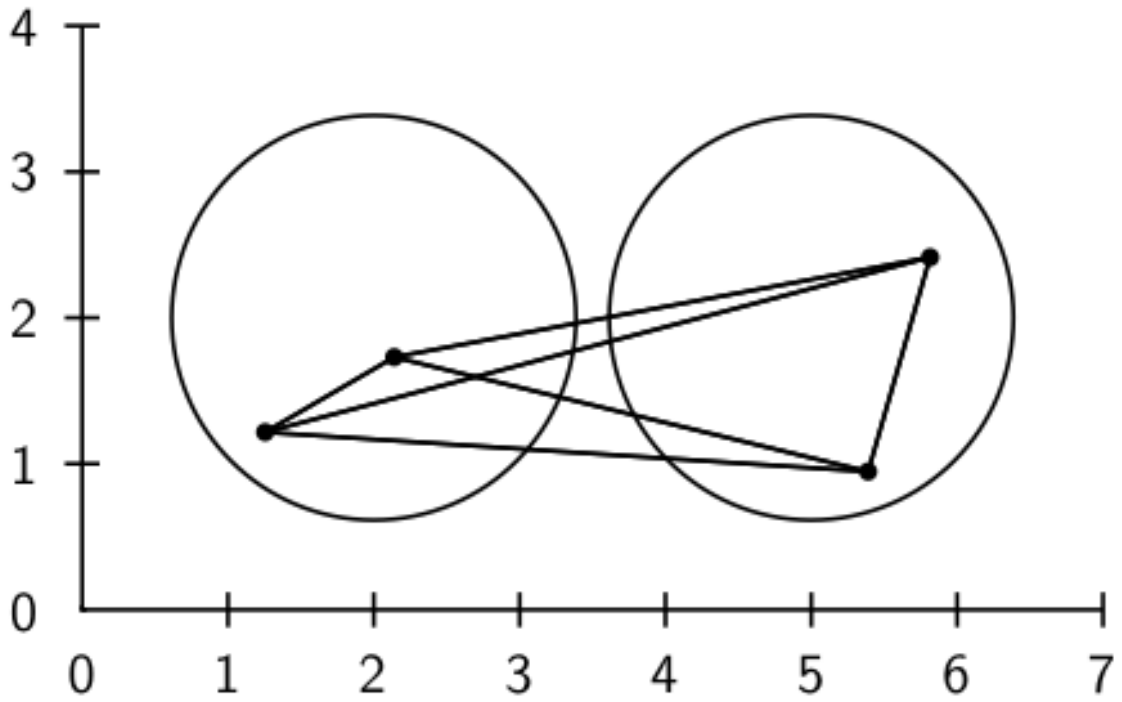
Centroid: Average intersimilarity

intersimilarity = similarity of two documents in different clusters



Group average: Average intrasimilarity

intrasimilarity = similarity of any pair, including cases where the two documents are in the same cluster



Flat or hierarchical clustering?

- For high efficiency, use flat clustering
- For deterministic results: HAC
- When a hierarchical structure is desired: hierarchical algorithm
- HAC also can be applied if *K* cannot be predetermined (can start without knowing *K*)

Major issue in clustering – labeling

- After a clustering algorithm finds a set of clusters: how can they be useful to the end user?
- We need a **descriptive yet concise** label for each cluster.
- How can we *automatically* find good labels for clusters?

Discriminative labeling

- To label cluster ω, compare ω with all other clusters
- Find terms or phrases that distinguish ω from the other clusters
- We can use any of the feature selection techniques: mutual information, χ^2 and frequency of the terms/phrases.

Non-discriminative labeling

- Select terms or phrases based solely on information from the cluster itself
- Terms with high weights in the centroid (if we are using a vector space model)
- Non-discriminative methods sometimes select frequent terms that do not distinguish clusters.
  - For example, MONDAY, TUESDAY, . . . in newspaper text

Outline

- Document Representation
  - Similarity measures
- Classification
- Clustering
  - Flat Clustering
  - Hierarchical
- Topic Modeling

Topic Modeling

1. Data are assumed to be observed from a generative probabilistic process that includes hidden variables.

- In text, the hidden variables are the thematic structure.

2. Infer the hidden structure using posterior inference

- What are the topics that describe this collection?

3. Situate new data into the estimated model.

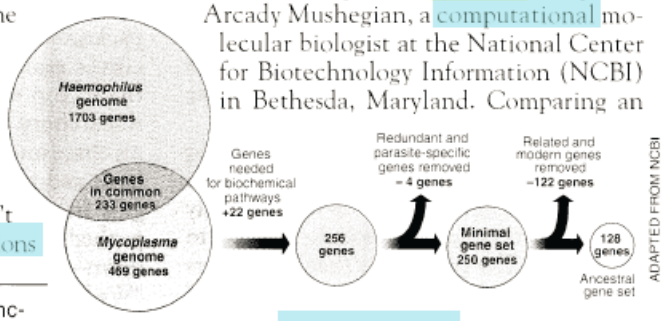
- How does a new document fit into the topic structure?

Latent Dirichlet allocation (LDA)

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

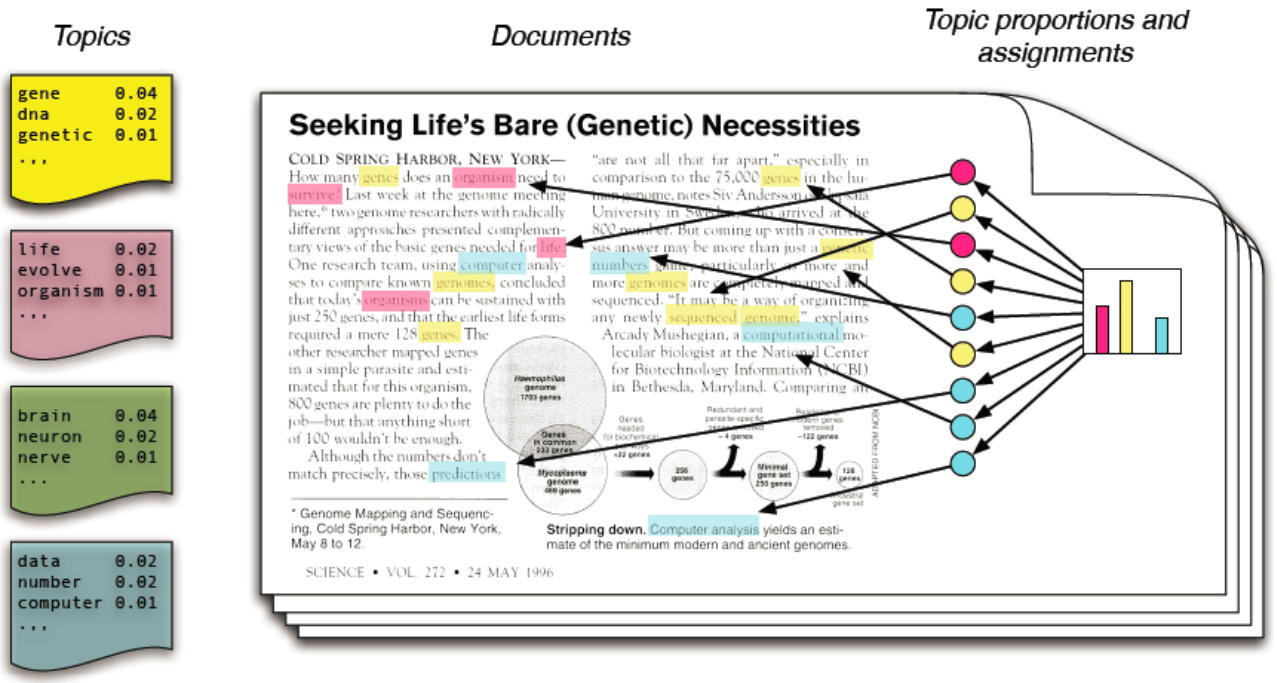
* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

Simple intuition: Documents exhibit multiple topics.

Blei, KDD 2011

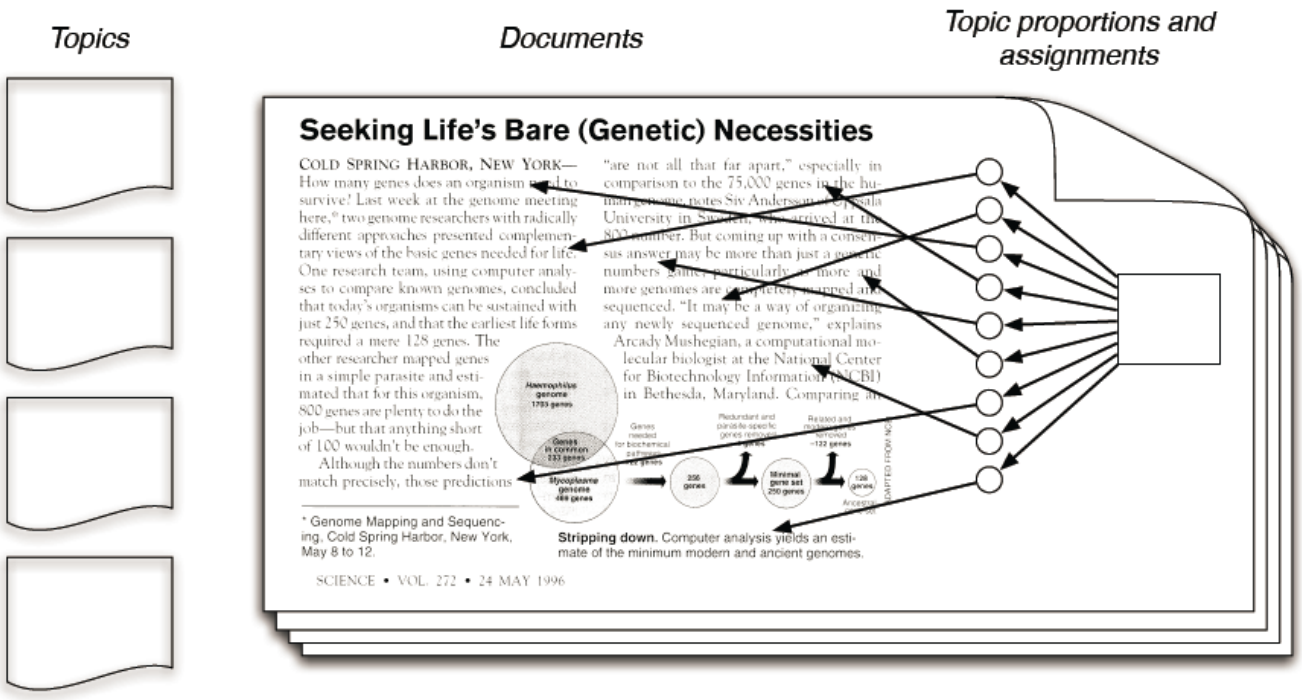
Generative model for LDA



Each **topic** is a distribution over words
Each **document** is a mixture of corpus-wide topics
Each **word** is drawn from one of those topics

Blei, KDD 2011

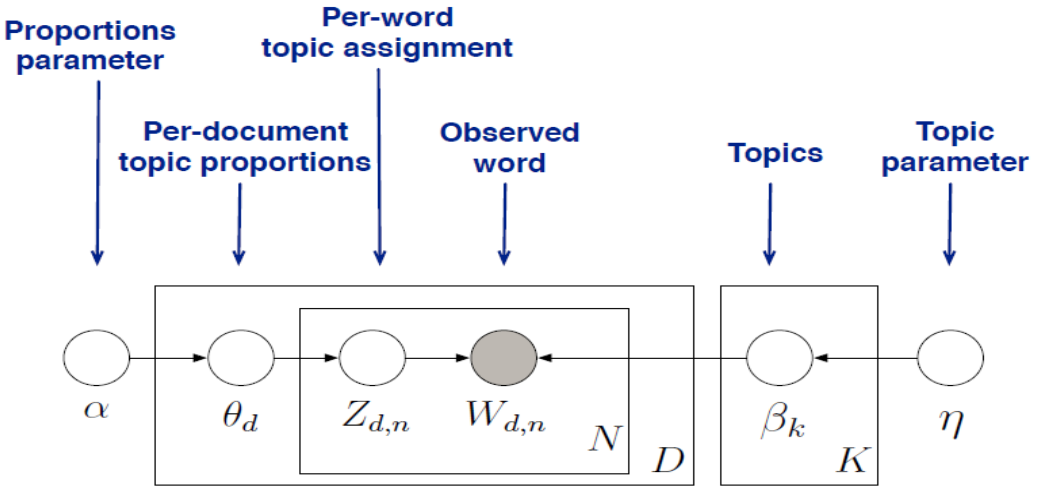
Generative model for LDA



In reality, we only observe the documents
The other structure are **hidden variables**

