MOVING

Deliverable 3.1: Technologies for MOVING data processing and visualisation v1.0

Till Blume, Falk Böschen, Lukas Galke, Ahmed Saleh, Ansgar Scherp, Matthias Schulte-Althoff/ZBW
Chrysa Collyda, Vasileios Mezaris, Alexandros Pournaras, Christos Tzelepis/CERTH
Peter Hasitschka, Vedran Sabol/KC
Aitor Apaolaza, Markel Vigo/UMAN
Tobias Backes, Peter Mutschke/GESIS
Thomas Gottron/Innovation Lab, SCHUFA Holding AG

31/03/2017

Work Package 3: Data processing and data visualisation technology

TraininG towards a society of data-saVvy inforMation prOfessionals to enable open leadership INnovation

Horizon 2020 - INSO-4-2015
Research and Innovation Programme
Grant Agreement Number 693092

Dissemination level	PU
Contractual date of delivery	31/03/2017
Actual date of delivery	31/03/2017
Deliverable number	3.1
Deliverable name	Technologies for MOVING data processing and visualisation v1.0
File	`MOVING_D3.1.tex`
Nature	Report
Status & version	Final v1.0
Number of pages	79
WP contributing to the deliverable	3
Task responsible	ZBW
Other contributors	CERTH, KC, UMAN, GESIS
Author(s)	Till Blume, Falk Böschen, Lukas Galke, Ahmed Saleh, Ansgar Scherp, Matthias Schulte-Althoff/ZBW Chrysa Collyda, Vasileios Mezaris, Alexandros Pournaras, Christos Tzelepis/CERTH Peter Hasitschka, Vedran Sabol/KC Aitor Apaolaza, Markel Vigo/UMAN Tobias Backes, Peter Mutschke/GESIS Thomas Gottron/Innovation Lab, SCHUFA Holding AG
Quality Assessors	Irina Bienia, Michael Wiese/EY
EC Project Officer	Hinano SPREAFICO
Keywords	Technologies, data acquisition, data processing, data visualisation, user data logging, common data model

Executive Summary

This deliverable "D3.1 Technologies for MOVING data processing and visualisation v1.0" provides an initial common data model as well as an initial set of data acquisition, data processing, user logging and data visualisation components. The common data model is able to represent full-texts, metadata, HTML content and video data (Section 2). Data acquisition and data processing is described in Section 3 and comprises seven different techniques. This includes concepts and prototypes for three different crawlers for webpages and social media content (Section 3.1), the use of word embeddings for improved information retrieval (Section 3.2), the text extraction from scholarly figures to increase the amount of textual content (Section 3.3), the metadata extraction from PDFs to make PDFs retrievable (Section 3.4), an adaptive index for Linked Open Data to harvest additional metadata (Section 3.5), video fragmentation, concept detection and video transcript analysis (Section 3.6) and the disambiguation of string representations of authors (Section 3.7). The logging of user interaction data as well as the dashboard to test hypotheses against the logging data is described in Section 4. A set of visualisations as well as a functional prototype is described in Section 5. Finally, we summarise the main contribution for each task highlighting the achievements of year 1 and give an outlook for year 2.

Table of contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Explanation
API	Application Programming Interface
AVS	Ad-hoc video search
BM25	Best Matching 25
BoW	Bag of Words
BRIEF	Binary Robust Independent Elementary Features
CCL	Connected Component Labelling
CLM	Concept Language Model
CNN	Convolutional Neural Network
CQ	Complex Query
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
D2V	Doc2Vev trained model
Doc2Vec	Paragraph Vectors
ER	Element Ratio
ES	Elasticsearch index
ESA	Explicit Semantic Analysis
eEQC	Extended Equivalence Class
EQC	Equivalence Class
eTC	Extended Type Cluster
FAST	Features from Accelerated Segment Test
FDC	Focused web-Domain Crawler
F1	F-score or F-measure
FT	Fine Tunning
GL	Global Levenshtein
GloVe	Global Vectors for word representation
GLV	GloVe trained model
GPU	Graphics Processing Unit
GSU	Gaussian Sample Uncertainty
GVF	Graph Visualisation Framework
HSV	Hue Saturation Value
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ID	Identifier
IR	Information Retrieval
ISO	International Organization for Standardization
IWCS	IDF re-weighted Word Centroid Similarity
JSON	JavaScript Object Notation
LL	Local Levenshtein
LDA	Latent Dirichlet Allocation
LOD	Linked Open Data
MAP	Mean Average Precision
MER	Matched Element Ratio
MRR	Mean Reciprocal Rank
MST	Minimum Spanning Tree
MXInfAP	Mean Extended Inferred Average Precision
NDCG	Normalised Discounted Cumulative Gain
NLP	Natural Language Processing
OCR	Optical Character Recognition
OPC	Operations Per Character
ORB	Oriented FAST and Rotated BRIEF
PDF	Portable Document Format
Pr	Precision
PSD	Perpendicular Squared Distance
PV-DM	Paragraph Vectors with Distributed Memory
RDF	Resource Description Framework

Abbreviation	Explanation
RDFS	Resource Description Framework Schema
Re	Recall
REST	Representational State Transfer
RGB	Red, Green, and Blue color space
SD	Standard Deviation
SEC	Search-Engine-based web Crawler
SSM	Social Stream Manager
SSOAR	Social Science Open Access Repository
SSOD	Single String Orientation Detection
SVM	Support Vector Machine
SVM-GSU	Support Vector Machine with Gaussian Sample Uncertainty
TC	Type Cluster
TF-IDF	Term Frequency - Inverse Document Frequency
TLM	Transcript Language Model
ToC	Table of Content
TQ	Type Query
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W2V	Word2Vec trained model
WCS	Word Centroid Similarity
WebGL	Web Graphics Library
WevQuery	Web Event Query Tool
WMD	Word Mover's Distance
Word2Vec	Neural network based word embedding
WoS	Web of Science
XML	Extensible Markup Language

1 Introduction

1.1 History of the document

Table 2: Document history

Date	Version
17/01/2017	v0.1: first ToC draft
27/01/2017	v0.2: second ToC draft
15/02/2017	v0.3: ToC ready for QA
13/03/2017	v0.4: content ready for QA
28/03/2017	v0.9: document ready for final QA
31/03/2017	v1.0: final document

1.2 Purpose of the document

This document provides an initial set of technology components developed for the MOVING platform. All technologies described in this document can be but not necessarily have to be integrated in the final MOVING platform. The upcoming deliverables D3.2 (Technologies for MOVING data processing and visualisation v2.0, M24) and D3.3 (Technologies for MOVING data processing and visualisation v3.0, M34) will further extend on this document to collect a final set of technologies well suited for the MOVING platform and its functional requirements elicited in D1.1 (User requirements and specification of the use cases, M12).

1.3 Structure of the document

This document is structured into four main sections encapsulating the different technologies. In Section 2, we describe the common data model. The data acquisition and the data processing components are described in Section 3 and the user logging components in Section 4. Finally, the data visualisation components are described in Section 5. A high-level view of the interaction between components developed in WP 3 as well as their interconnection to the MOVING web application developed in WP 4 is illustrated in Figure 1.

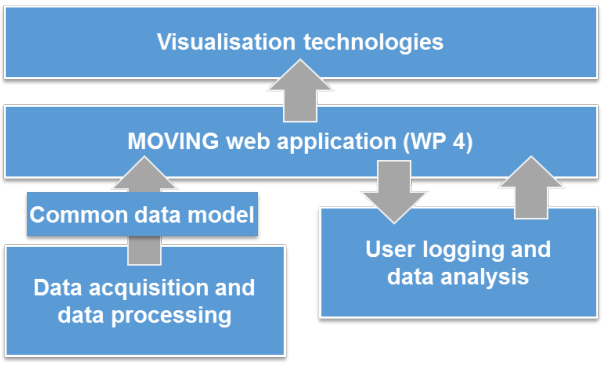


Figure 1: High-level view of the interaction between components developed in WP 3 and the MOVING web application.

2 Common data model

The common data model for MOVING lays the foundation for data integration in the project. The specific challenge is the integration of the variety of data sources like video lectures, publications and metadata from professional publishers. Therefore, we decided on a set of core attributes which should be present for each type of document. To this end, we have identified the core attributes for each document type and added them to a common data structure.

2.1 Model for full-texts and metadata

In MOVING, we have the full-texts and metadata of publications that come from different sources. Metadata may also be harvested from the web such as from the Linked Open Data (LOD) cloud. Those metadata records come with Universal Resource Identifiers (URIs) and possibly have links to related entities such as uniquely identifiable persons and locations. The full-texts are provided in PDF or text format and are stored as raw text as one attribute of the document representation. Additional information is extracted along the full-text (see Section 3.4) or is provided by the original data source. We represent full-text and metadata as shown in Table 3. A short description is given for each attribute. The last column indicates whether a certain attribute needs to be present to be a valid document. We do not describe each attribute in detail but rather focus on attributes which are not self-explanatory. Please note that we describe authors, venues, affiliations and locations not as simple strings but as complex objects. Since those entities extracted from a full-text are usually referred to by their names, we can assume that, e.g. if a document mentions the author, the modelled `author`-object has `name` as mandatory attribute. Such entities extracted from professional metadata can have additional information, e.g. the venue possibly links to a location which comes with geo coordinates. Also, since we collect data from various sources, the same document can have multiple `sourceURLs` and link to multiple `documentURLs` containing the full-text. The time aspect of publications is modelled as interval for compatibility reasons. For example, we regularly crawl webpages (HTML). We model those webpages' "lifespan" as an interval using the attributes `startDate` and `endDate`.

Table 3: Attributes of the common data model used to represent full-texts with metadata.

Attribute	Description	Mandatory
`source`	Dataset name according to Data Management Plan	yes
`docType`	Predefined values denoting the type, e.g. `publication`	yes
`sourceURLs`	List of links to the original source records (e.g. PDF, LOD)	yes
`title`	Document's title	yes
`authors`	List of `author`-objects	yes
`concepts`	Added during indexing process, e.g. using economics thesaurus	yes
`documentURLs`	List of links to the full-text (e.g. PDF or publisher website)	no
`fulltext`	Document's full-text	no
`abstract`	Document's abstract	no
`startDate`	Date document was published	no
`endDate`	Date document became unavailable	no
`venue`	Publishing event (`venue`-object)	no
`license`	License description or name, e.g. Open Access	no
`language`	ISO 639-1 language code	no
`keywords`	List of extracted author keywords	no
`references`	List of extracted author references	no
`author.name`	First and last name	yes
`author.URL`	Link to an external representation, e.g. Wikipedia, LOD record	no
`author.email`	Email address	no
`author.affiliations`	List of `affiliation`-objects associated to the author	no
`affiliation.name`	Name of the organisation	yes
`affiliation.location`	Extracted `location`-object	no
`venue.name`	Name of the publishing event	yes
`venue.URL`	Link to an external representation	no
`venue.startDate`	Start of the event	no
`venue.endDate`	End of the event	no
`venue.volume`	Series/journals etc. that have volumes	no
`venue.issue`	Series/journals etc. that have issues	no
`venue.location`	Extracted `location`-object	no
`location.name`	Name of the location	yes
`location.lat`	Extracted latitude	no
`location.lng`	Extracted longitude	no

2.2 Model for HTML content

HTML pages are provided by the web crawling process (described in Section 3.1). The attributes of the common data model are extracted by an HTML parser to the extent that the official HTML standard provides. Table 4 shows the attributes we are able to extract from HTML documents. The `sourceURLs` and `title` are the most basic attributes and can always be extracted, so they are indicated as mandatory. Please note, a full-text or metadata document may have multiple source URLs, but a webpage only has one URL. Also the attribute `fulltext`, which provides the raw content of the HTML document, is mandatory. Since we conduct regular crawls for certain webpages, we model a "lifespan" for each webpage using `startDate` and `endDate`. The start date denotes when one webpage was crawled the first time. Each HTML document is identified by its URL and the crawling date. When the webpage (referenced by URL) changes its content, the document's end date is set to the current crawl date and a new document is created to represent the current version. It is not guaranteed that the other attributes declared in Table 4 can be extracted.

Table 4: Attributes of the common data model used to represent HTML content.

Attribute	Description	Mandatory
`source`	Web	yes
`docType`	Predefined values denoting the type, e.g. `HTML`	yes
`sourceURLs`	Webpage's URL	yes
`title`	Webpage's title	yes
`fulltext`	Complete raw HTML page	yes
`authors`	List of webpage's `author`-objects	no
`abstract`	Webpage's abstract description	no
`startDate`	Date the webpage was crawled the first time	no
`endDate`	Date the webpage changed	no
`license`	License of the webpage's content	no
`language`	ISO 639-1 language code	no
`keywords`	Keywords extracted from the webpage's content	no
`author.name`	The author's name	yes

2.3 Model for video data

We define the common data model attributes used to represent videos. The attributes are summarised in Table 5. Transcripts and video metadata provide textual information about the video content and allow it to be treated the same manner as text content. Consequently, the `fulltext` attribute stores the available transcripts. The author, venue, affiliation and location objects possibly contain the same information as presented in Section 2.1 and is not repeated here. Concepts contain a list of the top visual concepts extracted from the video after performing the visual analysis described in Section 3.6.

Table 5: Attributes of the common data model used to represent videos.

Attribute	Description	Mandatory
`source`	Crawled videos, video lectures etc.	yes
`docType`	Predefined values denoting the type, e.g. `video-lecture`	yes
`sourceURLs`	URL of the video	yes
`title`	Title of the video	yes
`authors`	List of `author-objects`	no
`abstract`	Video's abstract description	no
`fulltext`	Video's transcripts	no
`startDate`	Video's publication time	no
`endDate`	Date video became unavailable	no
`language`	ISO 639-1 language code	no
`concepts`	Automatically generated top visual concepts list	no

2.4 Integration of the common data model in the MOVING platform

The attributes described above model common representations of documents. Each type of document is distinguishable by the attribute docType. Since for each document type some attributes can have a different usage, the docType attribute additionally works as namespace for the other attributes. Listing 1 shows the aggregated common data model as JSON object which is actively used and maintained for the MOVING web application in WP 4.

