Introduction to Structured Document Clustering & ExtMiner

18.11.2008

TIES447 Data and Software Mining

Miika Nurminen (minurmin@jyu.fi)
University of Jyväskylä

Outline

- Text mining & Information retrieval basic concepts
- Structured document retrieval & clustering
- ExtMiner demonstration

Text Mining

- Dörre *et al:* ”Text mining applies data mining techniques, such as clustering, classification or association rule search to textual information.”
  - Natural language processing (NLP), information extraction (IE), and machine learning techniques applied to text collections are sometimes regarded as a form of text mining.
  - Text Mining: relatively new (late 90s) term, but in many cases older, well-established techniques are applied
- Typically high-dimensional (or irreducible to a simple vector representation) data
  - E.g. Document term as a dimension
- Text mining is highly dependent on document representation, or index
  - Background knowledge on information retrieval is required
- Data preprocessing (=feature selection) is essential (stopword filtering, term stemming etc)

Information Retrieval

- Tiedonhaku (Information Retrieval, IR) on tietojenkäsittelytieteen osa-alue, joka tutkii relevanttien dokumenttien hakua dokumenttikokoelmasta. Dokumenttien haun tarkoitus on tyydyttää käyttäjän tietotarpeet (tieto viittaa informaatioon, ei dataan – vrt. sql)
- Relevanssin käsitteen tarkka määrittely on hankalaa, koska se riippuu käyttäjän yksilöllisistä tietotarpeista ja tiedonhakujärjestelmän kyselykieli asettaa rajoituksia näiden tarpeiden muotoilulle.
- Tiedonhakujärjestelmän suorituskykyä arvioidaan tarkkuuden (precision) ja saannin (recall) avulla. Tarkkuus on palautettujen relevanttien dokumenttien osuus kaikista palautetuista, saanti on palautettujen relevanttien dokumenttien osuus kaikista relevanteista dokumenteista
  - esim. kuvassa harmaalla merkityt ovat palautettuja dokumentteja (B), R:llä merkityt relevantteja (A). tarkkuus = 5/8~62.5% ja saanti 5/6~83.3%).

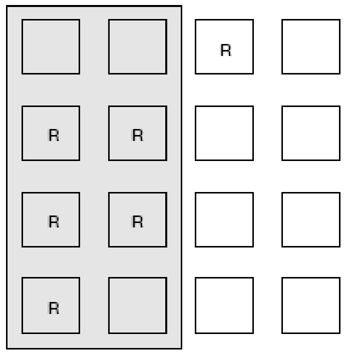
$$P = \frac{|A \cap B|}{|B|}$$

$$R = \frac{|A \cap B|}{|A|}$$

•Jos järjestelmällä on korkea tarkkuus, saanti on tyypillisesti matalampi ja päinvastoin.

•Esikäsittelytoimintoja

- •Document selection (e.g. web crawling)
- •Format conversions, data cleaning
- •Lexical analysis (separate terms from punctuation)
- •Stopwords removal (=toistuvia ja merkitykseltään vähäisiä sanoja)
- •Stemming (=perusmuotoon palautus)



Data

Clustering for Information Retrieval

- The Cluster hypothesis [van Rijsbergen]: *Closely associated documents tend to be relevant to the same requests. Document clustering should result in more effective, as well as more efficient, retrieval.*
  - Controversial, depends on the scale where document collection is perceived
  - Cluster model alone is not enough for IR – ranked lists are needed as well
- General approaches for clustering:
  - Pre-clustering (cluster-based searching)
    - Cluster the whole document collection in advance to general, thematic clusters
    - Divisive hierarchical clustering can be used – search results as clusters
    - Used in early IR systems to improve performance (compare the query only to limited set of documents or cluster prototypes) – search quality not improved.
  - Cluster-based browsing (see http://www.kartoo.com/)
    - User assesses the relevance of the clusters based on summarized descriptions (cluster keywords, prototypes etc) and navigates in the document collection using selection, clustering and query operations.
    - Scatter/Gather [Cutting *et al.*], ExtMiner
  - Search results clustering (see http://clusty.com/)
    - Useful with very large document collections (e.g. web search engines)
    - Can be used to combinine results from multiple search engines (retrieval fusion)
    - Usually better quality compared to pre-clustering

The baseline: Classical Vector model [Salton]

- THE Classical way to represent text documents.
- Documents and queries are represented as vectors. Each index term represents a dimension in document (or query) vector.
- Weighting can be used to emphasize relative importance of a term.
- tf*idf is a popular weighting scheme
  - tf (term frequency) is the relative count of term occurrences in a given document.
  - idf (inverted document frequency) is inverse proportional to number of documents where given term occurs
- Weighting assumption: documents are best described by terms that occur in a minority of documents in a collection (but multiple times documents where they occur in the first place)
- Documents and queries (i.e. ranking) or 2 documents (clustering) can be compared by calculating the dot product between vectors. This is also called cosine similarity.
- Term matrix is usually stored as *inverted file*: each term is associated with documents (and in some cases, positions) where the term occurs.