Blei, KDD 2011

Graphical Model for LDA



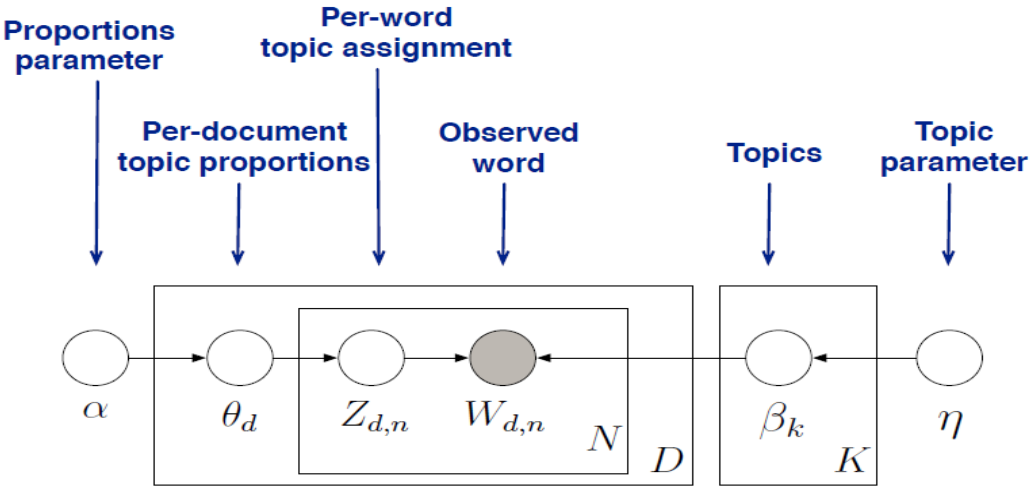
From a collection of documents, infer

- Per-word topic assignment Zd,n
- Per-document topic proportions d
- Per-corpus topic distributions k

Then use posterior expectations to perform the task at hand,
e.g., document similarity, exploration, ...

Blei, KDD 2011

Graphical Model for LDA



Approximate posterior inference algorithms

- Mean field variational methods (Blei et al., 2001, 2003)
- Expectation propagation (Minka and Lafferty, 2002)
- Collapsed Gibbs sampling (Griffiths and Steyvers, 2002)
- Collapsed variational inference (Teh et al., 2006)
- Online variational inference (Hoffman et al., 2010)

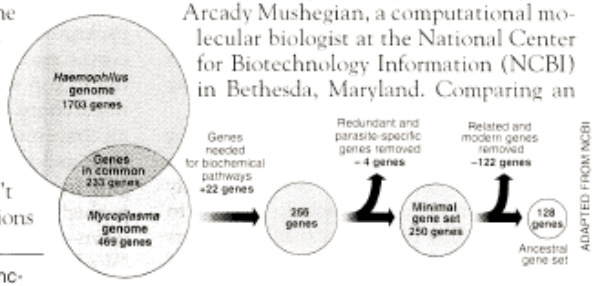
Also see Mukherjee and Blei (2009) and Asuncion et al. (2009).

Example

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

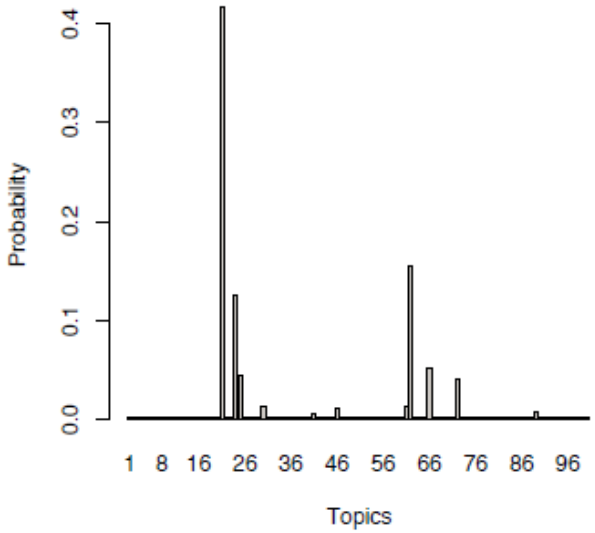
Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

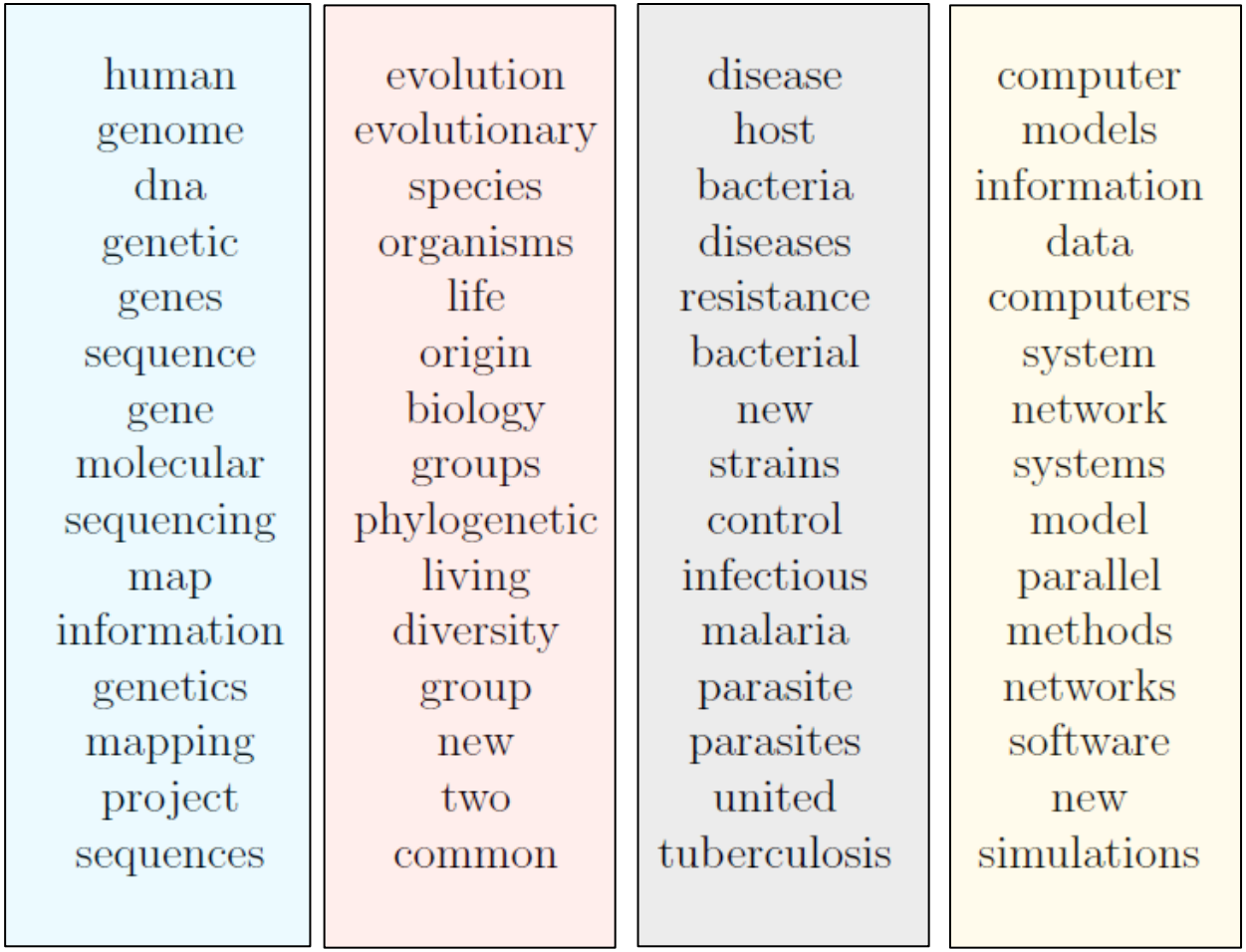
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996



Blei, KDD 2011

Some extracted topics



Tools

- Document similarity: Vector Space Model: Lucene/ Solr
- Classification: Naïve Bayes: Weka
- Clustering: Kmeans: Cluto/Carrot2
- Topic Modeling: LDA: Mallet

A Quick Review

STATE-OF-THE-ART

Duplicate Bug Report Detection

Between Nov-12 to Dec-12 –

- 5713 bugs were opened
- 3773 were marked duplicates

1. Detection of Duplicate Defect Reports Using Natural Language Processing. Runeson et al. ICSE 2007.
2. A approach to detecting duplicate bug reports using natural language and execution information. Wang et al. ICSE 2008
3. Duplicate bug report detection with a combination of information retrieval and topic modeling. Nguyen et al. ASE 2012

Detection of Duplicate Defect Reports using NLP – ICSE 2007

Technique:

- Calculate Cosine Similarity of bug report with other reports
  - Project name
  - Title
  - Description
- Give top 5, 10, 15 matches

Pre-Processing:

- Tokenization, stemming, stop word removal, project specific synonym dictionary, spellchecking

Detection of Duplicate Defect Reports using NLP – ICSE 2007

Experiment:

- Subject: Sony Ericsson Mobile Communications products. 10% bugs were marked as duplicates.
- Measurement criteria: Recall

Findings:

- Maximum recall of 39% with result size of 10, 42% with 15
- Very small improvements in recall because of – synonyms, spell checking, giving header double weight
- Recall rate best when bug reports only compared with bug reports opened in prior 50-60 days

An Approach to Detecting Duplicate Bug Reports using Natural Language and execution information – ICSE 2008

Hypothesis: detect similarity based on execution information and text similarity and combine to mark reports as duplicate

Intuition:

***Bug-260331:** After closing Firefox, the process is still running. Cannot reopen Firefox after that, unless the previous process is killed manually*

***Bug-239223:** (Ghostproc) – [Meta] firefox.exe doesn't always exit after closing all windows; session-specific data retained*

Very little text similarity

***Bug-244372:** "Document contains no data" message on continuation page of NY Times article*

***Bug-219232:** random "The Document contains no data." Alerts*

Very little execution information similarity

An Approach to Detecting Duplicate Bug Reports using Natural Language and execution information – ICSE 2008

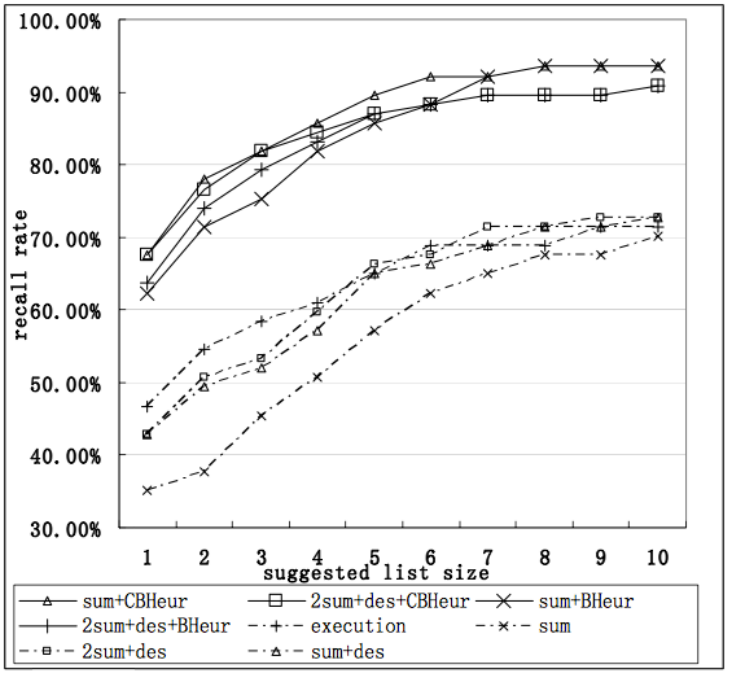
Approach:

- Calculate text similarity using Vector Space Model (NL-S)
- Calculate execution trace similarity by converting execution trace to text – retain method signature only (E-S)
- Combine above measures and rank
  - Variant 1: NL-S + E-S / 2
  - Variant 2: If NL-S > NL-Cutoff, then use NL-S only (NL dominant), if EL-S > EL-Cutoff, then use EL-S only (ES dominant)
    - Variant 2.1: Rank NS dominant bug reports higher
    - Variant 2.2: Rank ES dominant bug reports higher