Listing 1: Initial common data model to integrate full-texts, metadata, HTML content and video data.

```
{ "identifier":"$MOVING unique$",
  "sourceURLs":["$$"],
  "documentURLs":["$$"],
  "title":"$$",
  "abstract":"$$",
  "fulltext":"$$",
  "authors":[{
     "identifier":"$$",
     "inDocumentIdentifier":"$document.identifier + local ID$",
     "URL":"$$",
     "name":"$first and last$",
     "email":"$$",
     "affiliations":[{
        "name":"$$",
        "location":{
           "name":"$$",
           "lat":"$$",
           "lng":"$$"
        }
     }]
  }],
  "startDate":"$published (ISO 8601, 0 if unknown)$",
  "endDate":"$published (ISO 8601, 0 if unknown)$",
  "venue":{
     "identifier":"$$",
     "URL":"$$",
     "name":"$$",
     "startDate":"$(ISO 8601, 0 if unknown)$",
     "endDate":"$(ISO 8601, 0 if unknown)$",
     "volume":"$$",
     "issue":"$$",
     "location":{
        "name":"$$",
        "lat":"$$",
        "lng":"$$"
     }
  },
  "source":"$Predefined values, e.g. Data Management Plan name$",
  "license":"$Name of the license or license description$",
  "docType":"$Predefined values$",
  "language":"$(ISO 639-1 language codes)$",
  "concepts":["$ Added during indexing process, e.g. using economics
   thesaurus$"],
  "keywords":["$Extracted keywords$"],
  "references":["$Extracted or inserted references$]"
}
```

3 Data acquisition and data processing

3.1 Crawling of social media, websites and videos

3.1.1 Problem statement

The emergence of the world wide web in the 90s and the social media in the 00s has created immense amounts of data accessible to everyone. The sheer size of Internet data though, makes task the of its effective retrieval a challenging one. The typical way of accessing web data is through web search engines, which get their information from web crawlers. A web crawler is an Internet bot that systematically browses (crawls) the web following specific rules. Then the data crawled are indexed for efficient search.

Retrieving data from social media has become a manageable task, since the most popular social media platforms like Facebook, Twitter and YouTube have their own search engines and official APIs, so that developers can easily access them. Twitter has a search API[1], which is part of its REST API, allowing queries against indices of recent popular tweets. Queries may include keywords, user accounts and location, and are limited by rate limit policies. Similar functionality is supported in other social media platforms.

For the retrieval of data from the web, there are two separate requirements to be met in MOVING. One is topic-based search on the web and the other is the crawling of whole websites. Topical crawling was first introduced by Menczer (1997). Chakrabarti, van den Berg, and Dom (1999) used a text classifier to prioritise the crawl frontier of his "focused crawler" to a pre-defined set of topics. Our approach is much simpler as we utilise web search APIs to perform topical search in the web. Web search APIs include Google custom search API [2], Bing Search API [3] and the Faroo web search API [4]. The Google custom search API offers rate-limited access to the Google search engine via its REST API. The Bing Search API offers similar functionality, but does not offer any free access. Faroo has a free web search API that uses peer-to-peer technology and a distributed crawler that stores search data on users' computers instead of a central server. Crawling specific websites requires a web crawler that does not exceed the boundaries of the specified web domain. There are several tools capable of providing the above functionality like the Scrapy framework [5], Mechanize framework [6], Apache Nutch [7], which is a production-ready crawler, and even the GNU wget tool [8].

3.1.2 Method description

Crawling of social media

For crawling of social media, we used the Social Stream Manager (SSM)[9], a readily available open source tool developed for the FP7 project Social Sensor [10], which we configured for MOVING and integrated in the overall crawling architecture of the project (see Section 3.1.4). The SSM monitors a set of several social streams like Twitter, Facebook, Google+ and Youtube to collect incoming content relevant to a keyword, a social media user or a location, using the corresponding API that is supported from each service. The Twitter API works as a real-time service, whereas the others act as polling consumers, which means they perform requests to the network and receive replies periodically. An overview of the SSM architecture is depicted in Figure 2. The SSM uses Feeds as the core piece of information that can be monitored in social media platforms. For example, an account in Twitter or Facebook is a feed. A keyword is also a feed. These feeds are created from the sources inserted by users. The Social Stream Manager is a multi-threaded process that uses a separate thread for the management of each social media platform (Stream). Internally, each platform-specific thread creates new threads, one for each feed that is executed periodically in an adaptive way, based on the activity of each feed. A monitor thread is used to measure the activity of each feed and ensure that the rate limits imposed by each platform are respected.

Since the Facebook API does not support keyword queries, we employ for the MOVING platform only Twitter, Google+ and YouTube streams. The SSM stores the 'Items' (tweets, posts, etc.) (Listing 2) it extracts webpage and multimedia links and stores them in different database collections in MongoDB,

[1] https://dev.twitter.com/rest/public/search, last accessed: 20/03/2017
[2] https://developers.google.com/custom-search/, last accessed: 20/03/2017
[3] https://datamarket.azure.com/dataset/bing/search, last accessed: 20/03/2017
[4] http://www.faroo.com/, last accessed: 20/03/2017
[5] https://scrapy.org/, last accessed: 20/03/2017
[6] http://wwwsearch.sourceforge.net/mechanize/, last accessed: 20/03/2017
[7] https://nutch.apache.org/, last accessed: 20/03/2017
[8] https://www.gnu.org/software/wget/, last accessed: 20/03/2017
[9] https://github.com/MKLab-ITI/mklab-stream-manager, last accessed: 20/03/2017
[10] http://www.socialsensor.eu/, last accessed: 20/03/2017

'Webpages' and 'MediaItems' respectively (Listing 3). Then, the webpages are fetched and converted to the common data model (Section 2) before being sent to our common database, Elasticsearch.

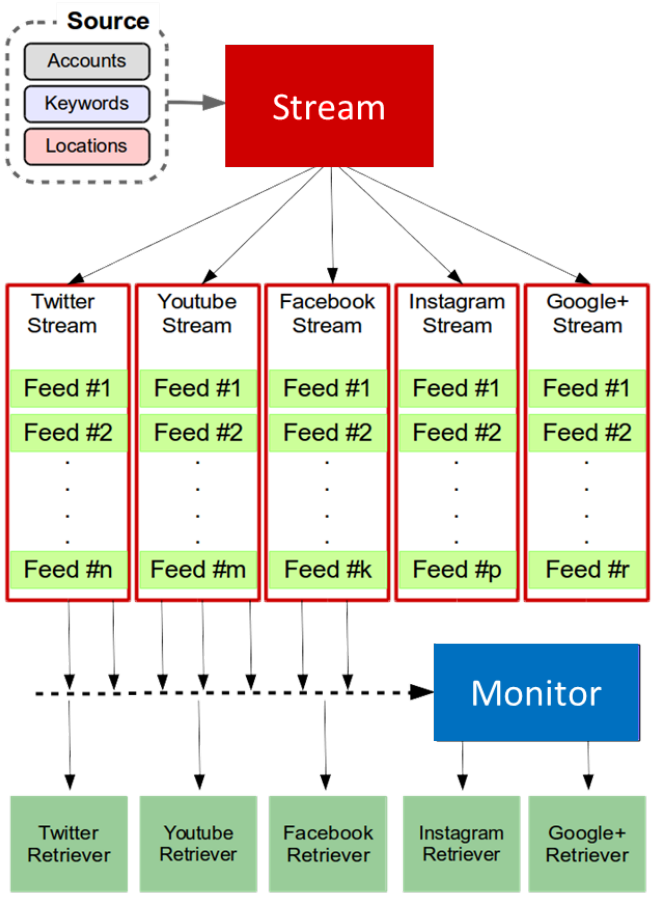


Figure 2: Social Stream Manager's (SSM) components.

Listing 2: The structure of an 'Item' document in MongoDB.

```
{ "_id" : "Twitter#747431957194211328",
  "className" :
   "gr.iti.mklab.framework.abstractions.socialmedia.items.TwitterItem",
  "source" : "Twitter",
  "title" : "Thrilled to hear that \"Social #Entrepreneurship and
   #Innovation\" has been nominated for Management Book Of The Year.
   https://t.co/7hHA75JZ6P",
  "tags" : ["Entrepreneurship", "Innovation" ],
  "uid" : "Twitter#11991612",
  "mentions" : [ ],
  "pageUrl" : "https://twitter.com/kiwanja/statuses/747431957194211328",
  "links" : ["http://yearbook.managers.org.uk/book/social-entrepreneurship
   -and-innovation/"],
  "publicationTime" : NumberLong("1467036639000"),
  "insertionTime" : NumberLong("1467036701609"),
  "language" : "en",
  "original" : true,
  "likes" : NumberLong(2),
  "shares" : NumberLong(0),
  "comments" : NumberLong(0)
}
```

Listing 3: The structure of a 'MediaItem' document in MongoDB.

```
{ "_id" : "Youtube#Vo9KHp04BCc",
  "url" : "https://www.youtube.com/embed/Vo9KHp04BCc",
  "thumbnail" : "https://i.ytimg.com/vi/Vo9KHp04BCc/hqdefault.jpg",
  "source" : "Youtube",
  "reference" : "Youtube#Vo9KHp04BCc",
  "title" : "Patrick van der Duin at CHIFTalk 2016, Rotterdam",
  "description" : "Patrick van der Duin is an assistant professor
   Foresight and Innovation Management at Delft University of Technology
   and Associate Professor Futures Research & Trendwatching at Fontys
   University of Applied Sciences, Academy for Creative Industries. He has
   published in journals such as Futures, Foresight, Technological
   Forecasting & Social Change, and the Journal of Futures Studies. Patrick
   studied macro-economics at the University of Amsterdam and formerly
   worked as a futurist at KPN Research. From September 1st 2016 he will be
   the director of the Netherlands Study Centre for Technology Trends.",
  "tags" : ["Patrick van der Duin", "science fiction", "CHIFT",
   "CHIFTalk"],
  "type" : "video",
  "publicationTime" : NumberLong("1467107303000"),
  "likes" : NumberLong(1),
  "shares" : NumberLong(0),
  "comments" : NumberLong(0),
  "views" : NumberLong(2),
  "ratings" : 0,
  "sentiment" : 0,
  "width" : 480,
  "height" : 360
}
```

Performing topic search in the web using search engines

We developed a Search-Engine-based web Crawler (SEC), which exploits web search APIs to collect webpages relevant to our topics of interest. Currently, we employ only the Google Custom Search API. It is a REST API, rate-limited to 100 calls per day. The results it returns are identical to those of the Google search engine. In order to use the API, we have enabled an API key for authentication and also configured a custom search engine, which is used to tell the API which sites to search (if not all the web). The following example shows a simple request which searches a custom search engine (CX) for the topic car:

`GET https://www.googleapis.com/customsearch/v1?key=KEY&cx=CX&q=car`

The key parameter contains the API key, cx contains an identifier for our custom search engine configuration and q is the search query. We use a scheduler to launch the API calls once a day until the rate limit is reached. The topics are queried to the API sequentially, first retrieving the first results page (1-10) for every topic, then the second (11-20) and so on, until 100 results are retrieved for every topic (the maximum possible). Following the retrieval of the results from the API, every webpage is fetched and converted to the common data model before being sent to Elasticsearch for indexing. The topics are scheduled to be re-searched periodically.

Crawling of specific domains

To perform domain-focused crawling on specific pre-defined websites we have developed the Focused web-Domain Crawler (FDC), which is based on the Scrapy web crawling/scrapping framework. Figure 3 shows an overview of FDC's architecture. The Scrapy project architecture is built around spiders, which are self-contained crawlers given a set of instructions. To meet our requirements, we employ one general purpose spider, the 'General Spider'. Our application periodically checks the database for new imported domains to crawl. When a new domain is found, it launches a general spider to crawl it. The spider starts from an inaugural URL and sends it to the scrapy engine. The engine is responsible for controlling the data flow between all components of the system, and triggering events when certain actions occur. The engine then sends the URL to the 'Downloader', which fetches the webpage and sends the result back to the spider through the engine. Subsequently, the spider performs two tasks on this webpage. First, it extracts the common data

model attributes from the HTML and sends them to the pipeline. The pipeline will then feed these data as a document to the Elasticsearch to be indexed. Second, it extracts the URL links from the page and sends them to the engine, which will schedule their downloading. From there, the process continues recursively until all the URLs of the page are crawled. Scrapy has an internal duplicate URL filter that we have extended to avoid re-crawling the same pages. We have also set Scrapy to comply with robots.txt and set a minimum crawl delay of 1 second for politeness. The application is scheduled to re-crawl the websites periodically. The period is currently set to one month.