Classical IR implementation concepts

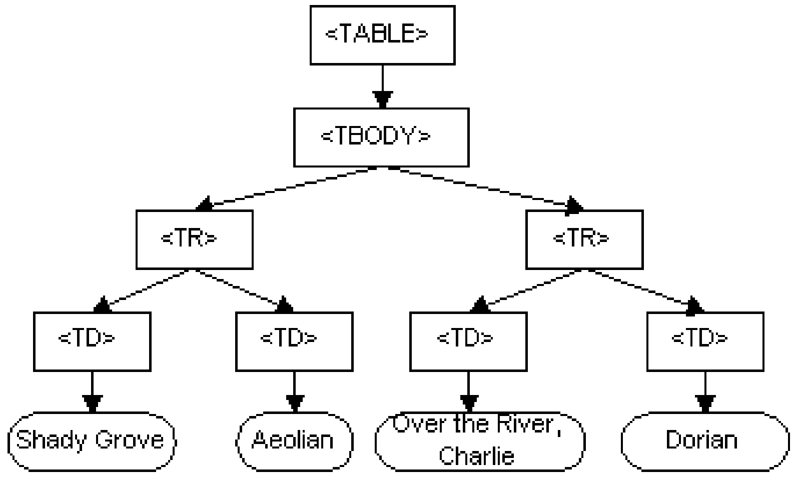
- Term/document matrix
  - Let T={1..n} term set and D={1..m} document set. The index can be represented with matrix $W_{n \times n}$
  - Columns as document vectors $d_j = (w_{1j}, w_{2j}, \ldots, w_{nj})^T \in \mathbb{R}^n,\ j \in \mathbf{D}$
  - Rows as term (weight) vectors $t_i = (w_{i1}, w_{i2}, \ldots, w_{im}) \in \mathbb{R}^m,\ i \in \mathbf{T}$
  - Matrix element w_{ij} denotes the weight (0..1) of term *i* in document *j*.
  - Queries are represented as weighted (document) vectors as well.
- Alternative weighting schemes:
  - Boolean model: only 0 (no term in doc) and 1 (term in doc) are used
  - One possible definition for tf*idf: $w_{ij} = tf_{ij} idf_i$, where

$$tf_{ij} = \frac{n(i,j)}{\max_{t \in \mathbf{T}}\{n(t,j)\}} \qquad idf_i = log\frac{|\mathbf{D}|}{|\mathbf{D}_i|}, \mathbf{D}_i = \{j \in \mathbf{D} | n(i,j) > 0\}$$

  - n(t,j) denotes the count of term t in document *j*, $|\mathbf{D}_i|$ the number of documents containing term *i*.
- Document vectors can be compared using cosine similarity
  - Jaccard, Dice, euclidean distance etc could also be used

$$sim_{cos}(d_j, q) = \frac{(d_j|q)}{\|d_j\|\|q\|} = \frac{\sum_{i=1}^n w_{ij} q_i}{\sqrt{(\sum_{i=1}^n w_{ij}^2)(\sum_{i=1}^n q_i^2)}}$$

Structured documents

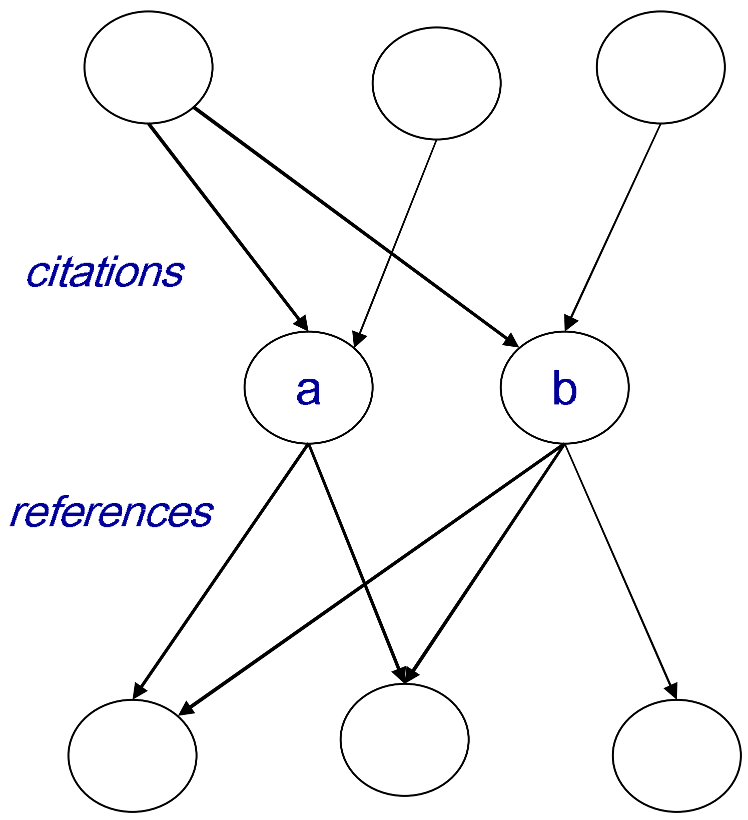


(Picture from http://www.w3.org/TR/DOM-Level-2-Core/)

- Unfortunately, classical vector model is not sufficient for structured documents (HTML, XML, LaTeX, process models, source code, software specifications & documents…)
- In addition to text **content**, documents can have **structure**, **hyperlinks** and (internal or external) **metadata**. Furthermore, text content is dispersed across document structures.
- All these aspects together should be taken to account for effective retrieval – ideally both in indexing and ranking.
- Another problem is document granularity: some documents are logically composed of multiple documents (e.g. chapters or sections in separate file). But a single document can have multiple themes as well (e.g. a single web page containing multiple small news articles - http://slashdot.org/).