An Approach to Detecting Duplicate Bug Reports using Natural Language and execution information – ICSE 2008

Experiment:

- Subject: Firefox (232 bug reports from Eclipse used for calibration). 1492 reports. 744 considered to be the search set. 748 as queries.
- Measurement criteria: Recall



Findings:

67 – 93% recall when using variant 2.1. Use title only for text similarity

Duplicate bug report detection with a combination of information retrieval and topic modeling. – ASE 2012

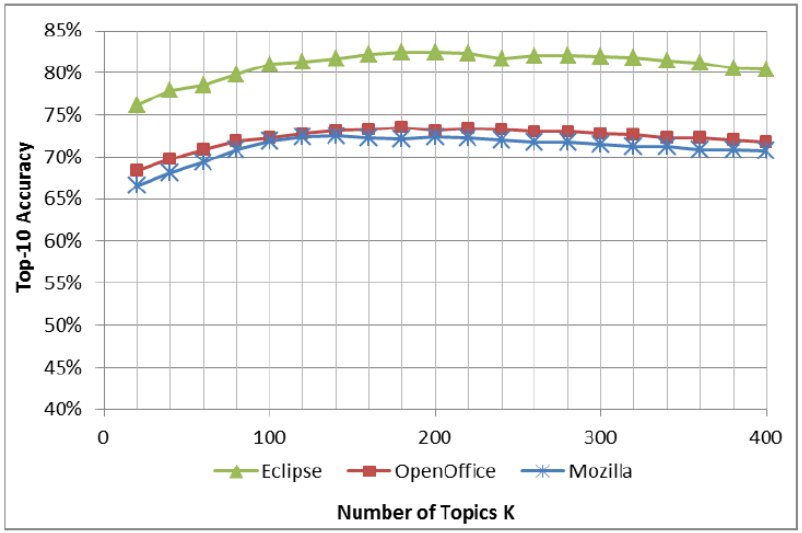
Idea:

- Model duplicate bug reports using historical data
  - Learn sets of different terms describing same technical issue
- Apply model to new bug reports and identify duplicate

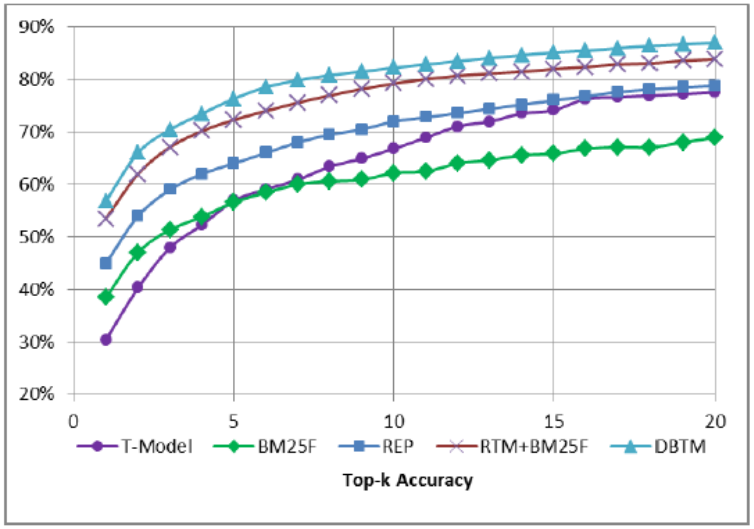
Approach:

- Each bug report modeled by LDA (topics proportion, etc)
  - Duplicate bug reports share same buggy topics
- Learn LDA parameters from training stage
- Report duplicate if high topic proportion similarity with group
  - Similarity using Jensen-Shannon divergence (method to estimate similarity of two distributions)
- IR scoring: using sum of tf-idf weights of all words in bug report

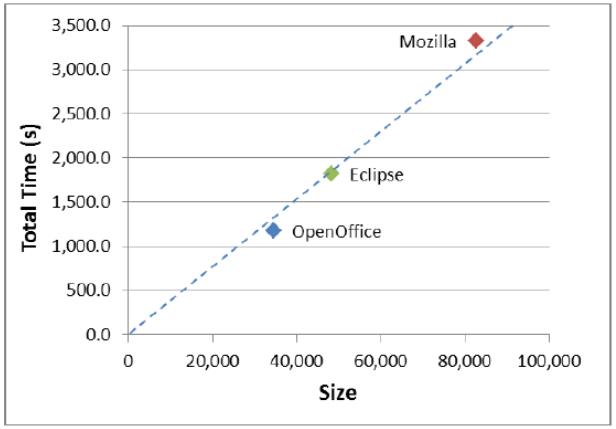
Duplicate bug report detection with a combination of information retrieval and topic modeling. – ASE 2012



Marked Correct if duplicate in top 10



Eclipse – comparison of approaches



Time complexity

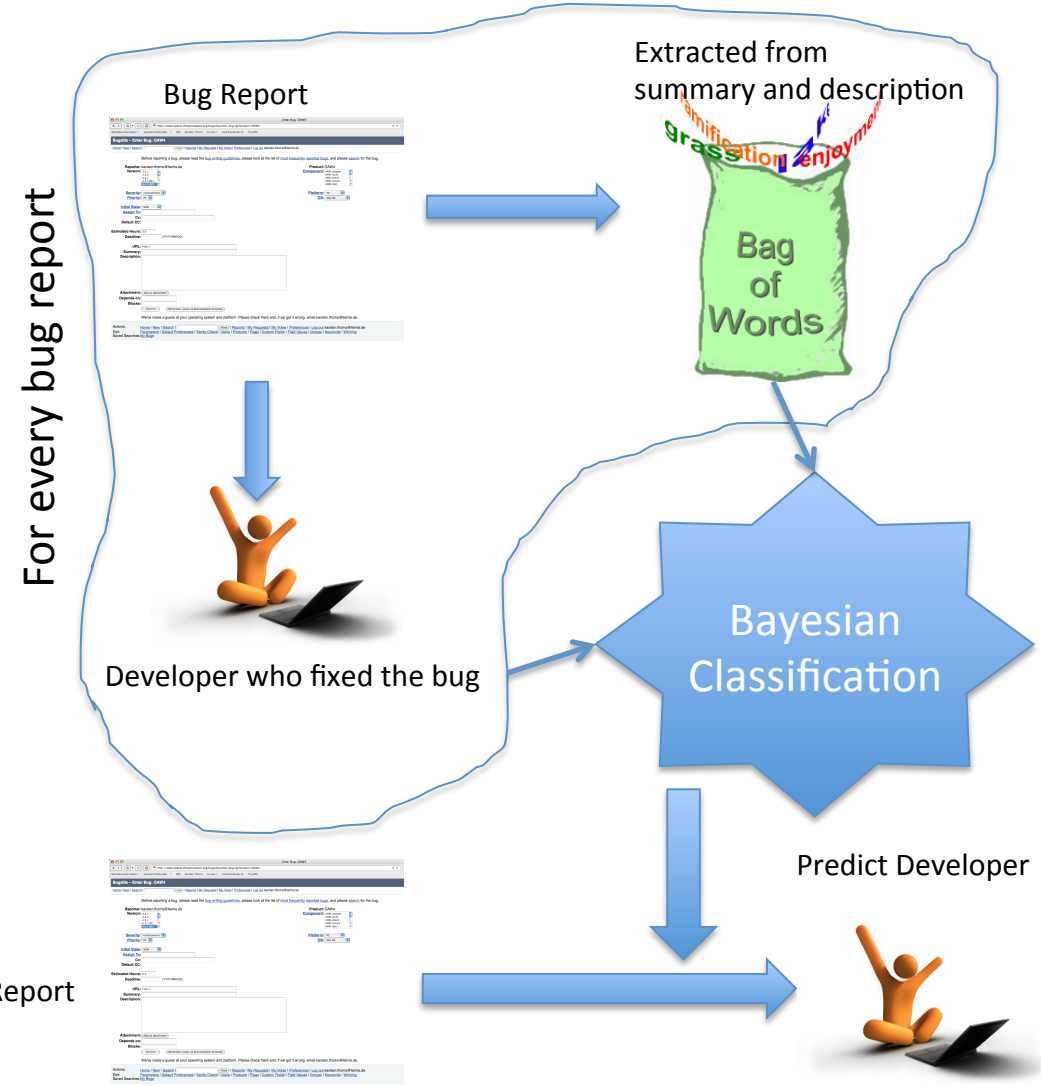
Bug Triaging

Automatic Bug Triage Using Text Categorization, Davor Cubranic and Gail C. Murphy ICSE 2004

Objective : Apply machine learning techniques to assist in bug triage by using text categorization to predict the developer that should work on the bug based on the bug's description.

Approach :

- Treat Bug Assignment Problem as 'text classification'.
- Use Supervised learning technique, by using past resolved bugs as the data set.
- Used Bayesian Learning Approach
  - Elegant, Adaptable to multi-class problem and performs well.



Automatic Bug Triage Using Text Categorization, Davor Cubranic and Gail C. Murphy ICSE 2004

Experiment

Preparing the Data Set : Selection of Eclipse Bug Reports (15859)

- Identifying the developer who fixed the bug
  - Assigned To Developer resolved it
  - Person who marked the bug as resolved (other than submitter)
  - Person who marked the bug as fixed
  - First responder to bug (when bug was not fixed – duplicate etc.)
  - No body responded to bugs (then those bugs were removed)
  - Not resolved then most recent assigned-to developer
- Final Set of reports – 15670 with 162 developers
- Words extracted from summary and description →tokenized → bag of words (no stemming)
- Randomly created training set and test set
- Multiple Runs and average reported

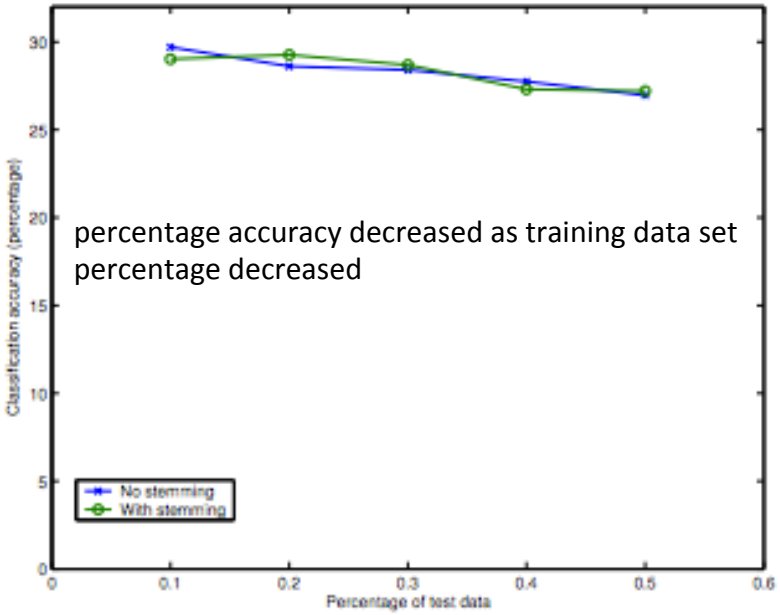
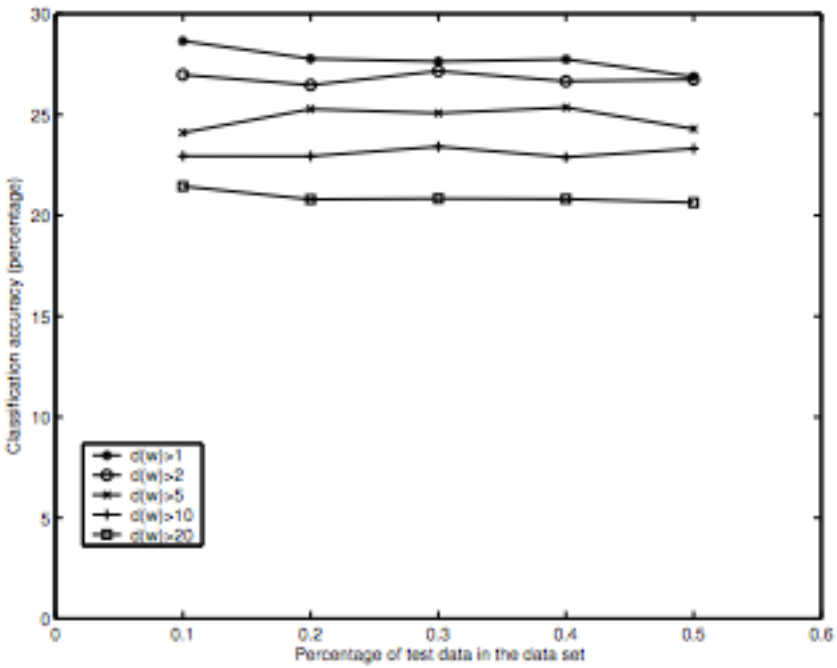


Figure 1. Classification accuracy without vocabulary truncation.