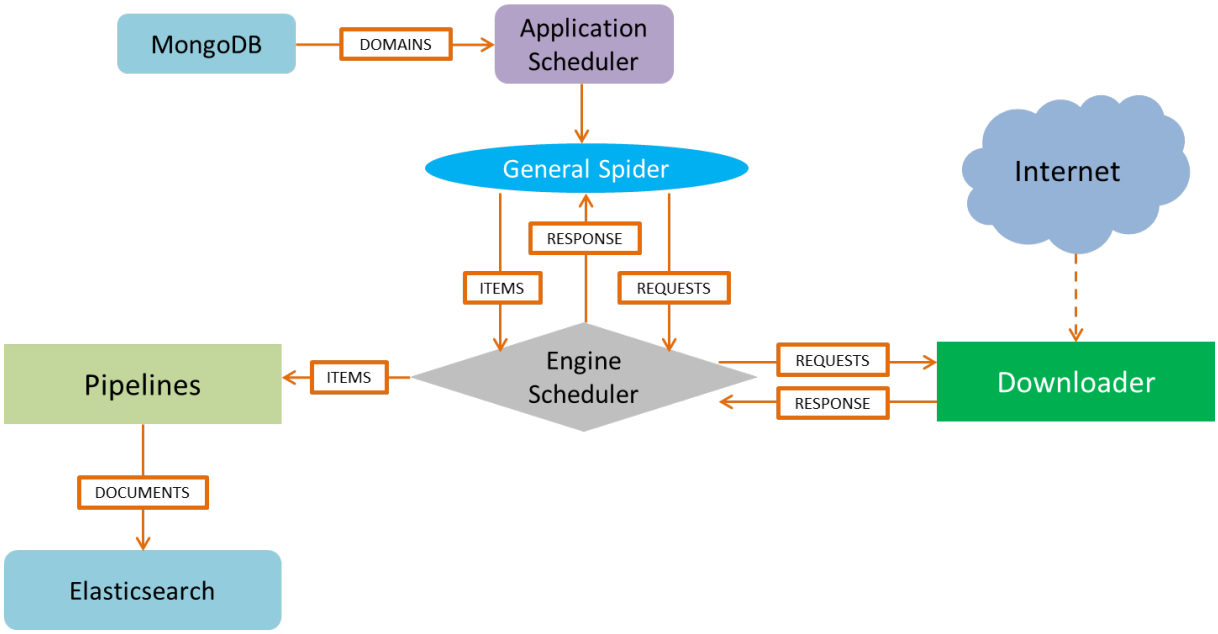


Figure 3: Focused web-domain crawler architecture.

3.1.3 Experimental evaluation and comparison

We quantitatively evaluate the crawling process by counting the number of documents collected by our three crawlers during a period of 24 hours. As input, we have inserted ten topics and four domains (see Table 6). Table 7 shows the number of documents collected by each crawler.

Table 6: Topics and domains used in the experimental evaluation of the three crawlers.

Domains
www.bayer.com
www.telekom.com
www.rwe.com
www.oliverwyman.com

Topics
neural networks
innovation management
big data
information retrieval
computer vision
natural language processing
machine learning
ontology alignment
computational neuroscience
computer aided design

The SSM and the SEC application have each collected less than 1000 documents in a day, as expected, limited by the rate limits imposed by the APIs they use. The FDC managed to collect over 20 thousand documents in the same time and crawled the four domains in their entirety. As one can see in Table 8 the crawl duration for a specific domain is heavily dependent on the number of pages it contains. For rwe.com, the FDC took 11 hours to collect its 17,218 pages, while for bayer.com it only took 25 minutes to crawl its 589 pages.

Table 7: Number of documents collected from each crawler in 24 hours.

Crawler	Documents collected
Search-Engine-based web Crawler (SEC)	762
Social Stream Manager (SSM)	761
Focused web-Domain Crawler (FDC)	23662

Table 8: Number of crawled pages and crawling duration for some of the tested domains.

Domain	Pages crawled	Duration (hr:mn)
www.bayer.com	589	0:25
www.telekom.com	3739	0:50
www.rwe.com	17218	11:21
www.oliverwyman.com	2115	1:15

3.1.4 Implementation, APIs and integration

The full architecture of the MOVING Crawler is visualised in Figure 4. The process pipeline starts from the 'Input UI' (Figure 5), which is used to insert the topics and domains to be crawled. The Input UI uses HTML/javascript in the frontend and python/bottle[11] in the backend to store the topics/domains to the MongoDB database. The 'Input UI' can also be used to remove active topics/domains. The MongoDB serves as a source of input to the crawlers. SSM requires a MongoDB database to store its output data. The fetcher, which is written in python, downloads and feeds the websites extracted from the items to Elasticsearch. Starting the SSM Fetcher, by executing its `main.py` module, also launches the Social Stream Manager. FDC and SEC are both written in python and are started through running their `main.py` modules. To feed the collected webpages to Elasticsearch, we use its REST API, more specifically the Index API [12]. The index API adds or updates a typed JSON document in a specific index, making it searchable.

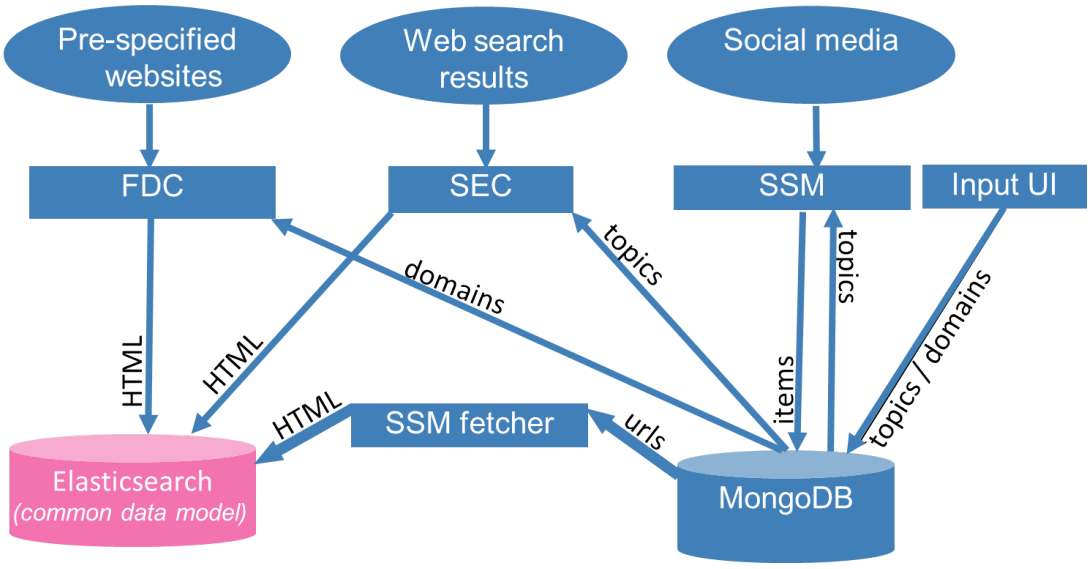


Figure 4: MOVING crawler architecture.

[11] `https://bottlepy.org/docs/dev/`, last accessed: 20/03/2017
[12] `https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-index_.html`, last accessed: 20/03/2017

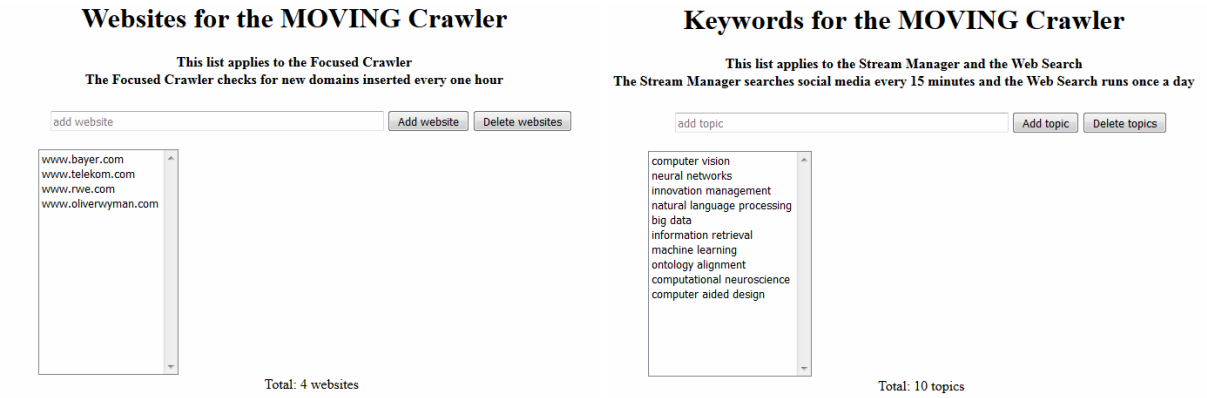


Figure 5: The user interface used to insert topics and domains to the crawlers.

3.2 Assessing word embeddings in practical information retrieval

3.2.1 Problem statement

Word embeddings have become the default representation for text in many neural network architectures and text processing pipelines (Bengio, Ducharme, Vincent, & Janvin, 2003; Bengio, Courville, & Vincent, 2013; Goth, 2016). In contrast to the typical bag-of-words representations, word embeddings are capable of capturing semantic and syntactic relations between the words (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013; Pennington, Socher, & Manning, 2014). So far, they have been successfully employed in various natural language processing tasks such as word analogies, clustering, and classification (Mikolov et al., 2013; Pennington et al., 2014; Kusner, Sun, Kolkin, & Weinberger, 2015; Balikas & Amini, 2016). Word embeddings are recognised as the main reason for Natural Language Processing (NLP) breakout in the last few years (Goth, 2016).

A word embedding is a distributed vector representation for words (Mikolov et al., 2013). Each word is represented by a low-dimensional (compared to the vocabulary size) dense vector, which is learned from raw text data. In several natural language processing architectures such as neural networks these representations serve as first layer for the conversion from raw tokens (words) to a more useful representation. The property that semantically related terms are clustered close to each other in the representation space proves the usefulness of this approach for classification and other NLP tasks. However, transferring the success of word embeddings to the ad-hoc Information Retrieval (IR) task is currently an active research topic. While embedding-based retrieval models could tackle the vocabulary mismatch problem by making use of the embedding's inherent similarity between distinct words, most of them struggle to compete with the prevalent strong baselines, namely TF-IDF (Salton & Buckley, 1988), Okapi BM25 (Robertson, Walker, Hancock-Beaulieu, Gatford, & Payne, 1995) and their relatives.

The majority of practical information retrieval systems rely on an extended boolean model (Salton, Fox, & Wu, 1983). Extended boolean models generalise both standard boolean models and vector space models. These extended boolean models are highly efficient, since the documents can be stored in an inverted index. Thus, the IR system stays responsive even if a huge amount of documents is indexed. Those practical IR systems always employ a binary matching operation on the inverted index to reduce the set of documents, to which the similarity of the query is computed (see Figure 6). However, some advanced techniques based on query expansion and relevance feedback can be safely incorporated in an extended boolean model.

We consider a practical ad-hoc IR task which is composed of two core steps: matching and scoring. In the matching step, documents of the corpus are matched against a query, typically by (binary) term co-occurrence: either the documents contain at least one term of the query or not (boolean OR query). The scoring step consists of ranking these matched documents according to their relevance to the query. As these core IR tasks are different from other NLP tasks, the incorporation of word embeddings is challenging. As we evaluate the suitability of embedding-based retrieval models in a practical context, we fix this matching operation and concentrate on the similarity scoring operation. Additionally, we restrict ourselves to purely unsupervised models. Please note that every retrieval model could be potentially improved by query-relevance information. We also exclude pseudo-relevance feedback, since it is typically not applied in practical IR setups [13].

[13] Pseudo-relevance feedback is not natively included in Apache Lucene, thus SOLR and Elasticsearch

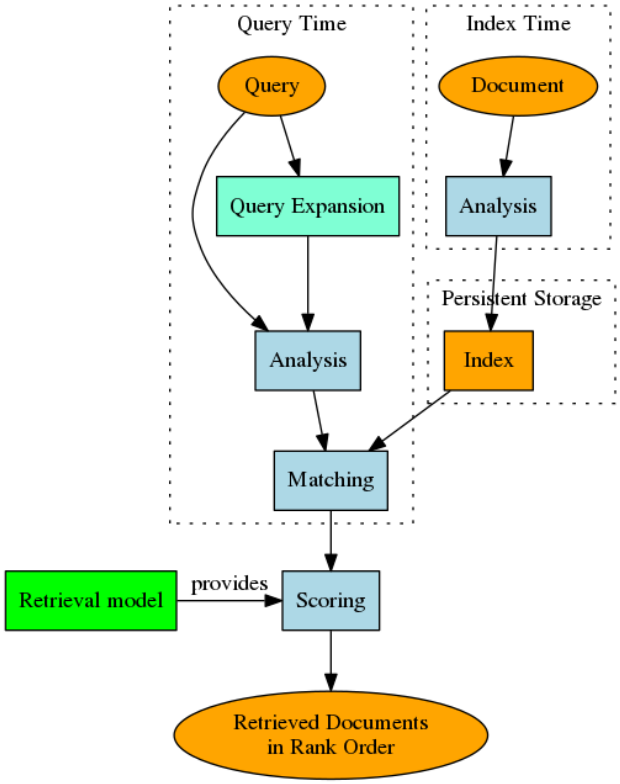


Figure 6: A simplified information retrieval system.

We compare and evaluate several similarity metrics for query-document pairs using word embeddings and assess their suitability in a practical IR system. The considered approaches are Word Centroid Distance and a TF-IDF re-weighted variant, Word Movers Distance (Kusner et al., 2015) and Paragraph Vectors (Le & Mikolov, 2014). Practical IR systems allow treating the fields (`title`, `full-text`, `date`, ...) of a document differently. Thus, we analyse whether the performance of the embedding-based techniques depends on document length. In summary, we will answer the following research questions:

1. Which embedding-based techniques are suitable for practical information retrieval?
2. How does their performance depend on document length?

Related work

Information Retrieval: Extended boolean models such as TF-IDF (Salton & Buckley, 1988) and Okapi BM25 (Robertson et al., 1995) rely on bag-of-words representations, re-weighted by inverse document frequency. While still considered a strong baseline, these models (along with others) struggle to deal with two typical difficulties of the IR task: *term dependencies* and *vocabulary mismatch*. The former means the independence assumption of terms does not hold in natural language, the latter describes the problem of disregarding semantically related terms, when exact matching fails. Early approaches to tackle the term dependency problem involved word n-gram models. However, Fagan (1987) showed that these approaches are not so successful, most probably caused by higher sparsity of the more complex n-grams. (Zhai, 2008). There are several probabilistic models that rely on language modelling. The documents are ranked either by each document language model's probability of generating the query or by the probability of generating the document, given the query language model (Beeferman, Berger, & Lafferty, 1999; Ponte & Croft, 1998; Miller, Leek, & Schwartz, 1999; Hiemstra, 1998). The divergence from randomness retrieval model was shown to outperform BM25 consistently on several TREC collections (Amati & van Rijsbergen, 2002).