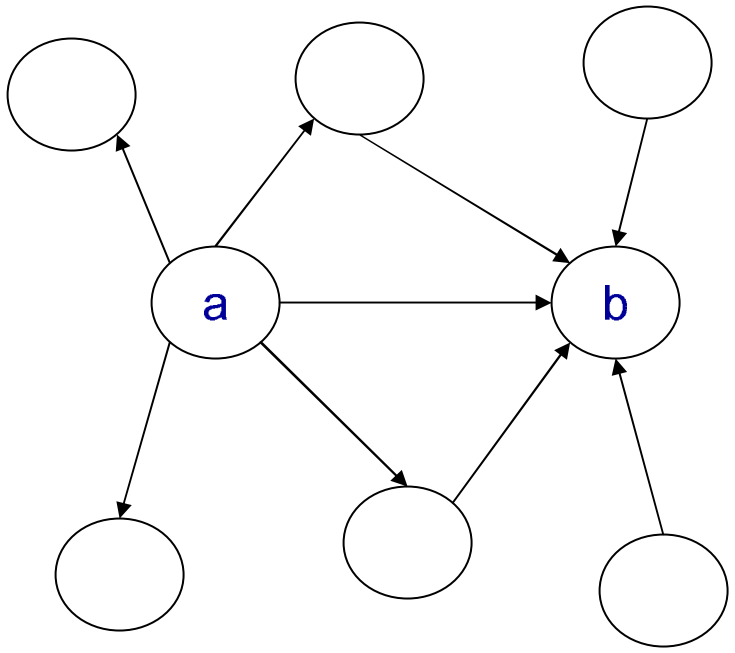
Link-based similarity

- Co-citation and bibiliographic coupling have been used widely for analyzing scientific articles in bibliometrics for decades.
- In the picture, co-citation between a,b is 1/3 (one document that cites both a and b), coupling 2/3 (a and b cite together 2 documents).
- For web documents, co-citation is often preferred over coupling because it evolves as new documents are updated.
- Could (and should) be combined to a new metric (Amsler)



Link-based ranking

- Essential algorithms: Pagerank [Brin & Page] & HITS [Kleinberg]
- PageRank resembles the ranking of scientific articles: the more high-ranking pages link to a page, the higher it scores.
  - Used to be key ranking component in Google search engine, but after the increased popularity of naturally highly connected web pages like blogs and wikis its importance seems to have been diminished somewhat
- HITS extracts Hub and authority pages from the link graph. Authority pages are approximately same as high-ranking PageRang pages.
- In the picture, a is a hub, b is an authority.
- In addition to link structure, anchor text can be indexed to describe the linked page content (anchor window)
  - Google bombs



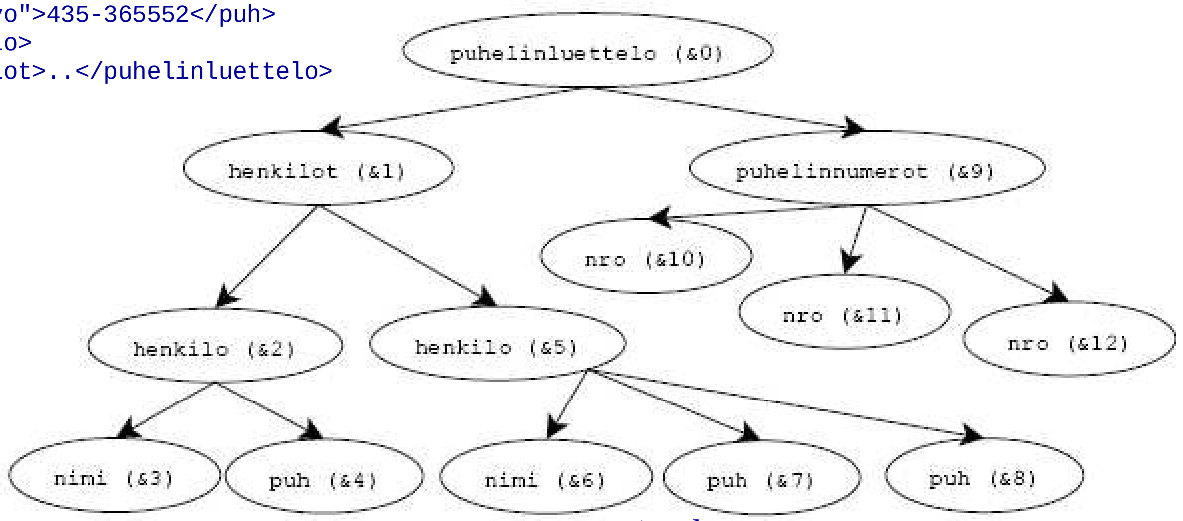
Rakenteinen indeksointi ja haku

- Rakenteisten dokumenttien indeksointitekniikat voidaan jakaa tekstipohjaisiin (ei ota huomioon rakennetta), puolirakenteisiin (ei ota huomioon skeemaa) tai rakenteisiin (ottaa huomioon skeeman, käytetään esim. tallennettaessa XML-dokumentteja relaatiotietokantaan)
- Käsitteellisellä tasolla XML-tietokanta koostuu puu/verkkorakenteessa olevista numeroiduista solmuista (vrt. DOM-puu)
- Rakenteiset indeksit voidaan jakaa [Weigel]
  - Perusindekseihin (yleistyksiä tekstidokumenttien käänteisindekseille, esim. sanalista, elementtilista),
  - polkuindekseihin (käyttävät dokumentin polkuja haun perusyksikköinä solmujen sijaan)
  - Puuindekseihin (dokumentin rakenteinen lyhennelmä, esim. DataGuide).
- Rakenteisessa haussa käsitellään esim. XPath/XQuery-muodossa annettava kysely, kootaan eri indekseistä mahdolliset hakutulokset ja yhdistetään ne (*content/structure join*)

```
<puhelinluettelo><henkilot>
<henkilo id="h1">
<nimi>A. A.</nimi>
<puh type="tyo">435-
345345</puh>
</henkilo>
<henkilo id="h2">
<nimi>N. N.</nimi>
<puh type="koti">11-
343255</puh><puh
type="tyo">435-365552</puh>
</henkilo>
</henkilot>..</puhelinluettelo>
```

Rakenteinen haku - esimerkki

Esim. Hakulause: /puhelinluettelo/henkilot/henkilo/[nimi='A. A.']