Automatic Bug Triage Using Text Categorization, Davor Cubranic and Gail C. Murphy ICSE 2004



Truncated vocabulary based on threshold of documents

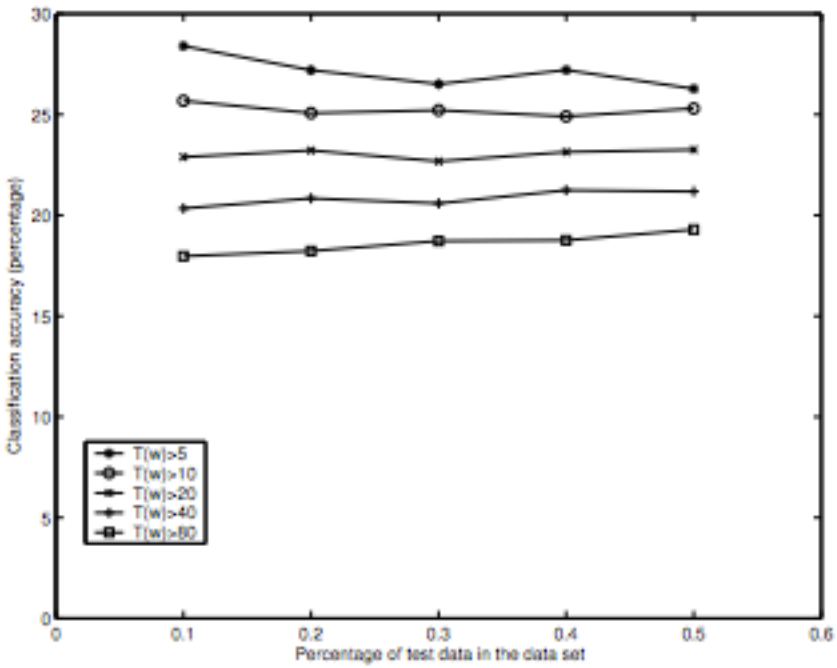


Figure 3. Classification accuracy when words occuring in the collection fewer than T times are removed from the vocabulary.

Conclusion:
Achieved 30% classification accuracy
Expected reduction in time/ effort for bug triager

http://www.cs.ubc.ca/labs/spl/papers/2004/seke04-bugzilla.pdf

Who should fix this bug? John Anvik, Lyndon Hiew and Gail C. Murphy ICSE 2006

Objective : Apply machine learning techniques to address bug triaging problem

Approach (when compared to previous paper):

- More thorough preparation of data
- Use of additional information beyond bug description
- Exploration of more algorithms
- Semi-automated : Triager has to choose the actual developer from the set recommended

Experiment

- Data Set : Eclipse and Firefox (bug reports from bugzilla). Training Set: 8655 – Eclipse and 9752 – Firefox. Test Set :170 – Eclipse and 22 – Firefox

Characterize bug reports → Assign a label to each bug report → Choose reports to train supervised algorithm

Normalized Feature Vector out of the words extracted from summary and description (using document length, intra and inter document frequency)

- If a report is resolved as FIXED, it was fixed by whoever submitted the last approved patch. (Firefox)
- If a report is resolved as FIXED, it was fixed by whoever marked the report as resolved. (Eclipse)
- If a report is resolved as DUPLICATE, it was resolved by whoever resolved the report of which this report is a duplicate. (Eclipse and Firefox)
- If a report is resolved as WORKSFORME, it was marked by the triager, and it is unknown which developer would have been assigned the report. The report is thus labeled as unclassifiable. (Firefox)

- 1% of Eclipse and 49% of Firefox removed
- Removed bugs if
  - developer not in project
  - developer fixed < 9 bugs reports

Who should fix this bug? John Anvik, Lyndon Hiew and Gail C. Murphy
ICSE 2006

Algorithm :
Support Vector Machines, Naïve Bayes and C4.5 Decision Tree

Table 3: The effect of different machine learning algorithms on recommender accuracy and recall.

Predictions	Naïve Bayes		SVM		C4.5	
	Firefox	Eclipse	Firefox	Eclipse	Firefox	Eclipse
1	59/2	54/6	64/2	58/7	64/2	40/5
2	59/2	49/11	52/3	52/13	41/3	34/9
3	59/2	44/15	57/6	47/16	42/5	31/12

$$Precision = \frac{\#\ of\ appropriate\ recommendations}{\#\ of\ recommendations\ made} \quad (1)$$

$$Recall = \frac{\#\ of\ appropriate\ recommendations}{\#\ of\ possibly\ relevant\ developers} \quad (2)$$

Table 4: Precision and recall for gcc.

Predictions	Precision/Recall
1	6/0.3
2	18/2
3	18/3

- One developer was very dominant
- Labeling strategy requires to be fine-tuned
- Developer based filtering reduced the number of developers to only 29

Extensions:
- Tried Unsupervised Algorithm (Expectation Maximization) – performed worst than Naïve Bayes
- Incremental Machine learning approach (Naïve Bayes) achieved only 28% accuracy

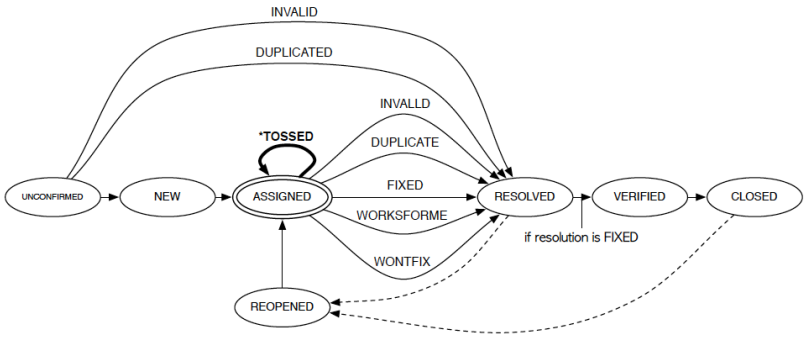
Source : http://www.cs.ubc.ca/labs/spl/projects/bugTriage/papers/icse2006.pdf

Improve Bug Triage with Bug Tossing Graph, Gaeul Jeong, Sunghun Kim, Thomas Zimmermann, FSE 2009

Objective : Apply graph model based on Markov chains to capture bug tossing history and use that to understand developer network and predict bug triaging

Approach :

- Bugs gets tossed between developers
- Create 'goal' oriented tossing graph



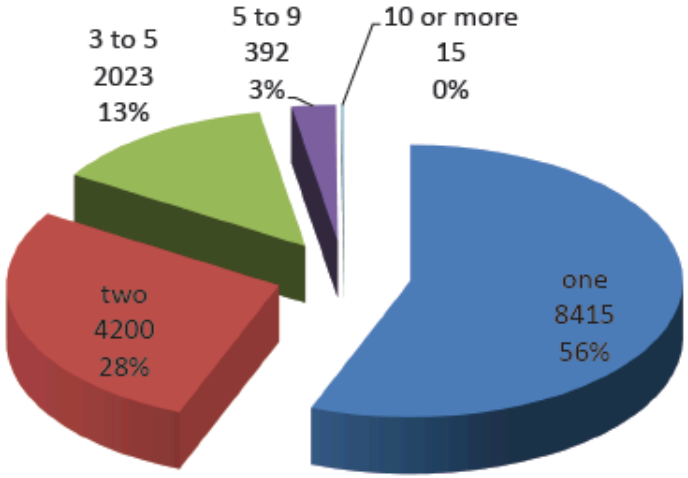


Figure 4: Eclipse bug report distributions based on the number of assigned developers. Only assigned and verified bugs are considered. About 56% of bugs are assigned to a single developer; 44% of bugs are assigned to more than one developer and have tossing events.

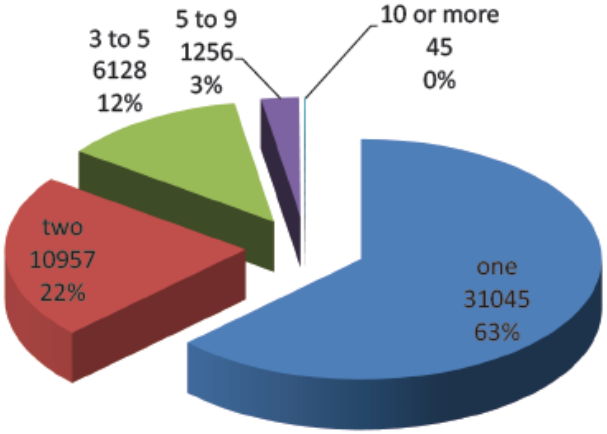


Figure 5: Mozilla bug report distributions based on the number of assigned developers. Only assigned and verified bugs are considered. About 37% of bugs have tossing events.

Improve Bug Triage with Bug Tossing Graph, Gaeul Jeong, Sunghun Kim, Thomas Zimmermann, FSE 2009

Table 2: Simple tossing paths.

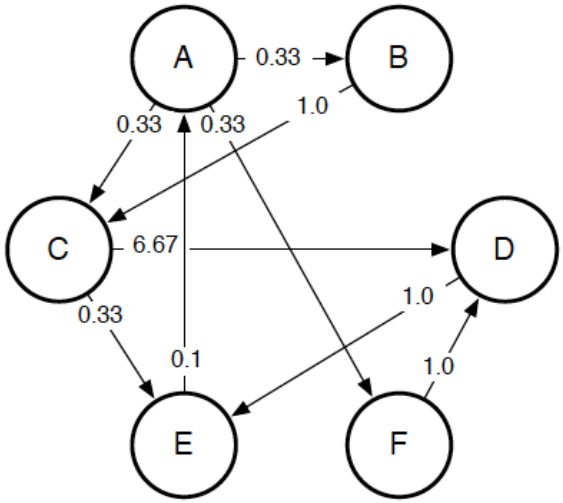
$A \rightarrow B \rightarrow C \rightarrow D$
$A \rightarrow C \rightarrow D \rightarrow E$
$C \rightarrow E \rightarrow A \rightarrow F \rightarrow D$

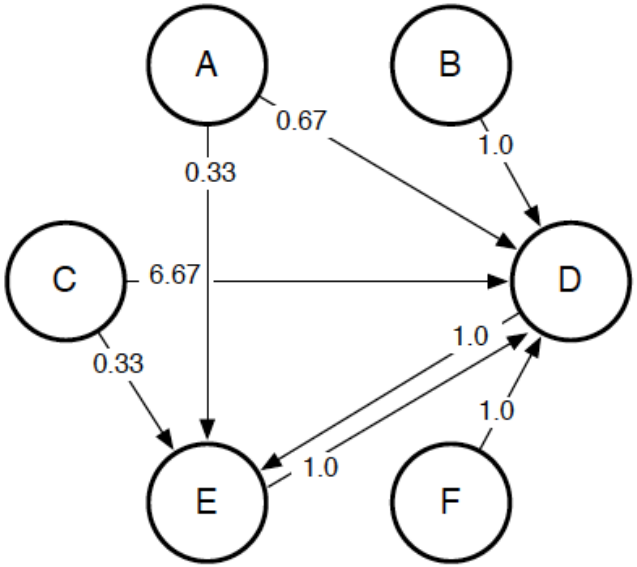
Table 3: Decomposed single steps from the sample tossing paths in Table 2 using two models. The numbers in parenthesis indicate the occurrences of each path.

actual paths	goal oriented paths
$A \rightarrow B$ (1), $B \rightarrow C$ (1), $C \rightarrow D$ (2), $A \rightarrow C$ (1), $D \rightarrow E$ (1), $C \rightarrow E$ (1), $E \rightarrow A$ (1), $A \rightarrow F$ (1), $F \rightarrow D$ (1)	$A \rightarrow D$ (2), $B \rightarrow D$ (1), $C \rightarrow D$ (2), $A \rightarrow E$ (1), $C \rightarrow E$ (1), $D \rightarrow E$ (1), $E \rightarrow D$ (1), $F \rightarrow D$ (1)

Calculate Probability of each Toss

C->D, C->D and C->E : hence C→D is 66% and C->E is 33%





Improve Bug Triage with Bug Tossing Graph, Gaeul Jeong, Sunghun Kim, Thomas Zimmermann, FSE 2009