Topic Models: The idea of distributed representations for documents goes back to latent semantic indexing by Furnas et al. (1988). It relies on a singular value decomposition of the term-document matrix. It was extended with a probabilistic variant by Hofmann (1999). Finally, Blei, Ng, and Jordan (2003) proposed the probabilistic topic model Latent Dirichlet Allocation (LDA) in 2003.

Word Embeddings: Bengio et al. (2003) first introduced a statistical language model based on neural networks, so-called neural net language models, forming the basis for word embeddings learned by a neural network. Mikolov et al. (2013) proposed a neural network based word embedding (Word2Vec), in which the representations are learned by training to reconstruct each word's context (skip-gram model). The success of the Word2Vec model relies on skip-gram training with negative sampling, an efficient training algorithm (not involving dense matrix multiplication). Beside other word embeddings (Collobert & Weston, 2008; Mnih, Yuecheng, & Hinton, 2009; Turian, Ratinov, & Bengio, 2010), it is notable that a word embedding may also be computed by directly factorising the global co-occurrence matrix as done with GloVe (Pennington et al., 2014). Le and Mikolov (2014) further extend the Word2Vec approach by additionally modelling representations of whole documents (Doc2Vec). Their experiments indicate that these distributed representations are useful for information retrieval tasks. However, the evaluation task is to find one relevant document out of three (given 80% training data), which is not a classical ad-hoc query task as it is considered in MOVING.

Word Embeddings for IR: Clinchant and Perronnin (2013) proposed a method for aggregating word vectors with the Fisher kernel to a document level. The authors applied their approach in ad-hoc retrieval outperforming Latent Semantic Indexing, but not TF-IDF or divergence from randomness. Zheng and Callan (2015) learn to re-weight word embeddings using BM25 in a supervised context. Kusner et al. (2015) proposed the Word Mover's distance, a similarity metric between documents based on word embeddings. Inspired by the Earth Mover's distance, the Word Mover's distances solves an optimisation problem for the minimum cost of transportation between the words of two documents. The cost of moving from a single word to another is the cosine distance of their respective word vectors. Recently, Zamani and Croft (2016) proposed embedding based query language models, a dedicated retrieval technique based on word embeddings which thrives to tackle the vocabulary mismatch problem by incorporating word embeddings into query language models.

3.2.2 Method description

In the following, we will sketch the two major algorithms for learning a word embedding: Word2Vec and GloVe. Then, we will depict the application of embedding-based techniques for computing a query-document similarity including the novel IDF re-weighted word centroid distance. A graphical overview of the interaction of the techniques is given in Figure 7.

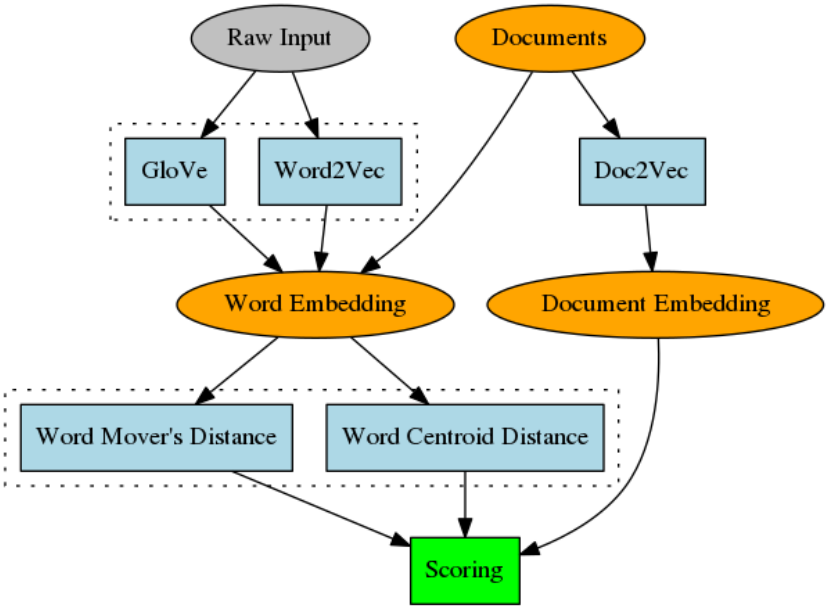


Figure 7: Overview of techniques for embedding-based retrieval.

Notation: We will use the following notation throughout the section:

$\boldsymbol{A}_i$ Row i of matrix $\boldsymbol{A}$ as a column vector (i. e. transposed row vector).

$\boldsymbol{X} \in \mathbb{R}^{n \times m}$ nBoW (l2-normalised bag of words) representation of n documents with m vocabulary words

$\boldsymbol{q} \in \mathbb{R}^m$ nBoW representation of the query

$\boldsymbol{W} \in \mathbb{R}^{m \times h}$ Word embedding consisting of m word vectors with h dimensions

$k \in \mathbb{N}$ number of documents to retrieve

From Words to Embeddings

In a first step, a word embedding is learned from raw text. The most prominent techniques are Skip-Gram Negative Sampling (Word2Vec) by Mikolov et al. (2013) (based on a shallow neural network) and Global Word Vectors by Pennington et al. (2014) (based on a global term co-occurrence matrix).

Skip-Gram Negative Sampling: Word2Vec (Mikolov et al., 2013) is an unsupervised algorithm learning a dense vector representation for each word from a corpus. One distinguishes between continuous bag of word (CBoW) and the skip-gram model. While the former aims to predict the centre word from its context, the latter aims to predict the context from the centre word. More formally, given a sequence of $w_1, \ldots, w_T$ words and a context window size, the training objective is to maximise:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(\boldsymbol{W}_{t+j} | W_t)$$

The conditional probability $p(w_o|w_i)$ is typically given by the (hierarchical) softmax distribution. Training with negative sampling is an approximation for the full softmax. During training, negative samples are drawn from the vocabulary (which do not appear in the actual context, but probably appear in similar contexts). A simplified algorithmic structure for skip-gram negative sampling can be defined as follows: Given a vocabulary $\mathbb{V} \subset \mathbb{N}$ and a stream of l words $\boldsymbol{w}$.

1. Let w_T be target word with context $\mathbb{C} = \{w_{T-c}, \ldots, w_{T-1}, w_{T+1}, \ldots, w_{T+c}\}$
2. Look up word vector $\boldsymbol{h} := \boldsymbol{W}_{w_T}$ for target word w_T
3. Predict via logistic regression from word vector $\boldsymbol{h}$ with:
   - positive examples: context words $\mathbb{C}$
   - negative examples: sampled from $\mathbb{V} \setminus \mathbb{C}$
4. Update word vector $\boldsymbol{h}$ by back-propagation.
5. Repeat with next word $T = T + 1$ while $T < l$

In the remainder of this work, we consider the skip-gram variant trained with negative sampling for Word2Vec.

Global Word Vectors: Global Word Vectors (GloVe) (Pennington et al., 2014) is an algorithm to learn word embeddings by directly factorising the term co-occurrence matrix. More specific, the training objective (See Equation 1) is to learn word vectors $\boldsymbol{W}$ whose dot product equals the joint probability given by co-occurrence matrix $\boldsymbol{C}$.

$$\boldsymbol{w}_i^T \tilde{\boldsymbol{w}}_k + b_i + \tilde{b}_k \stackrel{!}{=} \log(1 + \boldsymbol{C}_{ik}) \tag{1}$$

The obtained word vectors can be used as basis for the document-level similarities: word centroid distance and Word Mover's distance. When the global co-occurrence matrix is discarded (typically after initial learning of the word vectors), up-training of an existing model with GloVe is not possible.

Paragraph Vectors: Paragraph vectors (or Doc2Vec) (Le & Mikolov, 2014) extends the Word2Vec model by a paragraph id as an additional input. A dense document vector is learned for each document in addition to the word vectors. We consider the distributed memory approach (PV-DM), which models the paragraph identifier as if it was an artificial word token from the context. Given the query, a paragraph vector model is capable of inferring a document vector, whose cosine similarity to known document vectors can be used as a similarity metric. With fixed weights, we perform five training epochs with a linearly decaying learning rate from 0.1 to 0.0001

With Embeddings to Retrieval

Word Centroid Similarity (WCS): Given the term occurrence matrix $\boldsymbol{X}$, where X_{ij} is the number of occurrences of word j in document i, we compute the centroid of the document as follows. First, we normalise the each row $\boldsymbol{X}_i$ to unit l2-norm (nBoW representation). The word j also corresponds to the respective word vector in the embedding w_j. Second, we obtain the word centroid representation of documents by matrix multiplication $\boldsymbol{C} = \boldsymbol{X} \cdot \boldsymbol{W} \in \mathbb{R}^{n \times h}$. Now, the cosine similarity of the query to the centroids provides a notion of similarity:

$$\mathrm{WCS}(q,i) = \frac{\left(\boldsymbol{q}^T \cdot \boldsymbol{W}\right) \cdot \boldsymbol{c}_i}{||\boldsymbol{q}^T \cdot \boldsymbol{W}|| \cdot ||\boldsymbol{c}_i||}$$

The employed norm $||\cdot||$ is the l2-norm. Given a query, the documents are ranked by descending cosine similarity to the query. In case of length-normalised word frequency vectors, the resulting ranking of word centroid similarity is equivalent to the one of word centroid distance mentioned by Kusner et al. (2015).

IDF re-weighted Word Centroid Similarity (IWCS): In addition, we propose a variant of the WCS, where the documents' bags of words are re-weighted by inverse document frequency as in TF-IDF, before the centroids are computed. Consider a bag-of-words representation $\boldsymbol{X}$ of the documents, where X_{ij} corresponds to the number of occurrences of word i in document j. We first re-weight $\boldsymbol{X}$ with respect to inverse document frequency:

$$X'_{ij} = X_{ij} \cdot \mathrm{idf}(j)$$
$$\mathrm{idf}(j) = \log \frac{1+n}{1+\mathrm{df}(D,j)}$$

The document frequency $\mathrm{df}(D,j)$ is the number of documents that contain word j. Then, we again normalise the rows of $\boldsymbol{X}$ to unit l2-norm and compute the centroids: $\boldsymbol{C} = \boldsymbol{X'} \cdot \boldsymbol{W}$. Finally, we compute the cosine similarity to the query and rank the results in descending order (as in the WCS case).

Word Mover's Distance (WMD): The Word Mover's Distance is a distance metric between two documents. The cumulative cost of moving the words of one document to another document is minimised. The cost function for moving from one word to another is defined as the euclidean distance between the word vectors $c(i,j) = \left|\left|\boldsymbol{w_i} - \boldsymbol{w_j}\right|\right|_{\mathrm{l2}}$. The minimisation problem is constrained, such that all words of the source document and the destination document must be taken into account. The resulting transportation problem can be formalised as the following linear program, where $\boldsymbol{T}_{i,j}$ denotes how much of the word i in the source document is moved to word j of the destination document (Kusner et al., 2015):

$$\min_{T \geq 0} \sum_{i,j=1}^{m} \boldsymbol{T}_{i,j} \cdot c(i,j)$$
$$\sum_{j=1}^{n} \boldsymbol{T}_{i,j} = d_i \forall i \in \{1,\ldots,m\}$$
$$\sum_{i=1}^{n} \boldsymbol{T}_{i,j} = d_j \forall j \in \{1,\ldots,m\}$$

It can be shown that the WCS as well as the Relaxed Word Mover's Distance (a variant of WMD that leaves out one of the constraints) is a lower bound for the Word Mover's distance. These lower bounds can be used to reduce the computational cost (Kusner et al., 2015). In addition to the full Word Mover's distance (WMD), we also evaluate a variant which takes the top k documents returned by IWCS and re-ranks them according to Word Mover's distance (IWCS-WMD).

3.2.3 Experimental evaluation and comparison

Experimental setup

Given a collection of documents D, a collection of queries Q and relevance scores for each query-document pair $\mathscr{R}: Q \times D \to \mathbb{N}$ (the gold standard), the task is to return a ranked list of k (preferably) relevant documents. We evaluate these results according to $\mathscr{R}$. Depending on the dataset, the values of $\mathscr{R}$ can be restricted to binary $\{0,1\} \subset \mathbb{N}$, otherwise higher values indicate a higher relevancy. Since we are interested in the performance of the retrieval models in a practical setting, we put the following constraints on the retrieval models:

- We perform a binary matching operation, assuming the query is composed as an or-query.
- We make no assumptions about the queries when indexing documents.

In this setting, we compare the performance of the embedding-based techniques with respect to the document's field (e.g. `title`, `abstract`, `full-text`) and with respect to the query length. As query sets we compare a set of short (single term or phrase) to a set of longer queries (sentences). We evaluate the embedding-based retrieval models WCS, IWCS, IWCS-WMD for both embedding algorithms Word2Vec and GloVe . Additionally, we evaluate the cosine distance of Doc2Vec's inferred document vectors.

Datasets
NTCIR2: The NTCIR2 dataset consists of $134,978$ documents and 49 topics. The documents are composed of a `title` and an `abstract` field. The topics consist of the fields `title`, `description` and `narrative`. From these we use the `title` as short query and the `description` as *long* query. Additionally, two sets of relevance scores are provided that associate topics and documents (binary). From these we chose the second set of relevance scores `rel2` with 43.6 relevant documents per query. The relevance scores of the first set are always included, which results in a higher diversity for the ranking task. The relevancy judgements are not complete, i. e. there are query-document pairs for which no judgement is given. We assign these documents a relevancy of zero, when evaluating the models.

Economics: The economics dataset is the subset of $61,792$ documents of the ZBWEconomicsDataset (see D6.2: Data Management Plan) which is manually annotated by domain experts using concepts from a domain specific thesaurus. It covers $4,518$ topics with an average of 72.98 (SD: 329) relevant documents per query. The documents are scientific publications from the economics domain. As topics, we use the domain-specific thesaurus concepts. A concept in the thesaurus consists of one preferred label and several alternative label. We employ the preferred labels of the concepts as queries. Hence, we consider a document being relevant to a topic, if and only if the document is annotated with the corresponding concept.