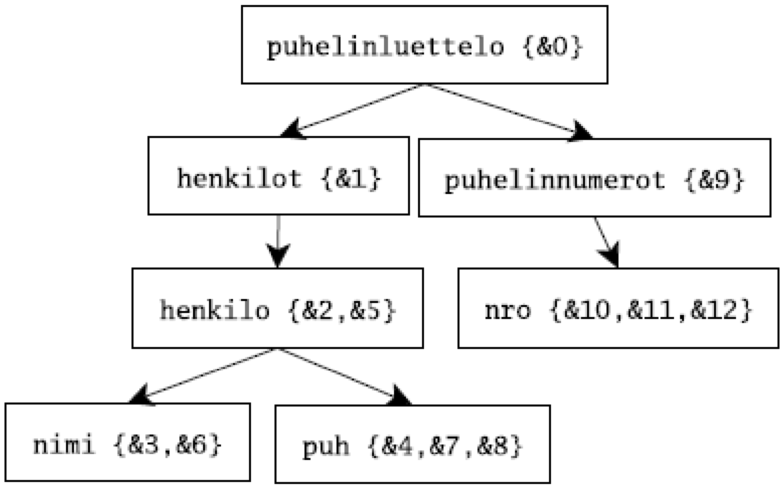
XML-tietokanta

Nimi	*Solmujoukko*
A.A	{3}
N.N	{6}
435-345345	{10}
11-343255	{11}
435-365552	{12}

Sanalista

Nimi	*Solmujoukko*
puhelinluettelo	{0}
henkilot	{1}
puhelinnumerot	{9}
henkilo	{2, 5}
nro	{10, 11, 12}
nimi	{3, 6}
puh	{4, 7, 8}

Elementtilista



DataGuide-indeksi

Nimi	*Polkujoukko*
A.A	$\{puhelinluettelo/henkilot/henkilo[0]/nimi\}$
N.N	$\{puhelinluettelo/henkilot/henkilo[1]/nimi\}$
435-345345	$\{puhelinluettelo/puhelinnumerot/nro[0]\}$
11-343255	$\{puhelinluettelo/puhelinnumerot/nro[1]\}$
435-365552	$\{puhelinluettelo/puhelinnumerot/nro[2]\}$

Käänteispolkulista

Structured indexing and ranking - summary

- Interpreting structured document can be interpreted as hypertext document collection: recursive indexing for content (for this special case, traditional vector model can be used) [Frisse]
- Structured searches require complex tree- or graph-based index structures [Weigel]
- Similarity comparison requires graph- or tree matching => slow [Luk]
- In general, vector-based index structures cannot be used effectively with structured documents, however…

Putting aspects together: Extended Vector Model

- …for many practical search schemes it suffices that *selected structures* are extracted from document and indexed as vectors
- or even better: for *metric* clustering a similarity measure will suffice, so document representation is not an issue
  - But: because of typical O(n^2) complexity, performance may be a problem
- Extended Vector Model [Fox] provides simple and general-purpose mechanism for calculating similarity from multiple components as weighted linear sum. Each component represents a collection of document features that can be calculated separately using classical vector model or other similarity measure.
- For example, there could be a component for document content in selected elements, another for link structures and third for various metadata fields.

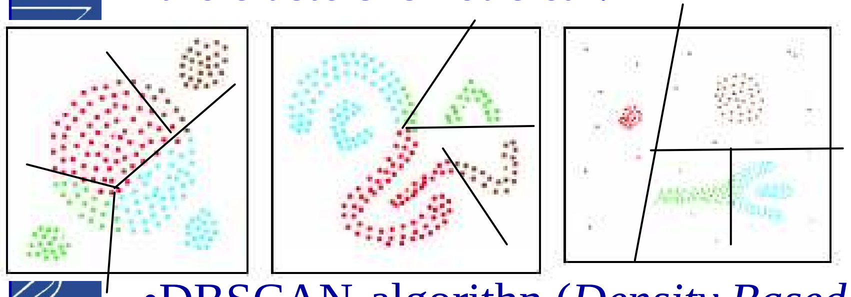
$$d_j = ((w_{1j}, w_{2j}, \ldots, w_{nj}), (c_{1j}, c_{2j}, \ldots, c_{mj}))$$

DBSCAN

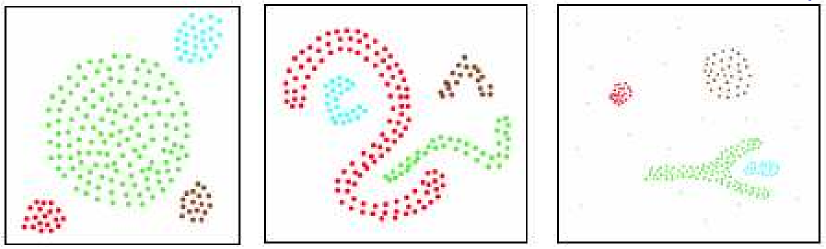
(Pictures from [Ester *et al.*])

•**Density**-based clustering method assume that the density of points inside clusters is larger compared to surrounding points.

•Density-based algorithms can find arbitrarily-shaped clusters and work well with outliers and noise with only similarity information. However, summarizing and interpretation of the clusters is not clear.

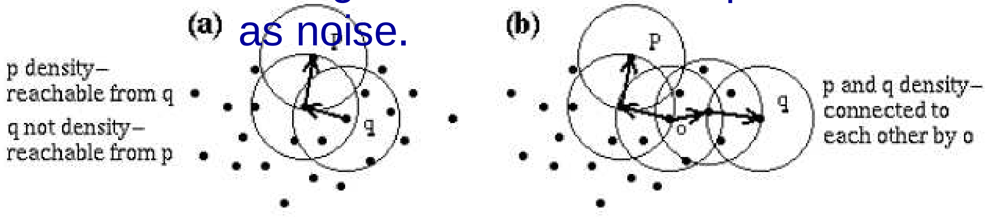


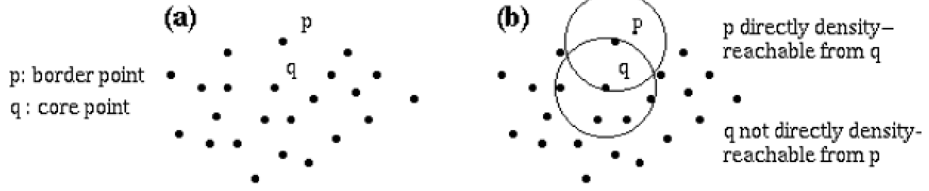
K-medoid vs DBSCAN



•DBSCAN-algorithn (*Density Based Spatial Clustering of Applications with Noise*) [Ester *et al.*] classifies the data objects to core points, border points, or noise.