Application of the Model for Understanding Developer Networks

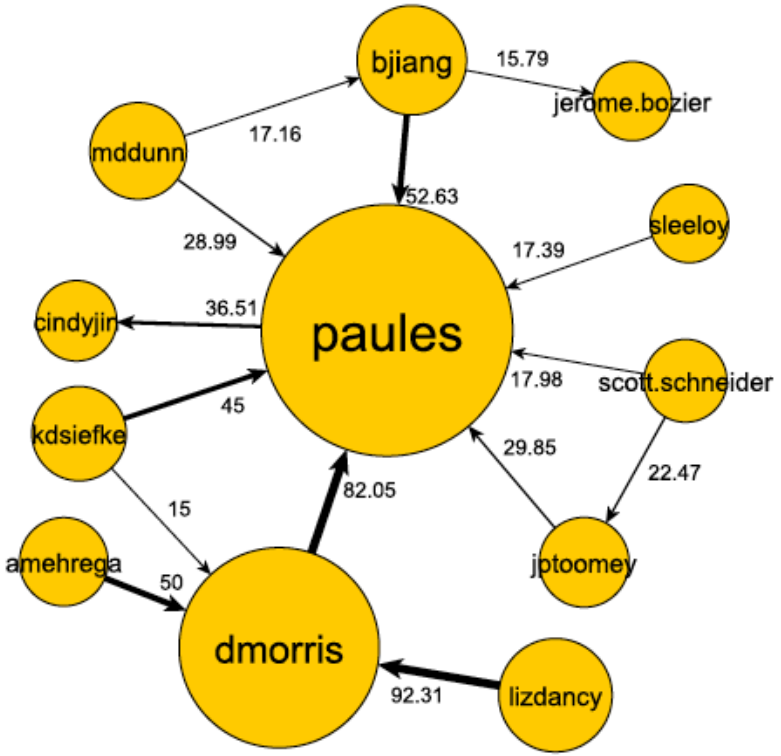


Figure 12: A partial tossing graph of Eclipse. Goal oriented with 25 minimum support and 15 transaction probability options are used. Nodes indicate developers and connecting lines (edges) represent tossing relationship. The numbers on edges show the tossing probability. The thick edges mean the corresponding edge has high tossing probability and the big nodes indicate they receive many bugs from others. For example, *paules* receives many bugs from other developers.

Application of the Model for Bug Triaging

Use goal oriented tossing graph and Weighted Breadth First Search to predict next developers

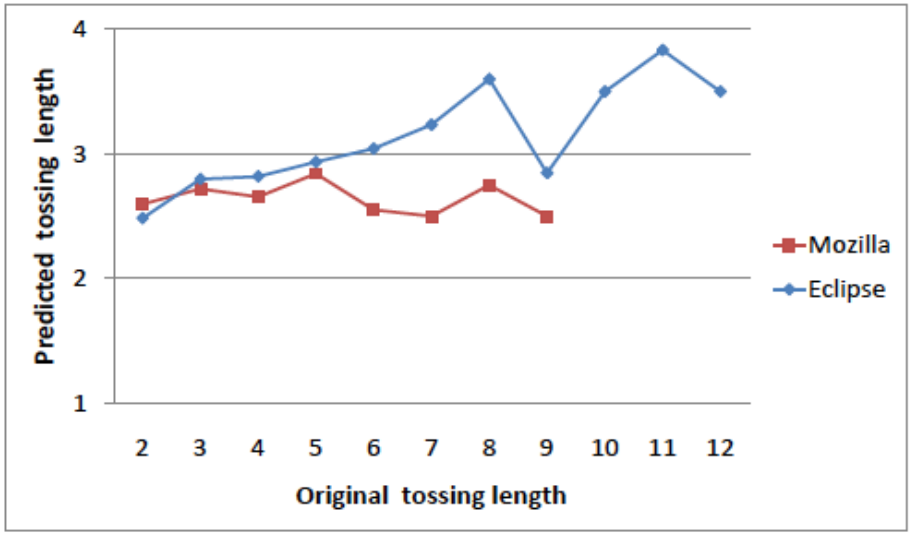


Figure 13: The reduced tossing length by predicting next proper developers using the tossing graph model.

Improve Bug Triage with Bug Tossing Graph, Gaeul Jeong, Sunghun Kim, Thomas Zimmermann, FSE 2009

Application of the Model for Bug Triaging

Combine goal oriented tossing graph with previous text based approaches

- If previous approaches suggest for a bug the set of developers as {robert.elves,mik.kersten,wuamy,susan_franklin,nitind}
- Goal Oriented Bug Tossing Graph suggest strong tossing relationship between
  - robert_elves →steffen.pingel and mik.kersten →relves
- Combined approach will predict the set as {robert.elves,steffen.pingel,mik.kersten,relves,wuamy}

Table 4: Bug assignment prediction accuracy in percentages using Naïve Bayes and Bayesian Network with/without tossing graph information. Since the accuracy of first 1 is the same with and without tossing graphs, it is omitted.

Program	ML algorithm	Selection	Accuracy (%)		Improvement
			ML only	ML + tossing graph	
Eclipse	Naïve Bayes	first 2	43.70	44.71	1.01
		first 3	49.87	53.15	3.27
		first 4	56.42	59.95	3.53
		first 5	60.71	63.48	2.77
	Bayesian Network	first 2	57.91	58.29	0.38
		first 3	66.71	68.47	1.76
		first 4	69.47	71.48	2.01
		first 5	75.88	77.14	1.26
Mozilla	Naïve Bayes	first 2	33.41	56.39	22.98
		first 3	45.39	63.82	18.43
		first 4	52.94	69.51	16.57
		first 5	59.35	72.92	13.58
	Bayesian Network	first 2	40.02	55.85	15.84
		first 3	50.25	63.05	12.81
		first 4	55.40	67.45	12.05
		first 5	59.53	70.82	11.29

Source : http://thomas-zimmermann.com/publications/files/jeong-esecfse-2009.pdf

Fuzzy Set and Cache Based Approach for Bug Triaging, Ahmed Tamrawi et. al FSE 2011

Objective : Use fuzzy set and cache based modeling of bug fixing expertise of developers to help in bug triaging

Approach (Bugzie):

1. Consider software system to have multiple technical aspects (technical terms)
2. For each technical term associated a fuzzy set of developers who can fix the bugs relevant to that corresponding aspect
3. Fixing correlation of developer towards a technical terms is represented by his/her membership score towards the corresponding fuzzy set
4. For a new bug report, Bugize combines the fuzzy set corresponding to its terms and ranks the developers based on their membership score to find the most capable fixer.

DEFINITION 1 (CAPABLE FIXER TOWARD A TERM). *For a specific technical term t, a fuzzy set C_t, with an associated membership function $\mu_t()$, represents the set of capable fixers toward t, i.e. the developers who have the bug-fixing expertise relevant to the technical aspect(s) described by t.*

DEFINITION 2 (MEMBERSHIP SCORE TOWARD A TERM). *The membership score $\mu_t(d)$ is calculated as the correlation between the set D_d of the bug reports that d has fixed, and the set D_t of the bug reports containing term t:*

$$\mu_t(d) = \frac{|D_d \cap D_t|}{|D_d \cup D_t|} = \frac{n_{d,t}}{n_t + n_d - n_{d,t}}$$

Fuzzy Set and Cache Based Approach for Bug Triaging, Ahmed Tamrawi et. al FSE 2011

Table 1: Statistics of All Collected Bug Report Data

Project	Time	Report	Fixer	Term
Firefox	04/07/98 - 10/28/10	188,139	3,014	177,028
Eclipse	10/10/01 - 10/28/10	177,637	2,144	193,862
Apache	05/10/02 - 01/01/11	43,162	1,695	110,231
NetBeans	01/01/08 - 11/01/10	23,522	380	42,797
FreeDesktop	01/09/03 - 12/05/10	17,084	374	61,773
Gcc	08/03/99 - 10/28/10	19,430	293	63,013
Jazz	06/01/05 - 06/01/08	34,228	156	39,771

Table 3: Term Selection for Eclipse's developers

Ed Merks	Darin Wright	Tod Creasey	James Moody
xsd	debug	marker	outgoing
ecore	breakpoint	progress	vcm
xsdschema	launch	decoration	itpvcm
genmodel	console	dialog	repository
emf	vm	workbench	history
xsdecorebuild	memory	background	ccv
xmlschema	jdi	font	team
eobject	suspend	view	cvs
xmlhandler	config	ui	merge
ecoreutil	thread	jface	conflict

Selection of fixer candidates

- Used locality principle, that people who have fixed recently are more likely to fix a new bug
- Bugzie choose top x% of developers sorted by their latest fixing time as Fixer candidates

Selection of Descriptive Terms

- For each developer (d) , Bugzie sorts the terms in the descending order based on correlation scores
- Selected top K terms in the list as significant terms for developer d.
- Such selection of terms across all developers is considered as selection for the system
- For any bug, which does not contain terms from set associated with the system, the bug is ignored by Bugziee

Training the model based on past resolved bug reports and testing the recommendation on the test set

Fuzzy Set and Cache Based Approach for Bug Triaging, Ahmed Tamrawi et. al
FSE 2011

Cache Strategy

- Caching support incremental updates
- Developer cache to store the list of candidate developers
- Terms cache to store the list of terms per developer
- Cache stored in descending values, and new entries are added and least values are removed

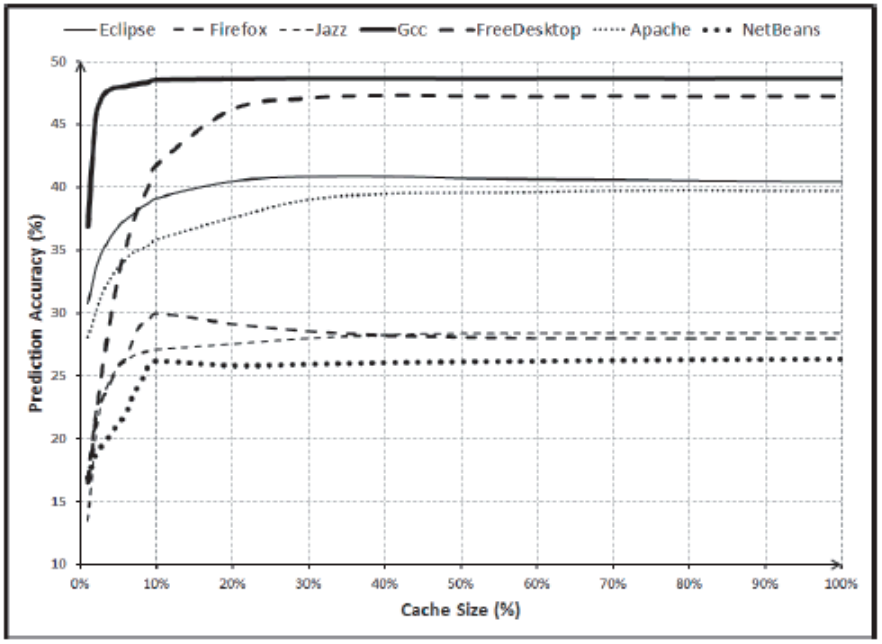


Figure 3: Top-1 Accuracy with Various Cache Sizes

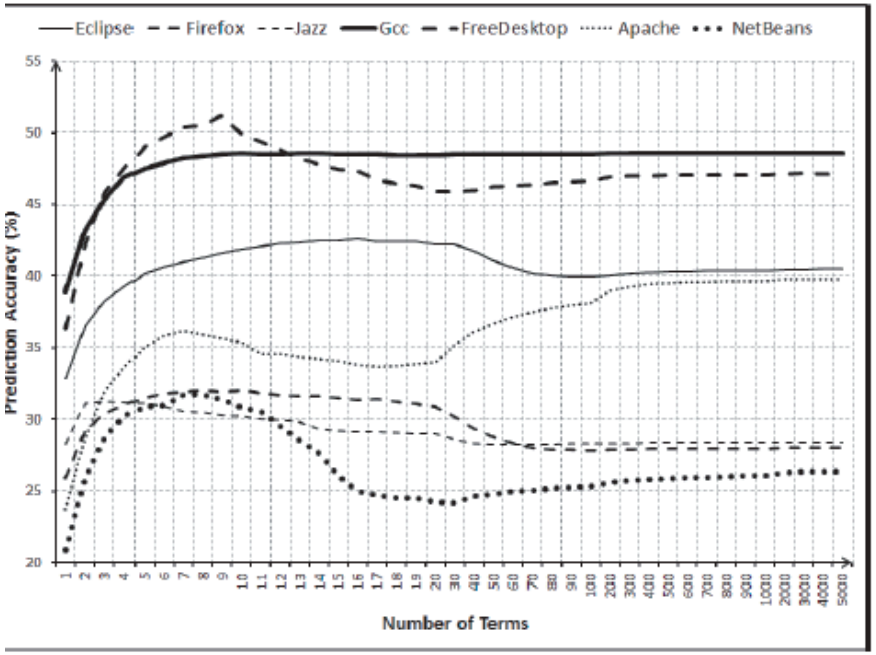


Figure 6: Top-1 Accuracy - Various Term Selection

How long will it take to fix this bug? Cathrin Weib et. al MSR 2007

Objective : Approach for automatically predicts the fixing effort, spent on fixing an issue