Reuters: The Reuters dataset consists of $100,000$ documents and 102 topics from the news domain. Once again the documents were annotated with one or more of the topics by humans. On average, there are $3,143$ (SD: $6,316$) relevant documents per topic. Each document consists of a `title` and a `full-text` field. The descriptor label of a topic consist of two to three words, i. e. they are short queries. We employ the descriptor label as query for which the assignment of the label to the document resembles relevancy.

Embedding Models
Following the results of Mikolov et al. (2013) and Kusner et al. (2015), employing a well-trained general purpose embedding model is preferable over a corpus-specific model (caused by the surplus in diversity of contexts for each word). For this reason, and for the sake of a consistent comparison over the datasets, we employ pre-trained general purpose word embeddings. Thus, the evaluation is not sensible to the dataset and its specific training procedure (hyper-parameters are often sensible to the training corpus). Although the absolute performance of the embedding-based techniques may be further increased by training a corpus-specific model, we assume that the relative performance of the different similarities would not differ as long as the model itself is trained properly. As representative for Word2Vec, we employ the popular `GoogleNews` model, while for GloVe we employ a similar model (vocabulary size of 2.2 billion, vectors of 300 dimensions, cased analysis), that is trained on the Common Crawl [14]. As treatment for out-of-vocabulary words, we found that ignoring them results in better over-all performance than initialising them with close-to-zero random vectors or up-training the missing words.

Table 9: Abbreviation, corpus, word count, vocabulary size, dimensionality, analysis and training algorithm of the pre-trained models.

Model	Corpus	Tokens	Vocab	Dim	Analysis	Training
W2V	GoogleNews	$100 \cdot 10^9$	$3 \cdot 10^6$	300	cased	Word2Vec
GLV	CommonCrawl	$840 \cdot 10^9$	$2.2 \cdot 10^6$	300	cased	GloVe
D2V	Wikipedia	$224 \cdot 10^9$	$3 \cdot 10^6$	1000	uncased	PV-DM

[14] A dataset of crawled web data from `https://commoncrawl.org/`, last accessed: 28/03/2017

Preprocessing
To obtain a meaningful and comparable evaluation over all the different datasets, we use the same preprocessing steps for all retrieval models: First, we transform the raw string into lower case. Second, we tokenise the string by splitting it into words of at least two characters length, while treating any non-word character as delimiter. Finally, we remove common English stop words. To keep complexity under control, we do not apply stemming and only consider uni-gram models. Furthermore, we do not remove queries that contain out-of-vocabulary words.

Evaluation
We consider three evaluation metrics: Mean Average Precision (MAP), Normalised Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR). For all metrics, we limit the considered documents to the top $k = 20$ retrieved documents, resembling the top pages of a typical web search IR task. Let D be the set of documents, Q the set of queries, and $\mathscr{R} : Q \times D \to \mathbb{N}$ the relevance score of a document for a query. Then a retrieval model can be described as $M : Q \to D^k, q \mapsto y$ with $y \in D^k$ being the top-k retrieved documents in rank order. Thus the multiset of results for queries Q and a retrieval model M can be written as:

$$R_{M,Q} = \left\{ (\mathscr{R}(q,d))_{d \in M(q)} \mid q \in Q \right\}$$

For a proper definition of the metrics, we operate on these sets $R_{M,Q}$.

Mean Average Precision (MAP): We compute average precision as the mean of average precision values over the result set $R_{M,Q}$:

$$\text{Precision}(r,k) = \frac{|\{r_i \in r \mid r_i > 0\}|}{|k|}$$

$$\text{AP}(r,k) = \frac{1}{|r|} \sum_{i=1}^{k} Precision((r_1, \ldots, r_i), i)$$

$$\text{MAP}(R_{M,Q},k) = \frac{1}{|Q|} \sum_{r \in R_{M,Q}} \text{AP}(r,k)$$

Mean Reciprocal Rank (MRR): The reciprocal rank of a query result is the fraction of the index of the first relevant document.

$$\text{MRR}(R_{M,Q},k) = \frac{1}{|Q|} \sum_{r \in R_{M,Q}} \frac{1}{\min\{i \mid r_i > 0\}}$$

In case none of the retrieved documents is relevant, the reciprocal rank is set to zero.

Normalised Discounted Cumulative Gain (NDCG): We compute the NDCG for a single result list as follows:

$$\text{DCG}(r,k) = r_1 + \sum_{i=2}^{k} \frac{r_i}{\log_2 i}$$

$$\text{NDCG}_q(r,k) = \frac{\text{DCG}(r,k)}{\text{IDCG}_{q,k}}$$

where $\text{IDCG}_{\text{q,k}}$ is the best possible (ideal) *DCG* for the specific query q with respect to the gold standard $\mathscr{R}$. In case there are more relevant documents than k, the IDCG is also computed on the truncated optimal results. Once again, we average NDCG over the queries, providing mean and standard deviation.

Results
NTCIR2: Considering the results for the NTCIR2 dataset, we inspect four configurations of either `title` or `abstract` and either short (See Table 10) or long (See Table 11) queries. We observe that using the `title` field leads to better results in all metrics and for all techniques. The TF-IDF baseline yields better results in terms of MAP on the `title` field than on the `abstract` field (compare .35 MAP to .46 with short queries, and

.35 MAP to .40 with long ones). For the two word embedding models, we observe that using the Word2Vec embedding leads to better results in all metrics for IWCS, while the relationship is inverted for the WMD re-ranked variant. Still, both variants of the Word Mover's distance perform consistently worse than IWCS as query-document similarity. The TF-IDF baseline is outperformed by IWCS in terms of MAP in three out of four configurations. Still, the margin is rather small (ranging from .01 to .02). In terms of MRR, the baseline could only be outperformed in one configuration by IWCS with a difference of .01.

Table 10: Results with respect to the evaluation metrics Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Normalised Discounted Cumulative Gain (NDCG) for the NTCIR2 dataset using short queries and either the title or the full-text field, k=20.

	`title`			`full-text`		
Technique	**MAP**	**MRR**	**NDCG**	**MAP**	**MRR**	**NDCG**
TF-IDF	**.46** (.38)	**.55** (.45)	**.19** (.18)	.35 (.37)	.41 (.43)	**.18** (.20)
WCS_{GLV}	.37 (.36)	.42 (.42)	.16 (.18)	.29 (.31)	.40 (.43)	.15 (.17)
WCS_{W2V}	.33 (.34)	.35 (.38)	.14 (.16)	.33 (.35)	.39 (.43)	.13 (.15)
$IWCS_{GLV}$	.41 (.36)	.49 (.44)	.18 (.18)	.32 (.32)	.39 (.41)	.17 (.18)
$IWCS_{W2V}$	.38 (.35)	.45 (.43)	.17 (.18)	**.36** (.34)	**.42** (.41)	.17 (.18)
$IWCS\text{-}WMD_{GLV}$	.35 (.32)	.40 (.38)	.17 (.17)	.35 (.36)	.41 (.42)	.17 (.18)
$IWCS\text{-}WMD_{W2V}$	.30 (.31)	.34 (.37)	.15 (.17)	.29 (.32)	.33 (.39)	.15 (.17)
WMD_{GLV}	.25 (.33)	.27 (.37)	.11 (.17)	.18 (.27)	.21 (.33)	.08 (.14)
WMD_{W2V}	.27 (.35)	.29 (.40)	.11 (.16)	.22 (.29)	.24 (.34)	.10 (.14)
D2V	.27 (.32)	.33 (.39)	.13 (.16)	.29 (.34)	.35 (.42)	.13 (.16)

Table 11: Results with respect to the evaluation metrics Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Normalised Discounted Cumulative Gain (NDCG) for the NTCIR2 dataset using long queries and either the title or the full-text field, k=20.

	`title`			`abstract`		
Metric	**MAP**	**MRR**	**NDCG**	**MAP**	**MRR**	**NDCG**
TF-IDF	.40 (.29)	**.51**(.39)	**.20** (.15)	.35 (.32)	**.47** (.43)	**.20** (.21)
WCS_{GLV}	.29 (.29)	.38 (.41)	.15 (.16)	.27 (.26)	.35 (.37)	.14 (.14)
WCS_{W2V}	.30 (.26)	.38 (.38)	.15 (.15)	.30 (.32)	.37 (.41)	.13 (.14)
$IWCS_{GLV}$	.37 (.34)	.45 (.43)	.17 (.16)	.33 (.30)	.44 (.41)	.16 (.16)
$IWCS_{W2V}$	.41 (.35)	.50 (.41)	.19 (.15)	**.36** (.33)	**.47** (.43)	.17 (.16)
$IWCS\text{-}WMD_{GLV}$	**.42** (.36)	.50 (.44)	.17 (.14)	.30 (.30)	.37 (.38)	.17 (.18)
$IWCS\text{-}WMD_{W2V}$	.40 (.31)	**.51** (.41)	.18 (.14)	.35 (.34)	.40 (.41)	.16 (.16)
WMD_{GLV}	.10 (.22)	.12 (.26)	.04 (.08)	.12 (.21)	.14 (.25)	.06 (.10)
WMD_{W2V}	.22 (.33)	.25 (.39)	.08 (.11)	.30 (.32)	.37 (.41)	.13 (.14)
D2V	.24 (.31)	.27 (.37)	.11 (.16)	.16 (.25)	.19 (.31)	.08 (.11)

Economics: For the economics dataset (see Table 12), we observe that once again the retrieval over titles yields consistently higher metric values in terms of MAP, MRR and NDCG. Considering the `title` field, the IWCS is similar to the baseline in terms of MAP (.37), while the MRR and NDCG values are slightly higher for IWCS than for WCS (.01). In case of `full-text`, no embedding-based technique could outperform the baseline. Doc2Vec inference is the closest competitor with .28 compared to .34 MAP of the baseline.

Reuters: Considering the results for the Reuters data set (see Table 13), we observe that IWCS outperforms the baseline in case of `title` as well as `full-text` fields. The IWCS attains a MAP of .60 compared to .52 of TF-IDF ($\approx 15\%$ relative improvement). The results for the two embeddings Word2Vec and GloVe are more or less tied in all cases. In case of full-text with the Word2Vec model, re-weighting the top k documents with WMD could slightly improve the MAP (.56 compared to .55), while the NDCG is equal to IWCS and the MRR is slightly lower (.58 of TF-IDF compared to .60).

Table 12: Results with respect to the evaluation metrics Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Normalised Discounted Cumulative Gain (NDCG) for the economics dataset using either the title or the full-text field, k=20.

	title			full-text		
Metric	**MAP**	**MRR**	**NDCG**	**MAP**	**MRR**	**NDCG**
TF-IDF	**.37** (.38)	.42 (.44)	.26 (.30)	**.34** (.35)	**.40** (.43)	**.26** (.30)
WCS_{GLV}	.36 (.37)	.42 (.44)	.25 (.29)	.21 (.29)	.25 (.36)	.13 (.19)
WCS_{W2V}	.36 (.37)	.41 (.43)	.25 (.29)	.26 (.31)	.32 (.40)	.19 (.24)
$IWCS_{GLV}$	**.37** (.37)	**.43** (.43)	.26 (.29)	.23 (.30)	.28 (.37)	.16 (.22)
$IWCS_{W2V}$	**.37** (.37)	**.43** (.43)	**.27** (.30)	.26 (.31)	.32 (.40)	.19 (.24)
IWCS-WMD_{GLV}	.33 (.35)	.38 (.41)	.25 (.28)		D. N. F.[15]	
IWCS-WMD_{W2V}	.32 (.34)	.36 (.41)	.25 (.28)		D. N. F.[15]	
WMD_{GLV}	.28 (.34)	.32 (.41)	.19 (.27)		D. N. F.[15]	
WMD_{W2V}	.27 (.34)	.31 (.41)	.19 (.27)		D. N. F.[15]	
D2V	.30 (.36)	.35 (.42)	.21 (.28)	.28 (.31)	.33 (.39)	.22 (.26)

Table 13: Results with respect to the evaluation metrics Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Normalised Discounted Cumulative Gain (NDCG) for the Reuters dataset using either the title or the full-text field, k=20.

	title			full-text		
Metric	**MAP**	**MRR**	**NDCG**	**MAP**	**MRR**	**NDCG**
TF-IDF	.52 (.35)	.61 (.43)	.41 (.32)	.51 (.37)	.58 (.43)	.44 (.36)
WCS_{GLV}	.55 (.31)	.63 (.40)	.42 (.29)	.51 (.33)	.60 (.41)	.44 (.33)
WCS_{W2V}	.54 (.33)	.63 (.41)	.43 (.31)	.52 (.35)	.57 (.41)	.46 (.35)
$IWCS_{GLV}$	.58 (.31)	**.69** (.39)	.45 (.29)	.54 (.34)	**.63** (.41)	.47 (.33)
$IWCS_{W2V}$	**.60** (.33)	**.69** (.40)	**.47** (.32)	.55 (.35)	.60 (.41)	**.49** (.36)
IWCS-WMD_{GLV}	.54 (.30)	.62 (.39)	.43 (.49)	.55 (.34)	.61 (.41)	.46 (.33)
IWCS-WMD_{W2V}	.54 (.33)	.58 (.40)	.44 (.32)	**.56** (.37)	.58 (.42)	**.49** (.37)
WMD_{GLV}	.49 (.32)	.54 (.39)	.38 (.29)	.43 (.32)	.50 (.41)	.37 (.31)
WMD_{W2V}	.48 (.34)	.53 (.41)	.39 (.31)	.41 (.34)	.45 (.41)	.33 (.32)
D2V	.48 (.32)	.55 (.41)	.36 (.30)	.43 (.33)	.52 (.43)	.36 (.32)

Discussion

Comparing the `full-text` to the `title` field, we notice that the ranking quality (in terms of MAP, MRR and NDCG) is higher for the `title` field, consistently over all datasets and over all metrics (the only exception being the NDCG on the Reuters dataset). We can only infer that the surplus of information from the full-text does not help the scoring algorithms to produce a better ranking, but rather clutters the relevancy to the information need of the query. The results for the NTCIR2 dataset show that the behaviour is similar on the query side. Longer queries lead to lower performance in all evaluation metrics.