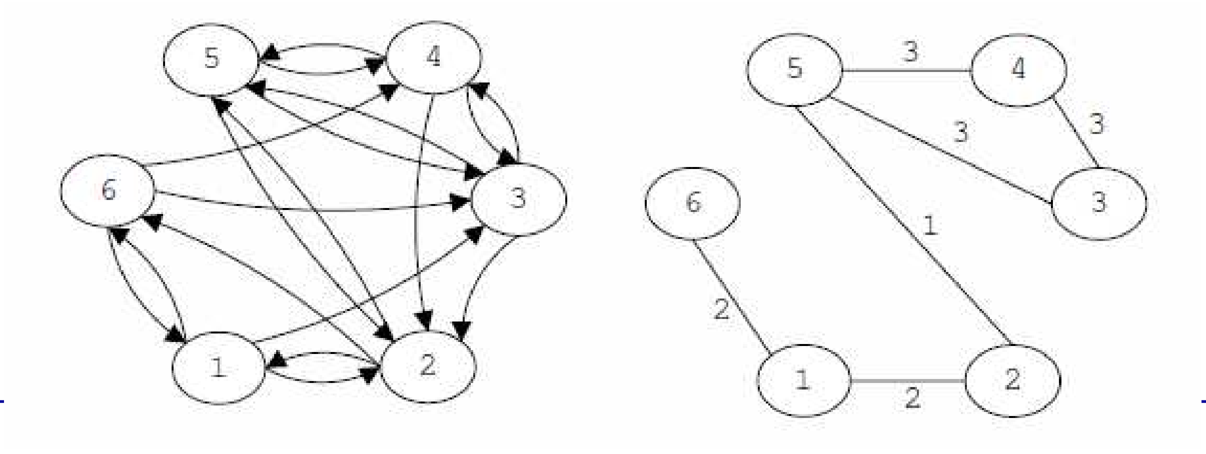
- •Clusters consist of core- and border points
- •For each point, a neighborhood in *eps*-radius is expanded iteratively, as long as at least *minpts* points exist in the neighborhood.
- •Core point always have at least *minpts* point in the neighborhood. Border points belong to a neighborhood of a core points but have less neighboring points. The rest is classified as noise.


(a)
p density-reachable from q
q not density-reachable from p
p
q
(b)
p
o
q
p and q density-connected to each other by o


(a)
p
q
p: border point
q : core point
(b)
p
q
p directly density-reachable from q
q not directly density-reachable from p

Shared Nearest Neighbor (SNN)

- A generic similarity metric for categorical data
- Useful if vector-based distances are not applicable
- Datajoukko hahmotetaan täydellisenä verkkona, jossa kaarien painot ovat alkioiden samanlaisuusarvot jonkin alustavan samanlaisuusmitan perusteella
  - Lasketaan suunnattu lähimpien naapurien graafi, jossa jokaisesta yksittäistä alkiota kuvaavasta solmusta on linkit n:ään lähimpään solmuun.
  - Muodostetaan suuntaamaton jaettujen lähimpien naapurien graafi siten, että jokaiselle alkioparille merkitään yhteisten naapurien määrä lähimpien naapurien graafista. SNN-graafiin otetaan mukaan vain pisteet, jotka ovat toistensa lähimpien naapurien joukossa.



Dokumenttiklusterin esittäminen [Modha]

Dokumentit esitetään laajennettuina vektoreina, joissa teksti- ja linkkikomponentit

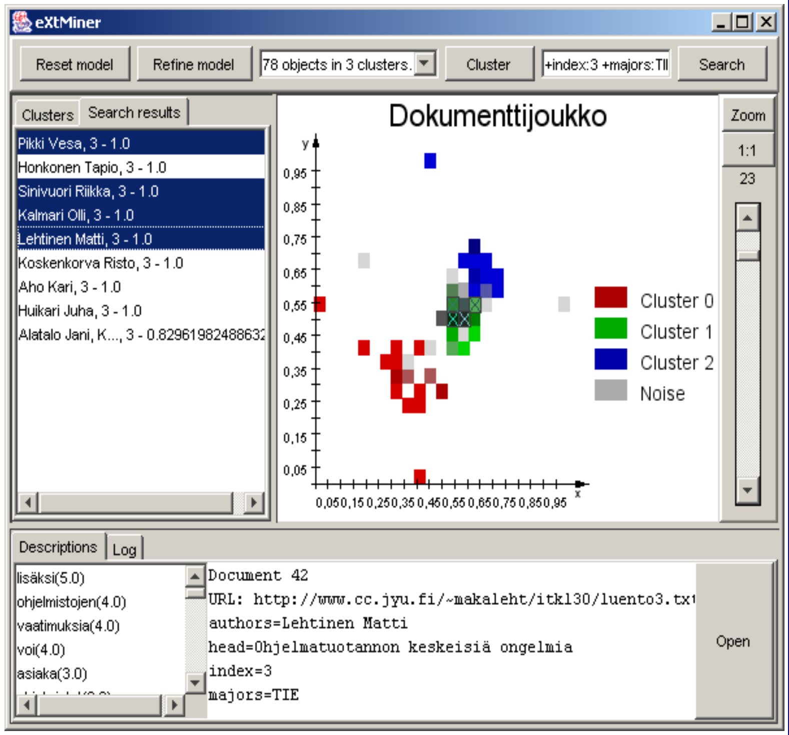
- *Lyhennelmä* on dokumentti, jolla on kaikkein tyypillisin sanavektori klustererin dokumenteista. Se vastaa tekstipohjaisten menetelmien prototyyppiä.
- *Läpimurto* (breakthrough) on dokumentti, jolla on kaikkein tyypillisin viittauslinkkien vektori klusterin dokumenteista. Viittaukset voivat tulla klusterin ulkopuolelta.
- *Yleiskatsaus* (review) on dokumentti, jolla on tyypillisin lähdelinkkien vektori. Lähdelinkit voivat mennä klusterin ulkopuolelle.
- *Avainsanat* ovat klusterin termikomponentin prototyyppivektorin voimakkaimmin painotetut termit. Ne kuvaavat klusterin tyypillisimpiä ja kuvaavimpia sanoja.
- *Viittaukset* ovat klusterin viittauskomponentin prototyyppivektorin voimakkaimmin painotetut linkit. Ne kuvaavat tyypillisimpiä dokumentteja, jotka viittaavat klusteriin.
- *Lähteet* ovat klusterin lähdekomponentin prototyyppivektorin voimakkaimmin painotetut linkit. Ne kuvaavat tyypillisimpiä dokumentteja, joihin klusterista viitataan.