Approach:

- Use Nearest Neighbor Approach to query database of resolved issues for textual similarity reports.
- To identify similar issues, distance function is used
- Issue Title and Description are used to compute the similarity between two issue reports
- Lucene used as text similarity measuring engine
- To address low similarity score – threshold based Nearest Neighbor Approach has been used
  - at least a minimum similarity
  - at least a minimum set of issues

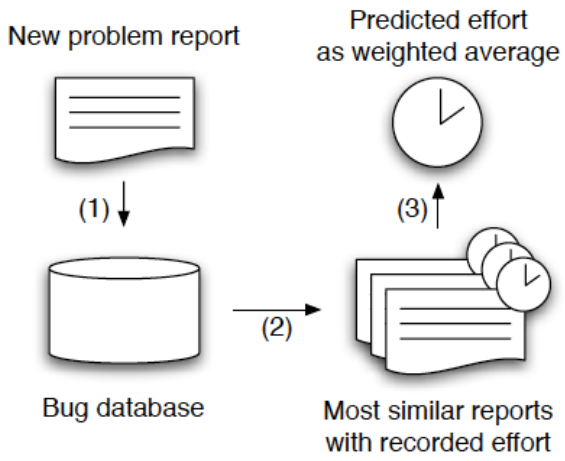


Figure 1. Predicting effort for an issue report

Experiment: JOBSS Bugs Reports used as subject

Average absolute residual.

$$r_i = |e_i - p_i| = |\text{Actual effort} - \text{Estimated effort}|$$

$$AAR = \frac{\sum_{i=1}^{n} r_i}{n}$$

Table 1. Prerequisites for issues.

	Count
Issues reported until 2006-05-05	11,185
Issues with	
– effort data (*timespent_sec* is available)	786
– valid effort data (*timespent_sec*≤*lifetime_sec*)	676
– *type* in ('Bug', 'Feature Request', 'Task', 'Sub-task')	666
– *status* in ('Closed, 'Resolved')	601
– *resolution* is 'Done'	575
– *priority* is not 'Trivial'	574
Issues indexable by Lucene	**567**

How long will it take to fix this bug? Cathrin Weib et. al MSR 2007

Percentage of predictions within ±x%.

$$Pred(x) = \frac{|\{i \mid r_i/e_i < x/100\}|}{n}$$

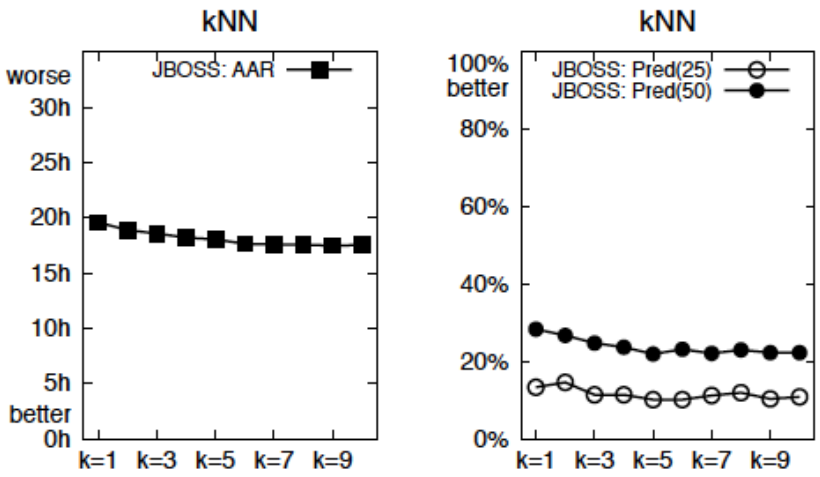


Figure 6. Accuracy values for kNN.

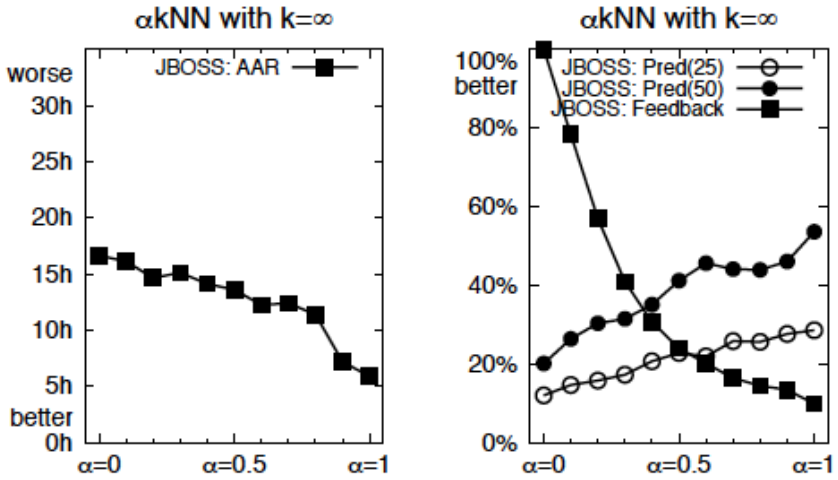


Figure 7. Accuracy for α-kNN with k=∞.

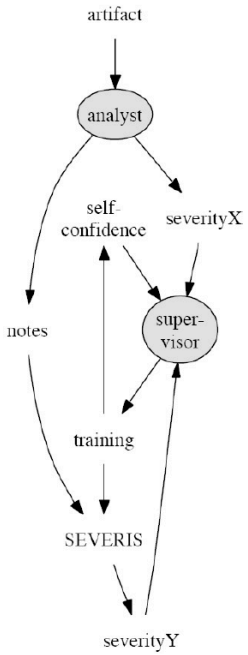
Results:

- No threshold, accuracy value is poor. Only 30% lie within a 50%
- With threshold – when similarity is 0, it is naïve approach (accuracy is low)
- Higher similarity, accuracy improves. However number of issues for which prediction is offered decreases (similarity is 0.9, predictions only for 13%)

Source : http://www.st.cs.uni-saarland.de/publications/files/weiss-msr-2007.pdf

Automated Severity Assessment of Software Defect Reports
ICSM 2008

Objective : Apply text mining and machine learning technique to assign severity to issue reports



Approach :

- Dimension reduction : Apply tokenization, stop word removal, stemming, tf*idf, Infogain over LSI – notes associated with bug report
- Rule Learner : Cohen's RIPPER rule learner.
  - Rules inferred from most informative tokens in reports with the severity level

Subject : NASA – PITS Ticket System

	Sev. 1	Sev. 2	Sev. 3	Sev. 4	Sev. 5
pitsA	0	311	356	208	26
pitsB	0	23	523	382	59
pitsC	0	0	132	180	7
pitsD	0	1	167	13	1
pitsE	0	24	517	243	41

severity	*precision*	*recall*	*f-measure*
1	0.91	0.59	0.71
2	0.20	0.72	0.31
3	0.83	0.89	0.86
4	0.08	1.00	0.14

How to fix this bug ?

What files should I investigate to fix this bug ?

Traditionally done using code-analysis.

DebugAdvisor: A Recommender System for Debugging. – FSE 2009

Objective: for a new bug report, find me past similar bug report, relevant people and/or related code files

Intuition:

- Bug fixed by People
- Fixing a bug you modify code files
- So once we find similar bugs, we can find relevant people and files transitively

- Contributions
  - A novel algorithm to find similar bugs
  - Use of factor graphs to transitively identify related people and files

DebugAdvisor: A Recommender System for Debugging. – FSE 2009

```
The customer experiences some deadlocks on a server. The problem is
random and may occur from several times a week to once a month. The
system looks hung because the global resource 'ObpInitKillMutant' is
help by a thread which tries to close a file for ever. So all the
processes having a thread waiting on 'ObpInitKillMutant' stop working
fine. Drivers such as TCP/IP continue to respond normally but it's
impossible to connect to any share.

Problem Impact:
The impact is high since the servers have to be rebooted when the
problem occurs. As no one can connect to the server anymore(net use),
the production is down. The problem was first escalated as a
severity A.

..
0: kd> !thread 82807020
ChildEBP RetAddr Args to Child
80210000 80c7a028 80c7a068 ntkrnlmp!KiSwapThread+0x1b1
80c7a074 00000000 00000000 ntkrnlmp!KeWaitForSingleObject+0x1b8
80c7a028 00000000 00000000 ntkrnlmp!IopAcquireFileObjectLock+0x58
82a6d7a0 80c7a028 00120089 ntkrnlmp!IopCloseFile+0x79
82a6d7a0 80c7a010 80f6da40 ntkrnlmp!ObpDecrementHandleCount+0x112
00000324 7ffdef01 00000000 ntkrnlmp!NtClose+0x170
00000324 7ffdef01 00000000 ntkrnlmp!KiSystemService+0xc9
00000324 80159796 000000c9 ntkrnlmp!ZwClose+0xb
000000c9 e185f648 00000000 ntkrnlmp!ObDestroyHandleProcedure+0xd
809e3008 801388e4 82a6d926 ntkrnlmp!ExDestroyHandleTable+0x48
00000001 82a6d7a0 7ffde000 ntkrnlmp!ObKillProcess+0x44
00000001 82a6d7a0 82a6d7f0 ntkrnlmp!PspExitProcess+0x54
00000000 f0941f04 0012fa70 ntkrnlmp!PspExitThread+0x447
ffffffff 00000000 00002a60 ntkrnlmp!NtTerminateProcess+0x13c
ffffffff 00000000 00002a60 ntkrnlmp!KiSystemService+0xc9
00000000 00000000 00000000 NTDLL!NtTerminateProcess+0xb
```

Figure 3: A bug description

```
FAILURE_BUCKET_ID:  0x8E_CLASSPNP!TransferPktComplete+1f5
SYMBOL_NAME:  CLASSPNP!TransferPktComplete+1f5
MODULE_NAME: CLASSPNP
IMAGE_NAME:  CLASSPNP.SYS
FAILURE_BUCKET_ID:  0x8E_CLASSPNP!TransferPktComplete+1f5
BUCKET_ID:  0x8E_CLASSPNP!TransferPktComplete+1f5
```

Bag of words based similarity

Sequence based similarity

Q = A,B,C

D1 = A,B

D2 = A,C

Q is closer to D1 than D2

Name-value pairs

Anatomy of a bug report

DebugAdvisor: A Recommender System for Debugging. – FSE 2009

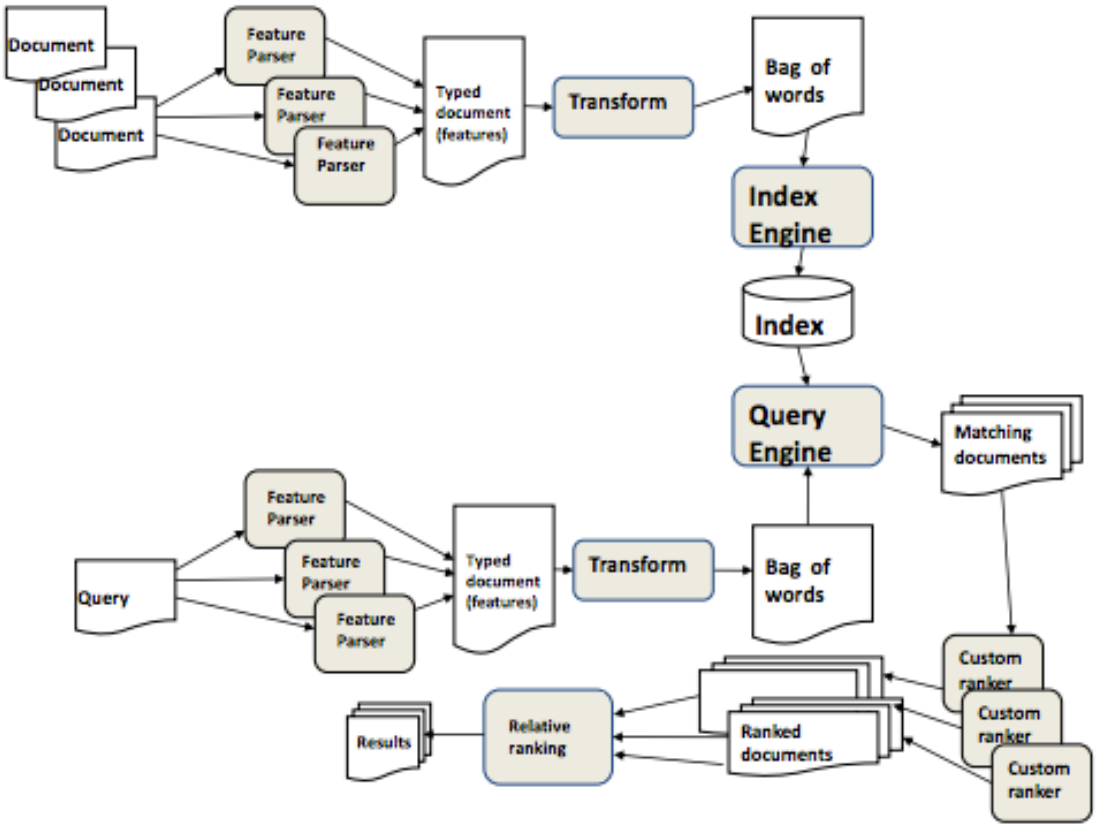
Consider each bug report as a typed document.