For the embedding-based retrieval models we observe that the newly proposed IWCS outperforms all other techniques. The only exception is the Reuters data set, on which the additional re-ranking by WMD (IWCS-WMD) slightly improves the attained MAP by 0.01 Comparing WCS to IWCS, we observe that the aggregation of IDF re-weighted term-document vectors lead to better performance in all metrics over all datasets. This implies that it is valuable to reduce the impact of common words in the corpus, also in case of word embeddings. Comparing IWCS to the TF-IDF baseline, we observe that the two techniques yield similar performances in all metrics. The two techniques have in common that both re-weight the terms by inverse document frequency.

We furthermore investigate in which cases the IWCS has an advantage over TF-IDF: assume a document containing a high amount of occurrences of the word `automobile` and query consisting of the term `car`. The document would be scored by TF-IDF relatively low since the term `car` does not occur frequently in the document. IWCS would score the document higher because the vector representations for `car` and `automobile` are close in the embedding space. Especially on the Reuters dataset from the news domain, the IWCS managed to outperform the baseline with a relative percentage of 15% (MAP .60 compared to .52). We assume,

[15]We cancelled the experiments after 500 hours.

that on the one hand the scientific datasets contain more domain-specific words that are not included in the embedding models. On the other hand, most of the words of the Reuters dataset from the news domain are contained in the word embeddings because they were also trained on news data.

The embedding algorithms Word2Vec and GloVe produce similar results. One theoretical advantage of Word2Vec is that it can be up-trained. However, up-training did not increase the performance in our case. Still, up-training might be valuable in an incrementally growing corpus of documents as in a typical practical IR setting. Without up-training, new and possibly domain specific words that carry valuable semantic information, could never contribute to a document's representation. Please note that in contrast to GloVe, Word2Vec is capable of learning a word embedding iteratively (also from a base-model that is learned by GloVe). During the experiments, we gained the following valuable insights:

- Ignoring out-of-vocabulary words yields a higher performance instead of initialising them by random vectors.
- Up-training of missing word vectors with frozen original vectors does not improve retrieval performance of embedding-based techniques (up to 100 epochs with skip-gram negative sampling).
- All-But-The-Top embedding post-processing (Mu, Bhat, & Viswanath, 2017) does not improve the performance of embedding-based retrieval models.

Considering the WMD, we inspect the relative performance of IWCS and IWCS-WMD in detail, in order to gain insight on their relation. Please recall, that IWCS-WMD takes the top k documents returned by IWCS and re-ranks them with respect to Word Mover's distance. While IWCS-WMD is able to improve the result of IWCS in case of long queries on the NTCIR2 dataset by .01, the performance of IWCS-WMD is in general lower than the one of IWCS. Therefore, we conclude that the WMD is not worth the additional computational effort in a time-sensitive practical IR setting. When comparing IWCS-WMD to the full Word Mover's distance (WMD), we notice a considerable drop in performance. The considered additional documents by WMD are not helpful for the ranking quality of the results. As the computational effort is even higher in this case, we cannot propose the usage unmodified WMD for practical IR.

Regarding Doc2Vec inference, we observe that in general more words on both the query and the document side, lead to a lower ranking quality. We assume that the semantics of the queries or documents are cluttered by numerous of too generic words. The only exception is the configuration of short queries and abstracts of the NTCIR2 dataset. In this case Doc2Vec was the only technique whose performance could be improved with using abstracts instead of titles. This behaviour can be explained by Doc2Vec's more sophisticated approach of creating a document representation by inference. However, Doc2Vec inference was not able to attain a ranking quality competitive to IWCS in our experiments.

As a limitation, we note that all embedding-based retrieval models do not tackle the 'complete' vocabulary mismatch problem (the document does not contain a single word of the query). Leaving out the matching operation decreases the performance of all embedding-based techniques considerably.

Conclusion
We showed that word embeddings can be successfully employed in a practical information retrieval setting. The proposed cosine distance of aggregated IDF re-weighted word vectors is competitive to the TF-IDF baseline and even outperforms it in case of the news domain with a relative percentage of 15%.

3.2.4 Implementation APIs and integration

We evaluated word embeddings for information retrieval in a practical setting. We concluded that the IDF re-weighted word centroid distance is favourable in terms of ranking quality. While our evaluation and comparison is based on a `python` implementation, we simulated a practical setting, which can be realised in Elasticsearch.

Since we apply the actual embedding operation on top of the token frequencies, the integration of word embeddings in Elasticsearch is possible. However, the required cost for a query-document similarity computation would be increased by summation of the word vectors of the specific document. As this is an expensive operation, it is desired to compute the (IDF re-weighted) word centroids at index time instead of query time. Therefore, we propose to implement a dedicated `TokenFilter`, which takes raw token counts as input and produces a token stream whose size matches the dimensionality of the word embedding. This output token stream resembles the (possibly IDF re-weighted) word centroid vector for the respective document. Please note, that this would effectively nullify the matching operation, since word vectors are dense. Thus, the word centroid has to be stored separately, in addition to the raw token counts and should not affect the matching

operation. To obtain the desired behaviour, a dedicated `Similarity` is also necessary. Thus, the following additions to Elasticsearch are required:

`EmbeddingTokenFilter` A token filter that adds the word centroid for the document to a token stream input.

`EmbeddedSimilarity` A similarity that disregards the raw token counts but takes only the word centroids into account.

At index time, the documents are analysed and then the centroid is computed by the `EmbeddingTokenFilter`, while the original token counts are retained. In the analysis process at query time, we only analyse the query up to the raw token counts and do *not* apply the `EmbeddingTokenFilter`. Then the matching operation matches the documents in the index. After the matching operation, the `EmbeddedSimilarity` instance may transform the query into its word centroid representation and compute the cosine distance with respect to the word centroid of the document. A data flow graph is shown in Figure 8. To summarise, word embeddings can be integrated into Elasticsearch by a plug-in that provides a new dedicated token filter and similarity. It is necessary, to carefully store the centroids in the index without interfering with the matching operation. At query time, the similarity has to aggregate the word vectors to their centroids, before the cosine distance is computed. In the case of embedding-based query expansion, a dedicated `QueryParser` would be a possible future extension.

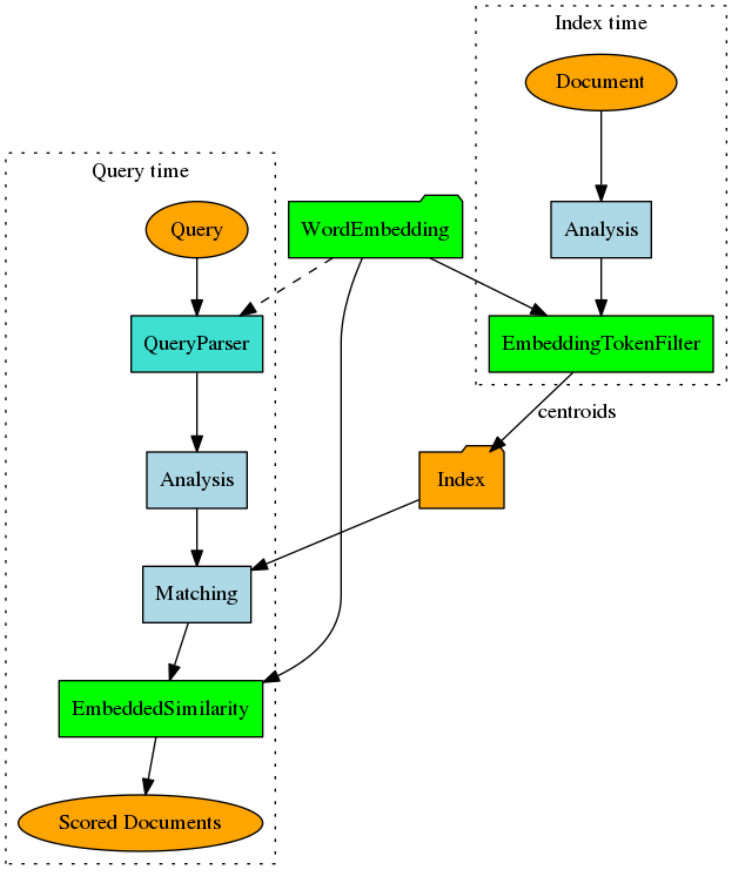


Figure 8: Data flow graph for the integration of embedding-based retrieval techniques into Elasticsearch. Ellipsoid shapes resemble volatile raw data, while rectangular shapes resemble algorithms. Folder-like shapes represent persistent data.

3.3 Text extraction from scholarly figures

3.3.1 Problem statement

Following Choudhury and Giles (2015) Scholarly figures are data visualisations in scientific papers such as bar charts, line charts and scatter plots. Many researchers use a semi-supervised text extraction approach (Chiang & Knoblock, 2015; Savva et al., 2011). However, semi-supervised approaches do not scale with the amount of scientific literature published today. Thus, unsupervised methods are needed to address the task of text extraction from scholarly figures. This task is challenging due to the heterogeneity in the appearances of the scholarly figures such as varying colours, font sizes and text orientations. Nevertheless, extracting text from scholarly figures provides additional information that is not contained in the body text (Sandra Carberry & Demir, 2006). To the best of our knowledge, we are the first who have compared different approaches for text extraction from scholarly figures (Böschen & Scherp, 2017).

Based on the related work (Weihua Huang & Leow, 2005; Chandrika Jayant, Krisnandi, Ladner, & Comden, 2007; Lu et al., 2009; Xu & Krauthammer, 2010; Sas & Zolnierek, 2013; Böschen & Scherp, 2015a, 2015b; Chiang & Knoblock, 2015), we have defined a generic pipeline of six sequential steps that abstracts from the various works on text extraction from scholarly figures. We have re-implemented and systematically evaluated the most relevant approaches for text extraction from scholarly figures as described in the literature (Böschen & Scherp, 2017). In total, 32 configurations of the generic pipeline have been investigated. Figure 9 shows the pipeline and the investigated methods for each step. We assess each pipeline configuration with regard to the accuracy of the text location detection via precision, recall and F1-measure. In addition, we evaluate the text recognition quality using Levenshtein distance based on the evaluation methodology of the Born-Digital Image Track of the ICDAR Robust Reading Competition.

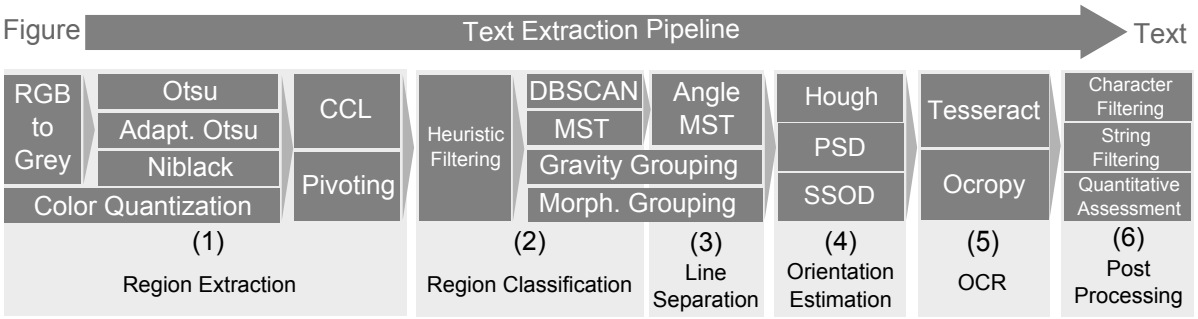


Figure 9: Generic pipeline for text extraction from scholarly figures.

3.3.2 Method description

Pipeline Structure

Based on the related work, we derived a generic pipeline for text extraction from scholarly figures as shown in Figure 9. The pipeline consists of six steps and can be implemented through different methods, which are described below. This allows to create different configurations of the pipeline and to conduct a fair comparison of these configurations.

The first step of the pipeline takes a scholarly figure (colour raster image) as input. The figure is converted into a binary image using either Colour Quantisation (Chiang & Knoblock, 2013), by reducing the number of colours in an image and taking each resulting colour channel as a separate binary image, or by converting the image to greyscale using the formula $Y = 0.2126R + 0.7152G + 0.0722B$ and subsequently applying a binarisation method. For binarisation, we use Otsu's method (Otsu, 1979), which finds the binarisation threshold by maximising the intra-class variance, Niblack's method (Khurshid, Siddiqi, Faure, & Vincent, 2009) which is often used for document image binarisation and Adaptive Otsu binarisation (Böschen & Scherp, 2015b), which hierarchically applies Otsu's Method to adapt to local inhomogenities. The output of the first step is a set of regions, where each region is a set of connected pixels. They can be extracted using classic Connected Component Labelling (CCL) (Samet & Tamminen, 1988), which iterates over the pixel of an image and connects adjacent foreground pixel into regions. Another option is the Pivoting Histogram Projection method (Xu & Krauthammer, 2010), which iteratively splits the binary image by analysing the horizontal and vertical projection profiles. The second step takes these regions as input and computes a feature vector for each region, consisting of coordinates of the centre of mass, dimension and area occupation, to classify

them into text or graphics. Heuristic Filtering (Sas & Zolnierek, 2013) can be applied, prior to more complex algorithms, to preprocess the set of regions and remove outliers. The classification of the remaining regions can be achieved using clustering methods like DBSCAN (Böschen & Scherp, 2015b), since text should be more dense in the feature space, or Minimum Spanning Tree (MST) clustering (Chandrika Jayant et al., 2007). Other approaches are Grouping Rules based on Newtons Gravity Formula (Weihua Huang & Leow, 2005) from classical physics or the Morphological Method (Chiang & Knoblock, 2015), which uses morphological operators to merge regions on pixel level. Subsequently, the generated sets of regions that are classified as text are fed into the third pipeline step to determine individual lines of text if necessary. For this step, we have only found one method in the literature, the Angle-Based MST clustering (Böschen & Scherp, 2015b). It computes a MST on the centres of mass of the regions and removes those edges that are not inside a predefined range of 60° around the main orientation. The fourth step of the pipeline computes the orientation for each text line using one of the following methods: The Hough Transformation (Böschen & Scherp, 2015b) can be used on the centres of mass of a text line's regions to transform them into Hough space, where the maximal value determines the orientation. A different option is to minimise the Perpendicular Squared Distance (PSD) of the bounding box of a text line to identify its orientation (Chandrika Jayant et al., 2007). The third option is the Single String Orientation Detection algorithm (SSOD) (Chiang & Knoblock, 2015) which determines the text line orientation using morphological operators. In the fifth step, existing OCR engines are used to recognise the horizontal text lines. We have evaluated the Tesseract OCR engine and Ocropy[16], since both are freely available, frequently updated and allow to reproduce our results without limitations. We used the English language models that are provided by the OCR engines and we deactivated any kind of layout analysis. The recognised text is post-processed in the sixth and last step of the pipeline. Here, we apply either Special Character Filtering that removes all special characters from the text, since they often appear when text was incorrectly recognised, Special Character Filtering per String (Sas & Zolnierek, 2013) that removes complete text lines, if they contain too many special characters, or Quantitative OCR Assessment (Chiang & Knoblock, 2015). The latter analyses the difference between the number of characters (regions) that went into the OCR process and the number of recognised characters in order to decide whether to discard a text line.