Further generalization: Retrieval fusion

- Principle of combination [Fox *et al.*]: *Effective integration of more information slould lead to better information retrieval.*
- Fusion hypothesis [Zhang *et al.*]: *Different ranked lists usually have a high overlap of relevant documents and a low overlap of non-relevant documents.*
- Approaches to retrieval fusion [Yang]:
  - Datafuusio tarkoittaa dokumentin ja/tai kyselyn esittämistä useilla rinnakkaisilla tavoilla ja hakutuloksen muodostamista niiden yhdistelmänä. Haku kohdistuu vain yhteen dokumenttikokoelmaan.
  - Kokoelmafuusio on yhdistelmä erillisiä tai osittain samoja kokoelmia indeksoivien hakukoneiden tuloksista (metahaku). Datafuusiosta poiketen kokoelmafuusion painopisteenä on hajautettujen ja muodoltaan vaihtelevien tuloslistojen tehokas yhdistäminen ja uudelleenjärjestely, jonka ansiosta käyttäjä näkee yhtenäisen kokoelman
  - Paradigmafuusio on useiden eri hakuparadigmojen yhdistämistä. Teksti- ja linkkihaun sisältävät yhdistetyt hakutavat. Luokittelun ja klusteroinnin käyttö tiedonhaussa. Rakenteinen haku, semanttinen haku jne.

ExtMiner

- A platform and a proof-of-concept for combining
  - Different document features (eg. text, structure, links, metadata)
  - Ranking algorithms (eg. Cosine measure, PageRank)
  - Clustering algorithms (eg. DBSCAN, hierarchical clustering)
  - Visualization algorithms (eg. FastMap projection)
- Integrates many of the features previously implemented in separate systems
- Continuous search process based on ranked lists and cluster model



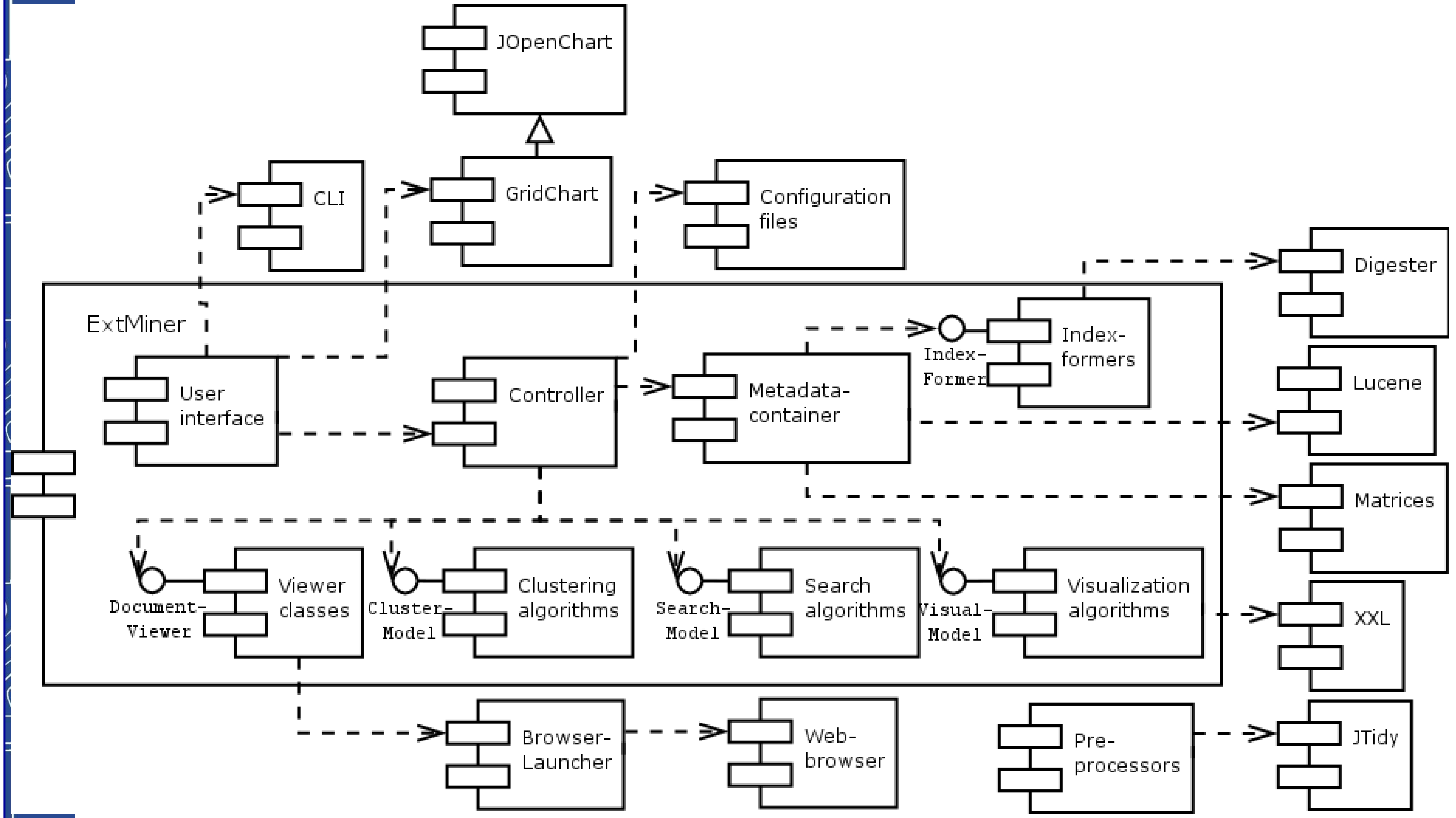
Pikki Vesa, 3 - 1.0
Honkonen Tapio, 3 - 1.0
Sinivuori Riikka, 3 - 1.0
Kalmari Olli, 3 - 1.0
Lehtinen Matti, 3 - 1.0
Koskenkorva Risto, 3 - 1.0
Aho Kari, 3 - 1.0
Huikari Juha, 3 - 1.0
Alatalo Jani, K..., 3 - 0.82961982488632