Same for query.

```
Type   BaseType   =     String
                   |   Int

Type   NamedDoc   =     Value(BaseType)
                   |   KeyValuePair(BaseType * Doc)

Type   Doc        =     Null
                   |   Base(BaseType)
                   |   Bag(NamedDoc set)
                   |   Ordered(NamedDoc list)

Type   D̂oc        =     Bag( Base set)
```

DebugAdvisor: A Recommender System for Debugging. – FSE 2009

Search for similarity based on each type separately and then combine results



DebugAdvisor: A Recommender System for Debugging. – FSE 2009

Experiment:

- Subject: Windows Servicibility Group
- Data source: Bug records, debug records – commands/ stack frames
- Measurement criteria: Precision/Recall & User Study

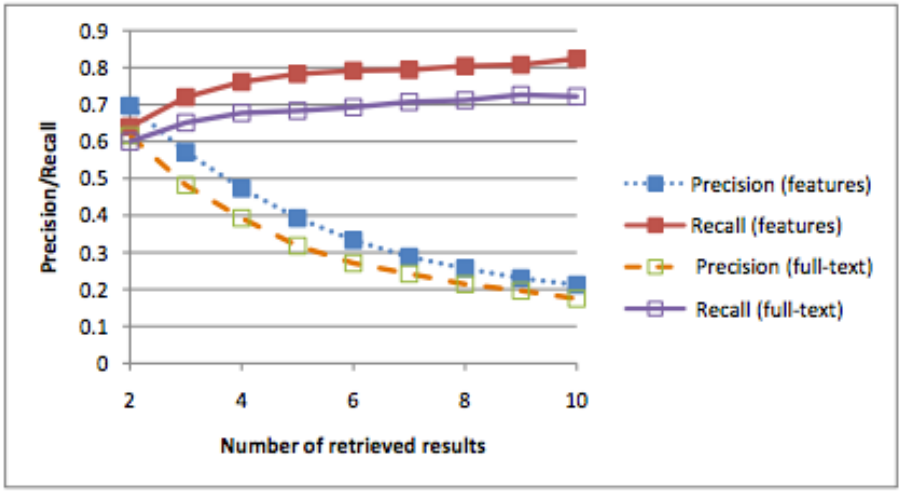
Findings:

- User study: 129 users, 628 queries, 208 responses out of which 78% said was useful

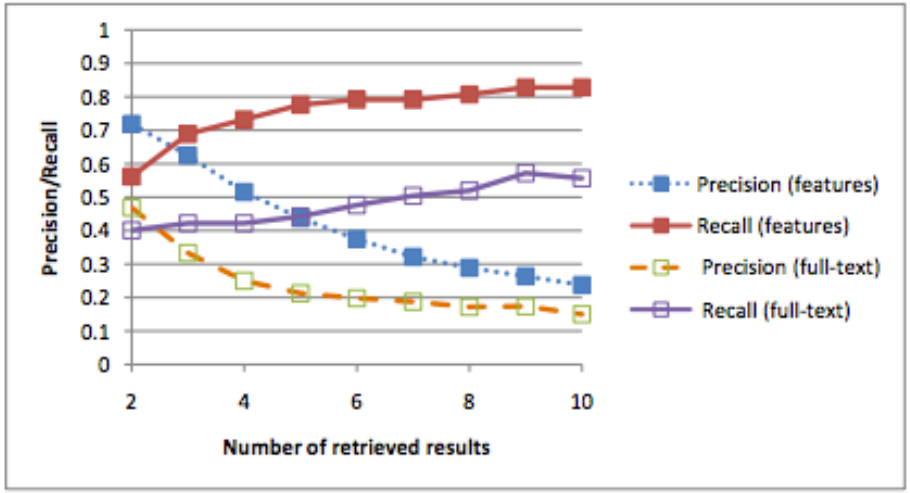
DebugAdvisor: A Recommender System for Debugging. – FSE 2009

Findings:

- For 50 queries, created truth set manually
- Compared precision and recall of proposed approach versus full-text
- Found to improve recall by 14%



With duplicate bug queries



Without duplicate bug queries (16)

Bug Report Summarization

Summarizing Software Artifacts: A Case Study of Bug Reports, Sarah Rastkar et. al ICSE 2010

Objective : Summarize bug reports using Supervised Learning Approaches

Approach :

- Manually generated summaries of bug reports, for the training set
- Evaluated multiple classifiers (logistic regression classifiers)
  - Trained on Email Thread (EC)
  - Trained on Email Threads and Meeting (EMC)
  - Trained on bug report corpus (BRC)
- Compared classifiers on metrics precision, pyramid precision and f-score
- Eclipse, Gnome, Mozilla and KDE.

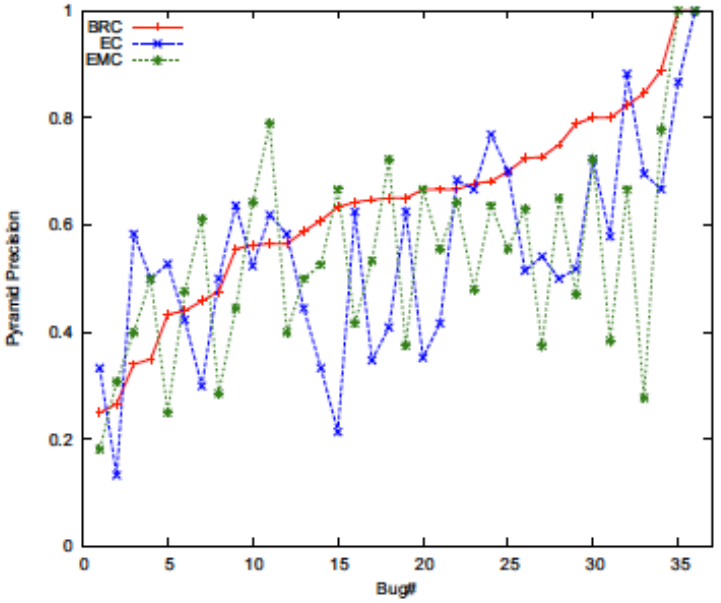


Figure 5: Pyramid precision for all classifiers.

Table 2: Evaluation measures.

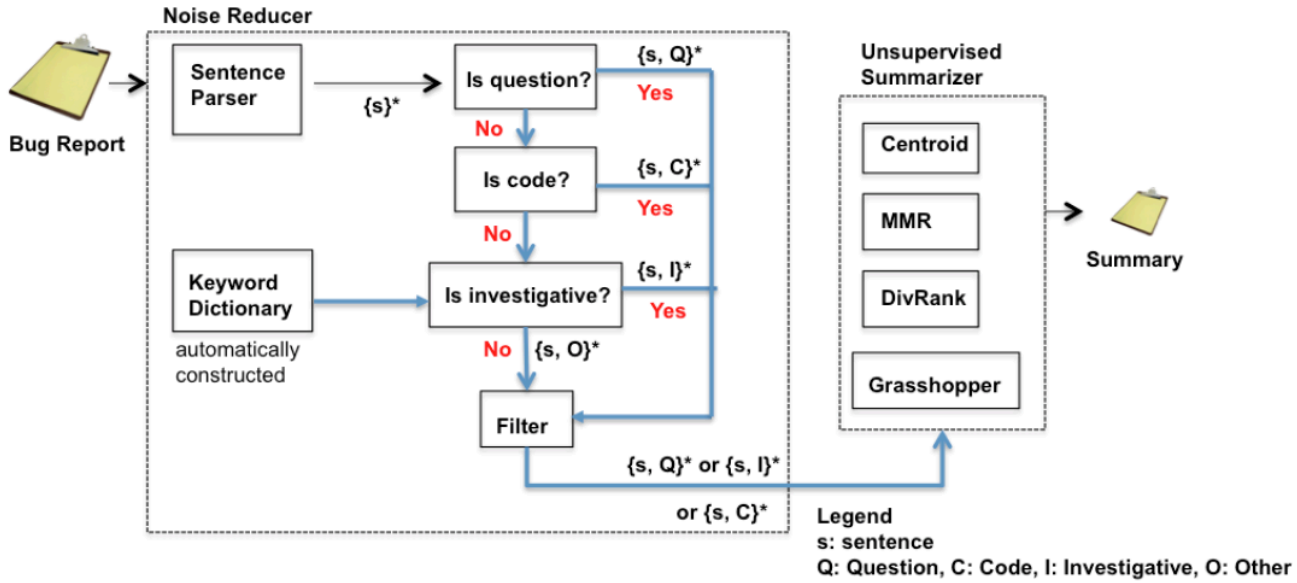
Classifier	Pyramid Precision	Precision	Recall	F-Score
BRC	.63	.57	.35	.4
EC	.54	.43	.3	.32
EMC	.53	.47	.23	.29

AUSUM : Approach for Unsupervised Bug Report Summarization, Senthil Mani et. al FSE 2012

Objective : Summarize bug reports using Un-Supervised Learning Approaches

Approach :

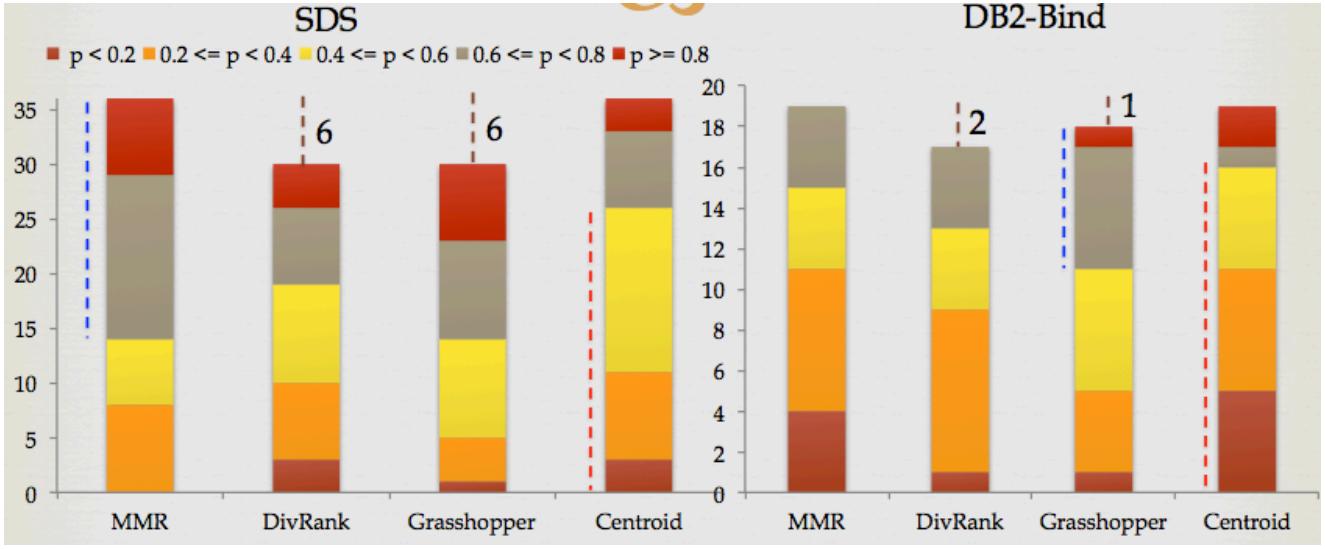
- Applied 4 unsupervised document summarization techniques (Centroid, MMR, Grasshopper, DivRank)
- How to select sentences?
  - Identify sentences as Question, Code, Investigative or useless
  - Pass the subset of these sentences to the summarizer
- Compared unsupervised with previous supervised techniques