Pipeline configurations

From the previously defined methods, one can create various pipeline configurations. Some methods are restricted in how they can be combined as illustrated in Figure 9. Each of the seven configurations is identified by (`<configuration-name>`), an acronym created from the contributing author(s).

The first configuration (SZ13) is inspired by the work of Sas and Zolnierek (2013). It uses Otsu's method for binarisation, followed by CCL. Subsequently, it applies heuristic filtering similar to the original approach. The decision tree used by Sas and Zolnierek is replaced by the line generation approach based on MST. Since the original work by Sas and Zolnierek does not include a method for orientation estimation, we do not use any replacement in step 4. Tesseract is used as OCR engine, since it was also used in the original paper. In the post processing step, all strings are removed that contain too many special characters.

The second configuration (Hu05) is based on the work of Weihua Huang and Leow (2005). After region extraction using Otsu binarisation and CCL, the Heuristic Filter method is applied and the regions are grouped using the Gravity method. Finally, the grouped regions are processed with Tesseract.

Based on the work of Chandrika Jayant et al. (2007), the configuration (Ja07) starts with Otsu's method and CCL. Subsequently, it clusters the regions using a MST and approximates the orientation by minimising the perpendicular squared distance. Text recognition is achieved by applying Tesseract.

Different from the previous configurations, the fourth configuration (CK15) – inspired by Chiang and Knoblock (2015) – uses Colour Quantisation to generate multiple binary images, followed by a CCL. Subsequently, it applies heuristic filtering and Morphological Clustering on the regions. This step differs from the original paper, where the relevant colour levels were manually selected. Thus, we assess all extracted binary images. The orientation of each cluster is estimated using the SSOD method, followed by Tesseract OCR and quantitative post-processing.

Similar to the previous pipeline configuration, the fifth configuration (Fr15), inspired by Muhammad Fraz and Edirisinghe (2015), starts with Colour Quantisation and CCL. The original approach uses a supervised SVM to form words, which we replaced with unsupervised methods from our methods set. The extracted regions are filtered and DBSCAN is applied, followed by a MST clustering into text lines. The orientation of each text line is calculated using Hough method and the text is recognised using Tesseract.

All configurations so far use CCL to extract regions. The sixth configuration (XK10), motivated by Xu and Krauthammer (2010), uses the pivoting algorithm after binarisation with adaptive Otsu. The regions are filtered using heuristics and grouped into lines using DBSCAN and MST. This differs from the original work,

[16] `https://github.com/tmbdev/ocropy`, last accessed: 28/03/2017

which only applied heuristic filtering to remove the graphic regions. The reason behind this is that the authors only aimed at finding text regions and not to recognise the text. Thus, we filled the rest of the pipeline steps with suitable methods. The orientation of each line is estimated via Hough and OCR is conducted with Tesseract.

Finally, configuration (BS15) resembles our own work (Böschen & Scherp, 2015b). It uses adaptive Otsu for binarisation and CCL for region extraction. Heuristic Filtering is applied on the regions and DBSCAN groups them into text elements. Text lines are generated using the angle-based MST approach and the orientation of each line is estimated via Hough transformation, before applying Tesseract's OCR.

Influence of individual methods

In order to evaluate the influence of the individual methods, we chose the pipeline configuration (BS15) as basis for systematic modification, since our evaluation showed that it produces the best results. The systematic modifications are organised along the six steps of the generic pipeline in Figure 9. Each of the systematic configurations has an identifier (BS-XYZ) based on the original configuration, where X is a number that refers to the associated pipeline step and YZ uniquely identifies the method. The systematically modified configurations are described below.

Modifications of Step (1): The binarisation and region extraction is evaluated with the following configurations: (BS-1NC) differs from (BS15) by using Niblack instead of adaptive Otsu for binarisation. Configuration (BS-1OC) uses the third option for binarisation, Otsu's method. Colour Quantisation is combined with the pivoting region extraction in (BS-1QP).

Modification over Steps (2) and (3): The next step is the region classification and generation of text lines. Configuration (BS-2nF) differs from the base configuration by not applying the optional heuristic filtering method. Configuration (BS-2CG) uses the Gravity Grouping instead of DBSCAN and MST. Configuration (BS-2CM) applies MST to cluster regions and create text lines. Morphological text line generation is used in configuration (BS-23M).

Modifications of Step (4): The following two configurations assess the methods for estimating the orientation of a text line: Configuration (BS-4OP) uses the Perpendicular Squared Distance method and configuration (BS-4OS) uses the Single String Orientation Detection method to estimate the orientation.

Modifications of Step (5): For all configurations, both OCR engines are used to generate the results. The identifier of a configuration is extended to (BS-XYZ-T) or (BS-XYZ-O), when referencing the configurations that use Tesseract or Ocropy, respectively. Furthermore, we assess the direct impact of the OCR engine on the recognition results with configuration (BS15-O), which only differs with respect to the OCR method from the base configuration by using the Ocropy OCR engine instead of Tesseract.

Modifications of Step (6): The last step of the pipeline is the post-processing. We use three configurations to evaluate the different post-processing methods: Configuration (BS-6PC) uses the Special Character Filter method for post-processing. Configuration (BS-6PS) uses the String Filter method for post-processing. Configuration (BS-6PQ) uses the Quantitative Assessment method for post-processing.

3.3.3 Experimental evaluation and comparison

Datasets

We have used four datasets of varying origin and characteristics with in total 441 figures in our evaluation. We have created the EconBiz dataset, a corpus of 121 scholarly figures from the economics domain. We obtained these figures from a corpus of 288,000 open access publications from EconBiz[17] by extracting all images, filtering them by size and other constraints, and randomly selecting the subset of 121 figures. The dataset resembles a wide variety of scholarly figures from bar charts to maps. The figures were manually labelled to create the necessary gold standard information. We manually labelled the DeGruyter dataset as well, which comprises scholarly figures from books provided by DeGruyter[18] under a creative commons license[19]. We selected ten books, mostly from the chemistry domain, which contain figures with English text and selected 120 figures randomly from these books. The gold standard for these figures was created using the same tool which has been used for the creation of the EconBiz dataset. The Chart Image Dataset[20] consists of two subsets. The CHIME-R dataset comprises 115 real images that were collected on the Internet or scanned from paper. It has mostly bar charts and few pie charts and line charts. The gold standard was created by L. Yang,

[17] https://www.econbiz.de/, last accessed: 28/03/2017
[18] http://www.degruyter.com/, last accessed: 28/03/2017
[19] http://www.degruyter.com/dg/page/open-access-policy, last accessed: 28/03/2017
[20] https://www.comp.nus.edu.sg/~tancl/ChartImageDataset.htm, last accessed: 28/03/2017

Huang, and Tan (2006). The CHIME-S dataset consists of 85 synthetically generated images. This set mainly contains line charts and pie charts and few bar charts. The gold standard was created by Jiuzhou (2006).

We have also looked at ImageNet, TREC, ImageClef and ICDAR datasets. But none of them can be used to evaluate the specific challenges of scholarly figures. They either do not have the necessary ground truth information about the contained text or the dataset does not consist of scholarly figures. But we adopted the evaluation scheme of the Born-Digital Images track of the ICDAR Robust Reading Competition (RRC) (Karatzas et al., 2015), which is described below.

Procedure

We have selected three measures to evaluate the pipeline configurations and compare their results. Our gold standard consists of text elements which represent single lines of text taken from a scholarly figure. Each text line consists of one or multiple words which are separated by blank space. Each word may consist of any combination of characters and numbers. Every text line is defined by a specific position, size and orientation. Each pipeline configuration generates a set of text line elements as well. These text lines need to be matched to the gold standard. Since we do not have pixel information per character, we match the extraction results with the gold standard by using the bounding boxes. This is based on the first evaluation task of the ICDAR RRC and evaluates the text localisation on text line level. We iterate over all text lines in the gold standard and take all matches that are above the so-called intersection threshold. Our matching procedure calculates the intersection area between all pairs of the pipeline output and gold standard text lines. If the intersection comprises at least ten percent of the combined area of both text elements, than it is considered a match. This reduces the error introduced through elements which are an incorrect match and only have a small overlap with the gold standard. But it still allows to handle text lines that are broken into multiple parts. We look at each gold standard element and take all elements from the pipeline as matches that are above the intersection threshold. Thus, a gold standard element can have multiple matching elements and an element from the pipeline can be assigned to multiple elements from the gold standard if it fulfils the matching constraint for each match. We have defined three measures to assess these matches. The first two measures analyse the text localisation. The third measure compares the recognised text, similar to the word recognition task of the ICDAR RRC, although we compare text lines and not individual words.

First, we evaluate how accurate the configurations are at the Text Location Detection. If at least one match is found for an element from the gold standard set, it counts as a true positive, regardless of what text was recognised. If no match was found, it is considered as false negative. A false positive is an element from the pipeline output which has no match. From these values, we compute precision, recall and F1-measure. This measure is a binary evaluation and assesses only whether a match to an element exists or not. In addition, we report the Element Ratio (ER) which is the number of elements recognised by the pipeline divided by the number of elements in the gold standard and the Matched Element Ratio (MER) which is the number of matched items from the pipeline divided by the number of elements of the gold standard. These ratios give an idea whether gold standard elements get matched by multiple elements and whether the configuration tends to find more elements or less elements than it actually should find.

Second, we investigate the matching in more detail by assessing the Text Element Coverage. For each gold standard text element, we take the pixel of the bounding boxes and compute their overlap to calculate precision, recall and F1-measure over all of its matches. The true positives in this case are the overlapping pixel and the false positives are those pixel from the text elements from the pipeline which are not overlapping. The false negatives are the pixels of the gold standard element which were not covered by a text element from the pipeline. The values are averaged over all gold standard text elements in a figure.

Third, we assess the Text Recognition Quality by computing the Levenshtein distance between the extracted text and the gold standard. We calculate the distance for each match and report the average for the whole figure. Since multiple text elements from the pipeline can be matched to a gold standard text line, we have to combine their text into one string. We combine the elements using their position information. Besides a (local) Levenshtein distance per match, we also compute a global Levenshtein distance over all extracted text. This means that for each figure, we combine all characters from the text elements of the gold standard and add them to one string. Likewise, we create a string from the text elements extracted by the pipeline. The characters in both strings are sorted alphabetically and we compute the Levenshtein distance between these strings. This approximates the overall number of operations needed to match the strings without considering position information. Since the global Levenshtein distance depends on the number of characters inside a figure, we normalise it to an operations per character (OPC) score, which is computed by dividing the global Levenshtein distance by the number of characters in the gold standard. This makes the results comparable across scholarly figures with different amounts of characters.

Table 14: Average Precision (Pr), Recall (Re) and F1 values for Text Location Detection and Text Element Coverage, Element Ratio (ER) and Matched Element Ratio (MER) over all datasets for configurations from the literature.

	Text Location Detection					Text Element Coverage		
Config.	**Pr**	**Re**	**F1 (SD)**	**ER**	**MER**	**Pr**	**Re**	**F1 (SD)**
SZ13	0.63	0.47	0.54 (0.23)	0.80	0.59	0.52	**0.59**	0.47 (0.21)
Hu05	0.61	0.43	0.48 (0.28)	0.77	0.57	**0.79**	0.54	**0.57 (0.20)**
Ja07	0.59	0.45	0.49 (0.28)	0.83	0.51	0.41	0.32	0.32 (0.21)
BS15	0.66	**0.55**	**0.58 (0.25)**	**1.04**	0.69	0.60	0.49	0.50 (0.24)
CK15	0.52	0.50	0.53 (0.23)	1.37	0.60	0.53	0.41	0.42 (0.21)
Fr15	0.55	0.51	0.54 (0.25)	1.44	**0.72**	0.65	0.54	0.54 (0.23)
XK10	**0.73**	0.35	0.45 (0.26)	0.43	0.39	0.33	0.34	0.30 (0.22)

Table 15: Average local Levenshtein (LL), global Levenshtein (GL) and Operations Per Character (OPC) over all datasets for the configurations from the literature using Tesseract.