lisäksi(5.0)
ohjelmistojen(4.0)
vaatimuksia(4.0)
voi(4.0)
asiaka(3.0)

```
Document 42
URL: http://www.cc.jyu.fi/~makaleht/itkl30/luento3.txt
authors=Lehtinen Matti
head=Ohjelmatuotannon keskeisiä ongelmia
index=3
majors=TIE
```

http://extminer.sf.net/

ExtMiner architecture



ExtMiner architecture (decomposed)

•3 layers: UI, Application logic and Document index

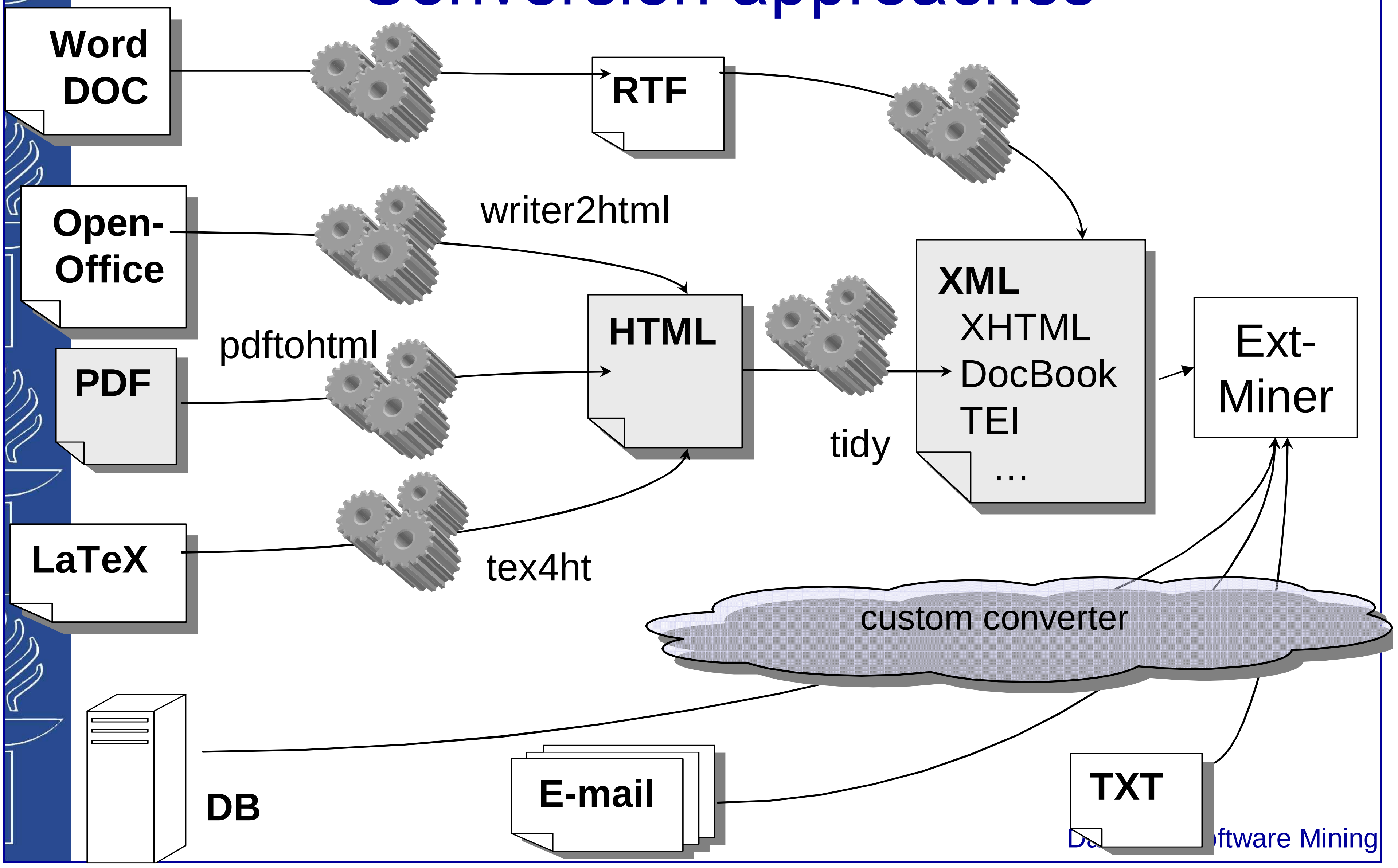
•Document index consists of similarity matrices and a field-based term/link index

•Application logic includes pluggable ranking, clustering and visualization algorithms and extensible mechanism for index creation from various document repositories

•UI provides customizable views for documents, ranked search result list and cluster model tree