AUSUM : Approach for Unsupervised Bug Report Summarization, Senthil Mani et. al FSE 2012

Experiment

Subjects	Number of Bugs	Average Comment Count	Average Comment Size	Total Sentences
SDS (Eclipse, Gnome, Mozilla and KDE)	36	6.5	9.5	2361
DB2-Bind (IBM)	19	21	16	6304



AUSUM : Approach for Unsupervised Bug Report Summarization, Senthil Mani et. al FSE 2012

Experiment : Comparing with Supervised Technqiues

Technique	Algorithm	Pyramid	Precision	Recall	F-score
Supervised	BRC (bug reports)	.63	.57	.35	.4
	EC (Email corpus)	.54	.43	.3	.32
	EMC (Email threads and meetings)	.53	.47	.23	.29
Unsupervised	Centroid	.52	.42	.43	.43
	MMR	.6	.47	.49	.48
	Grasshopper	.6	.49	.51	.5
	DivRank	.5	.45	.46	.46

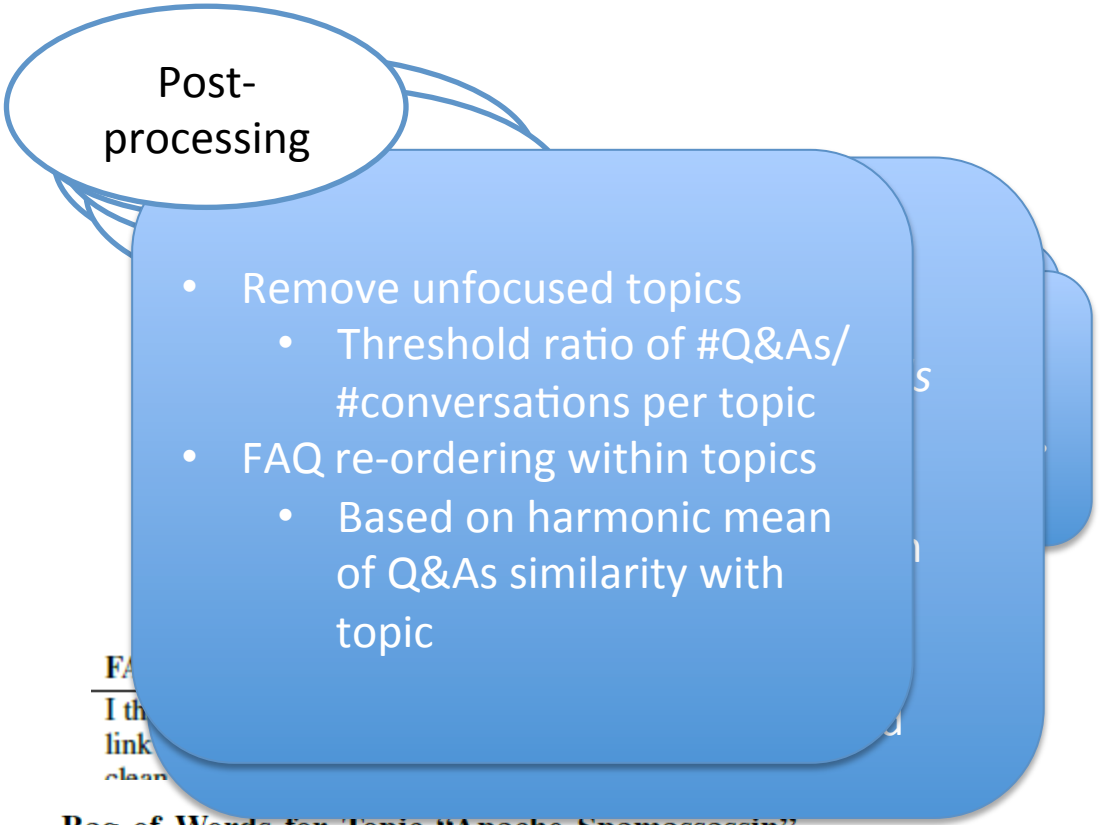
MINING SUPPORT FORUMS

Semi-automatically Extracting FAQs to Improve Accessibility of Software Development Knowledge *(Henss, Monperrus, Mezini ICSE 2012)*

Objective: Create FAQ candidates from mining mailing lists, Q&A forums, that can be validated/edited by humans

Approach Overview:

- Pre-processing
  - Noise removal
  - Look and feel
- Topic Mining
  - LDA for clusters of related conversations
- FAQ Assembly
  - Select "good" Q&As
  - Relevant topics selection & reordering



Semi-automatically Extracting FAQs to Improve Accessibility of Software Development Knowledge *(Henss, Monperrus, Mezini ICSE 2012)*

Qualitative Evaluation:

- Open source projects
- FAQs presented to top contributo
- 40 highest scoring Qs shown
- Answers presented
  - Accept answer
  - Modify answer
  - Discard the Question itself

Quantitative Evaluation:

- Merge 50 mailing lists and extract 50 topics
- Precision and recall for mined topics
- PP - preprocessing only
- FAQ – all steps

RECONSTRUCTING 50 MAILING-LISTS WITH LDA. THE FAQ ASSEMBLY TECHNIQUES IMPROVE THE PRECISION AND FILTER OUT UNFOCUSED TOPIC.

Topic Model (Main List)	Recall		Precision	
	PP	FAQ	PP	FAQ
Best topics				
#8 (NHibernate)	1.0	0.3743	0.9632	1,0
#46 (D-Bus)	0.8564	0.3479	0.8167	0.9408
#31 (Mutt)	0.86	0.32	0.8487	0.96
#44 (Hudson)	0.6767	0.29	0.7778	0.9305
#17 (Log4J)	0.8063	0.3099	0.4761	0.8627
Worst topics				
#20 (Firebug)	0.075	x	0.1546	x
#15 (Hudson)	0.03	x	0.0957	x
#49 (Mediawiki)	0.0183	x	0.1134	x
#36 (Firebug)	0.015	x	0.0769	x
#25 (Greasemonkey)	0.0611	x	0.14	x
Std Deviation	0.1712	0.0763	0.2817	0.1911
Average	0.6071	0.1829	**0.5711**	**0.7634**
Median	0.7717	0.1858	**0.6275**	**0.8261**

Mining whining in support forums with frictionary
(Andrew J Ko CHI 2012)

Objective: Aggregating and organizing problem topics from user requests in support forums

Approach Overview:

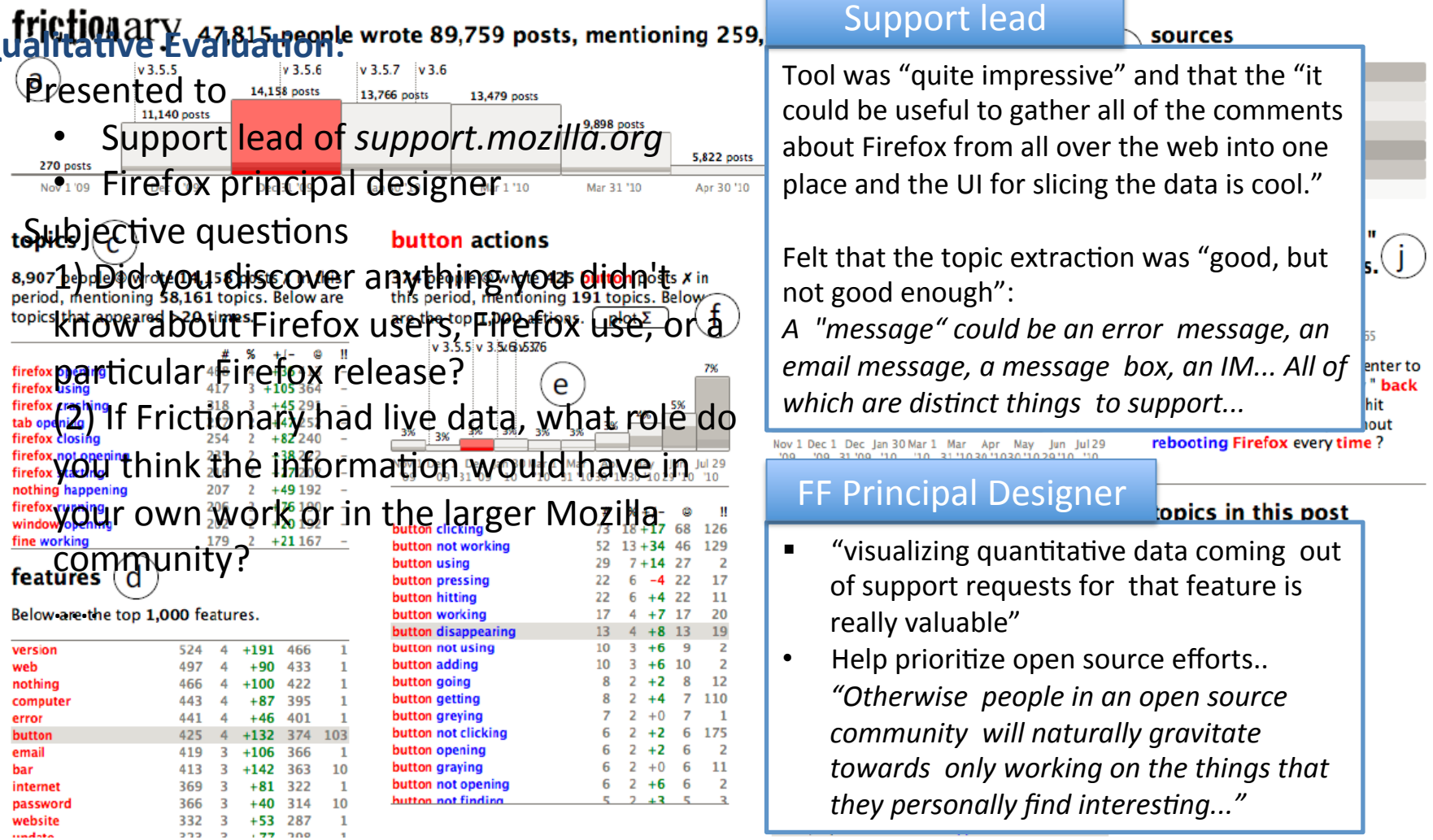
- NLP to Extract topics from Subject-Verb-Object pattern
  - Classify as problem/non-problem clause based on attributes
- Aggregate topics by synonyms (WordNet), spellings, hyphenation

kind	problem	example clauses	verb	animate	copular	past	desire	context	able	not
user inability	✓	*I cannot open windows*	✓	✓	×	×	×	×	✓	✓
user input	✓	*after I open a tab; if I don't open a tab; after I would open a tab*	✓	✓	×	×	-	✓	×	-
problematic behavior	✓	*firefox tabs will not close; firefox tabs should close*	✓	×	×	-	-	-	-	-
problematic state	✓	*Firefox is slow; tabs are stuck.*	✓	×	✓	-	-	-	-	-

Frictionary's clause classification function, via 8 attributes

Mining whining in support forums with frictionary
(Andrew J Ko CHI 2012)

Faceted Browsing -- Sort by frequency, prevalence, etc



Qualitative Evaluation:

- Presented to
  - Support lead of *support.mozilla.org*
  - Firefox principal designer
- Subjective questions
  1) Did you discover anything you didn't know about Firefox users, Firefox use, or a particular Firefox release?
  (2) If Frictionary had live data, what role do you think the information would have in your own work or in the larger Mozilla community?

Support lead

Tool was "quite impressive" and that the "it could be useful to gather all of the comments about Firefox from all over the web into one place and the UI for slicing the data is cool."

Felt that the topic extraction was "good, but not good enough":
A "message" could be an error message, an email message, a message box, an IM... All of which are distinct things to support...

FF Principal Designer

- "visualizing quantitative data coming out of support requests for that feature is really valuable"
- Help prioritize open source efforts..
  "Otherwise people in an open source community will naturally gravitate towards only working on the things that they personally find interesting..."

Thank you

Resources

- Introduction to Information Retrieval: http://nlp.stanford.edu/IR-book
- Introduction to Topic Models: http://www.cs.princeton.edu/~blei/kdd-tutorial.pdf