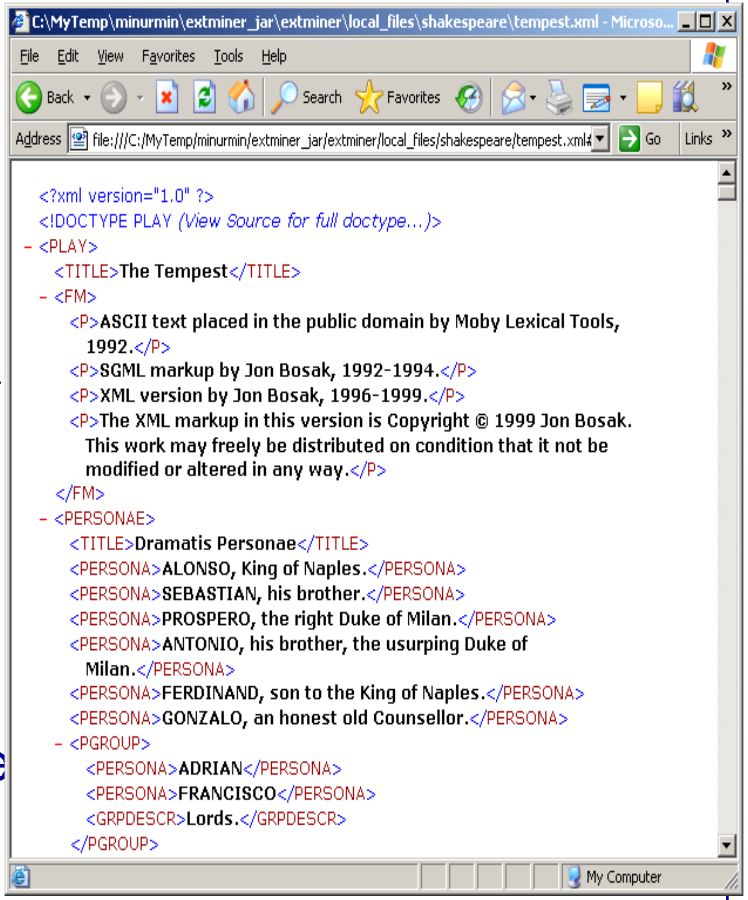
•Implemented with Java, published as open source. Third-party open source components (eg. Jakarta Lucene, JOpenChart) are utilized.

Conversion approaches



Indexing and configuration

- •Documents must be available in a local filesystem
- •Stemming,stopword removal and *tf*idf* weighting is performed by Lucene
- •Digester handles rule-based XML parsing
- •Documents are represented as field-based index (eg. tuples of vectors)
- •Fields can be index terms, links, headers or document type –specific external metadata or structural information encoded as vectors
- •Document-to-document similarities are precalculated for clustering
- •Different index formers and field definitions can be utilized, depending on document type and application domain


C:\MyTemp\minurmin\extminer_jar\extminer\local_files\shakespeare\tempest.xml - Microso...
File Edit View Favorites Tools Help
Back Search Favorites
Address file:///C:/MyTemp/minurmin/extminer_jar/extminer/local_files/shakespeare/tempest.xml
Go Links
My Computer

```
<?xml version="1.0" ?>
<!DOCTYPE PLAY (View Source for full doctype...)>
- <PLAY>
    <TITLE>The Tempest</TITLE>
  - <FM>
      <P>ASCII text placed in the public domain by Moby Lexical Tools,
        1992.</P>
      <P>SGML markup by Jon Bosak, 1992-1994.</P>
      <P>XML version by Jon Bosak, 1996-1999.</P>
      <P>The XML markup in this version is Copyright © 1999 Jon Bosak.
        This work may freely be distributed on condition that it not be
        modified or altered in any way.</P>
    </FM>
  - <PERSONAE>
      <TITLE>Dramatis Personae</TITLE>
      <PERSONA>ALONSO, King of Naples.</PERSONA>
      <PERSONA>SEBASTIAN, his brother.</PERSONA>
      <PERSONA>PROSPERO, the right Duke of Milan.</PERSONA>
      <PERSONA>ANTONIO, his brother, the usurping Duke of
        Milan.</PERSONA>
      <PERSONA>FERDINAND, son to the King of Naples.</PERSONA>
      <PERSONA>GONZALO, an honest old Counsellor.</PERSONA>
    - <PGROUP>
        <PERSONA>ADRIAN</PERSONA>
        <PERSONA>FRANCISCO</PERSONA>
        <GRPDESCR>Lords.</GRPDESCR>
      </PGROUP>
```

Searching and clustering

•Extended vector model is applied both in ranking and clustering similarity calculation

•Let d be a document and q a query, both represented as tuples of n vectors (fields). Relevance estimate R is calculated as

$$R(d,q) = \sum_{k=1}^{n} w_k sim_k(r_k(d), r_k(q))$$

•r denotes the restriction that extracts k-th vector from the tuple, sim is the similarity measure (such as boolean matching, cosine measure or co-citation), w denotes a field-specific weight supplied by the user (or matched evenly by default)

•Substitute q with another document and you have a document-to-document similarity measure for clustering

•Any metric clustering algorithm can be used, provided that the implemantation is available

User interface and visualization

- Iterative search and clustering process
  - Search and clustering can be performed iteratively and focused to an appropriate subset of the collection
- Interactive cluster model
  - The user can select documents from any of the views provided by the application: ranked list, cluster tree or visual projection. Cluster tree is interactive: a cluster can be marked as noise or subclusters of a single cluster can be merged (useful with hierarchical clustering)
- Simultaneous views for lists and clusters
  - Both views are needed since lists and clusters support different search objectives. Clusters are easy to understand and help to cope with ambiguous terms, although they do not improve search quality as such.
- Any MDS (multidimensional scaling) –style projection algorithm can be used for visualization (currently FastMap)
- Documents can be opened in web browser or custom viewer (eg. text editor, XML tree view)