Config.	$\boldsymbol{AVG_{LL}(SD)}$	$\boldsymbol{AVG_{GL}(SD)}$	$\boldsymbol{OPC}$
SZ13	6.67 (4.82)	122.28 (141.03)	0.70
Hu05	6.65 (5.41)	126.35 (138.95)	0.71
Ja07	7.92 (5.56)	150.25 (140.59)	1.13
BS15	6.23 (4.93)	**108.81 (108.53)**	**0.67**
CK15	**6.07 (5.08)**	120.12 (125.87)	0.71
Fr15	6.72 (6.02)	135.64 (201.31)	0.85
XK10	7.06 (5.41)	125.45 (134.88)	0.74

Results

We have executed all configurations over the datasets described above. For reasons of simplicity, we are only reporting the average values for Text Location Detection, Text Element Coverage and Text Recognition Quality over all datasets. The detailed results per dataset can be found in our Technical Report (Böschen & Scherp, 2016). We compute the average Precision/Recall/F1-measure over the elements of each figure. We report the average Precision/Recall/F1-measure in terms of mean and standard deviation over all figures. The local Levenshtein distance is reported as the average of the mean values per figure and the average standard deviation. The global Levenshtein distance is defined by the mean and standard deviation over all figures and the average of the normalised OPC score.

First, we report the results of the configurations from the literature. Subsequently, we present the results for the systematically modified configurations. The Text Location Detection and Text Element Coverage results for the configurations from the literature computed over all datasets are reported in Table 14. The best result, based on the F1-measure, is achieved by configuration (BS15) with a F1-measure of 0.58. The coverage assessment in Table 14 shows the best precision of 0.79 for (Hu05), the best recall of 0.59 for (SZ13) and the best F1-measure of 0.57 for (Hu05). The text recognition quality is presented in Table 15. We obtain the best results with (BS15) with 0.67 operations per character (OPC), an average global Levenshtein of 108.81 and an average local Levenshtein of 6.23. The best local Levenshtein of 6.07 is achieved by configuration (CK15). For the systematically modified configurations, Table 16 shows the Text Location Detection results and the Text Element Coverage. Table 17 shows the Text Recognition Quality. The best location detection F1-measure of 0.67 is achieved by (BS-4OS), which is also supported by the coverage assessment with the highest F1-measure of 0.65. Configuration (BS-4OS-O) also produces the best text recognition results with an average local Levenshtein of 4.71 and an OPC of 0.53. In addition, configuration (BS-4OS-O) shows the best results of 95.49 for the average global Levenshtein distance. Comparing the different, systematically modified configurations per step of the pipeline shows that the only major improvement is achieved by (BS-4OS). Please note, a performance analysis of the different configurations can be found in our Technical Report (Böschen & Scherp, 2016).

Table 16: Systematically modified configurations: Average Precision (Pr), Recall (Re) and F1 values for Text Location Detection and Text Element Coverage, Element Ratio (ER) and Matched Element Ratio (MER) over all datasets.

	Text Location Detection					Text Element Coverage		
Config.	**Pr**	**Re**	**F1 (SD)**	**ER**	**MER**	**Pr**	**Re**	**F1 (SD)**
BS15	0.66	0.55	0.58 (0.25)	1.04	0.69	0.60	0.49	0.50 (0.24)
BS-1NC	0.64	0.52	0.57 (0.25)	0.96	0.64	0.59	0.44	0.47 (0.24)
BS-1OC	0.67	0.40	0.49 (0.26)	0.74	0.53	0.46	0.40	0.38 (0.26)
BS-1QP	0.61	0.44	0.48 (0.25)	0.96	0.75	0.41	0.57	0.42 (0.23)
BS-2nF	0.60	0.46	0.51 (0.23)	0.86	0.52	0.59	0.54	0.50 (0.21)
BS-2CG	0.62	0.50	0.55 (0.27)	0.90	0.64	0.76	0.54	0.57 (0.20)
BS-2CM	0.61	0.54	0.59 (0.25)	1.19	0.74	0.57	0.47	0.47 (0.24)
BS-23M	0.67	0.55	0.62 (0.23)	1.08	0.65	0.60	0.47	0.48 (0.22)
BS-4OP	0.62	0.53	0.57 (0.24)	**1.01**	0.66	0.49	0.40	0.41 (0.20)
BS-4OS	0.67	**0.63**	**0.67 (0.22)**	1.27	**0.88**	**0.77**	**0.63**	**0.65 (0.17)**
BS-6PC	**0.69**	0.54	0.59 (0.25)	0.97	0.70	0.59	0.49	0.49 (0.24)
BS-6PS	0.67	0.55	0.60 (0.25)	**1.01**	0.69	0.59	0.49	0.49 (0.24)
BS-6PQ	0.66	0.38	0.48 (0.25)	0.60	0.43	0.39	0.29	0.31 (0.21)

Table 17: Average local Levenshtein (LL), global Levenshtein (GL) and Operations Per Character (OPC) over all datasets for the systematic configurations.

	Tesseract			Ocropy		
Config.	$\boldsymbol{AVG_{LL}(SD)}$	$\boldsymbol{AVG_{GL}(SD)}$	$\boldsymbol{OPC}$	$\boldsymbol{AVG_{LL}(SD)}$	$\boldsymbol{AVG_{GL}(SD)}$	$\boldsymbol{OPC}$
BS15	6.23 (4.93)	108.81 (108.53)	0.67	5.47 (4.98)	108.55 (106.64)	0.64
BS-1NC	6.27 (4.95)	117.58 (124.23)	0.69	5.70 (5.09)	117.46 (128.73)	0.66
BS-1OC	6.55 (5.06)	131.58 (142.74)	0.75	6.16 (5.21)	131.39 (143.16)	0.73
BS-1QP	8.31 (6.14)	154.54 (168.10)	1.09	7.06 (5.62)	136.40 (132.05)	0.82
BS-2nF	6.55 (4.94)	111.30 (105.13)	0.75	6.29 (5.50)	120.71 (109.18)	0.76
BS-2CG	6.68 (5.65)	108.86 (102.93)	0.66	6.22 (5.75)	130.21 (127.87)	0.69
BS-2CM	6.30 (5.29)	115.43 (113.79)	0.69	5.85 (5.34)	110.74 (107.23)	0.67
BS-23M	6.15 (5.12)	104.61 (105.97)	0.63	5.52 (5.10)	106.71 (104.05)	0.64
BS-4OP	8.30 (5.59)	147.91 (129.55)	1.04	7.23 (5.60)	135.21 (122.48)	0.85
BS-4OS	**5.47 (4.39)**	**96.29 (99.44)**	**0.58**	**4.71 (4.66)**	**95.49 (94.80)**	**0.53**
BS-6PC	5.96 (4.88)	105.50 (107.16)	0.61	5.46 (5.00)	109.07 (104.57)	0.63
BS-6PS	6.20 (4.90)	108.06 (109.38)	0.64	5.45 (4.96)	106.38 (103.29)	0.63
BS-6PQ	6.07 (5.03)	120.78 (122.44)	0.67	5.79 (4.97)	126.92 (124.06)	0.71

Discussion

Comparing the different configurations from the literature shows that the best performing configuration is (BS15). A possible reason is that our pipeline does not make many assumptions about the figures, e. g. figure type, font, or colour. Thus performing better on the heterogeneous datasets. In the following, we will discuss the results for the individual pipeline steps based on the results from the systematically modified configurations. Comparing the configurations for the first pipeline step leads to the conclusion that the adaptive binarisation works best, because it can adapt to local variations of the appearance in a figure. Otsu's method is too simple and Niblack's method is more suited for document images which have fewer colour variations. The lower results for the pivoting algorithm can be explained with the larger regions and the possibility that a region can be a mixture of text and graphic elements due to the only horizontal and vertical subdivision. Looking at step 2 and 3 of the pipeline, only the morphological clustering shows slightly better results than the DBSCAN-MST combination, most likely due to its processing on pixel level. The overall best results, when also considering the systematic configurations, are achieved by (BS-4OS). This can be explained by the fact that the orientation estimation via Hough works on the centres of mass of character regions, which is an aggregated region representation, while the SSOD in (BS-4OS) computes the orientation on the original pixels. Thus, it avoids a possible error, which could be induced by the pixel aggregation. When comparing the OCR engines from step 5, Ocropy generally produces better results than Tesseract. Ocropy seems to be more conservative, having built in much more restrictions about what input to accept and when to execute the

OCR. Furthermore, each OCR engine comes with its own English language model and we did not evaluate their influence. The methods for post-processing do not improve the results. One reason might be the simplicity of methods. Thus, some more advanced techniques may be developed in the future. Overall, there are many more options for the different pipeline steps, e. g. other binarisation methods, different clustering algorithms or post-processing methods that could be used. However, we made a selection of relevant approaches and methods to limit the combinatorial complexity.

3.3.4 Implementation, APIs and integration

We provide the datasets and the implementation of the generic pipeline that was used in our experiment to the public[21]. This allows for integrating and comparing new methods as well as the reproduction of our results.

3.4 Metadata extraction from PDF

3.4.1 Problem statement

It might be desirable to enrich existing metadata entries with additional information that is not given in the metadata fields of the source collection. Some of this information is available in full-text PDF files linked with the metadata entries. In the extreme case, a metadata entry is completely empty and thus all meta information has to be extracted from the PDF. Usually, only certain information is relevant in a specific application context, e.g. for authorship disambiguation one needs affiliations and emails of each author in the document. In general, we can distinguish the following main extraction purposes within the task of metadata extraction from PDF:

1. Full-text extraction: This type of extraction is mostly applied for indexing purposes. All words that can be found in the PDF are extracted and counted. The quality of extraction is not particularly important as incorrect words are usually over-specific and do not influence retrieval. In a more advanced setup, we might be interested in extracting sentences or paragraphs from the full-text PDF to allow application of Natural Language Processing (NLP) methods like entity recognition in a later step. Here, high precision is essential. For full-text extraction, it can be sufficient to deploy a standard PDF-to-txt extractor and apply tokenisation.
2. Header extraction: The task of extracting specific (standard) types of information common in many metadata schemas is often referred to as header extraction (Lipinski, Yao, Breitinger, Beel, & Gipp, 2013). The challenging issue here is to map certain parts of the PDF to specific metadata fields. This task is often considered central in many PDF-to-metadata extractors.
3. Reference extraction: A special case of metadata extraction from PDF is the extraction of reference items in the list of documents cited in (scientific) documents. This problem can be separated into the subproblems of extracting single reference items from the PDF and the task of parsing the item into several fields. In our case, if we decide to include reference information into our database, we would also have to match existing document metadata entries with these reference items in order to include a link between these two documents.
4. Author-specific extraction of information: For author disambiguation and a comprehensive presentation of real-world authors, it can be helpful to extract author-specific information from PDFs. Usually, when authors are mentioned on (scientific) documents, under each name, one can find an email address and an affiliation. This task is relatively specific, but highly relevant for our purposes.

3.4.2 Method description

We will use an existing tool or a pipeline of existing tools for metadata extraction from PDFs. As described in the previous section, full-text extraction is the most basic method and can be extended by tokenisation methods and some quality checks. Converting PDFs to text files is a preprocessing step to a more complex task like header extraction. PDF is a proprietary format owned by Adobe Systems Inc. Adobe also provides a solution called Acrobat Document Cloud [22] for this task, but it is not free software. Besides being open software, a requirement for our purposes is that the tools can be run in batch mode from the command line without any interaction required. A popular tool that fulfils both requirements is PDFtoText from the free Poppler [23] library. Lipinski et al. (2013) compared different free batch-mode tools for header extraction and

[21] http://www.kd.informatik.uni-kiel.de/en/research/software/text-extraction, last accessed: 28/03/2017
[22] https://acrobat.adobe.com, last accessed: 23/03/2017
[23] https://poppler.freedesktop.org/, last accessed: 23/03/2017

found that the best performing tool is Grobid (Lopez, 2009). As an alternative to tools that consider only the current PDF document in order to extract metadata, there are also some tools that use the extracted text to look up higher quality entries in existing literature databases. Zotero (Ahmed & Al Dhubaib, 2011) is the most prominent solution applying this approach. Our comparisons turned out that Grobid works at least as good as Zotero. Therefore, we prefer to use Grobid. Zotero can only outperform Grobid if it does a Google Scholar lookup. We currently intend to use only Grobid with the exception of non-English documents (Grobid only works for English). It is yet to be determined what is a good solution for other languages, in particular German.

For reference extraction, we make use of future achievements of the DFG-funded project EXCITE[24] at GESIS. We know that CERMINE (Tkaczyk, Szostek, Fedoryszak, Dendek, & Bolikowski, 2015) and Grobid offer relatively reliable reference extraction for English, but EXCITE will also focus on German documents.

For extraction of author-specific information (i.e. email or affiliation), we can use certain fields of header extraction methods. However, the correct extraction of affiliations does not necessarily mean that the information is linked correctly to the right author. An additional challenge therefore is the correct linking between the extracted information and the respective author mention on the document.

3.4.3 Experimental evaluation and comparison

In order to narrow down the number of tools that could be integrated into an enrichment workflow, we consider the results presented in the comparison paper by Lipinski et al. (2013). In most cases, results were quite satisfying when trying out the tools on a small number of documents. For comparison of Grobid and CERMINE, we performed a slightly more formal evaluation and compared the results for 27 documents from different research areas in the Social Science Open Access Repository (SSOAR[25]) (see D6.2: Data Management Plan) in the header fields title, authors, affiliations per author, affiliations as a set and the publication year. We compared each field to the gold standard metadata present in the SSOAR dataset. For each extraction it was recorded whether the correct value (good), no result (not so good) or the wrong value (bad) was given. The comparison showed that Grobid performs best.

As we are currently not focusing on PDF extraction, we have not yet tested a complete extraction framework, but we plan to do so using the following setup: We will compose a set of PDFs of different types (i.e. traditional layout, many images, etc.) for which we have metadata available. Furthermore, we will define a similarity measure of metadata field values (for each field separately). Then, we can directly compare results returned by the extraction framework and the gold standard. By the similarity measure we can identify differences between the gold standard and the extracted values (i.e. an author is extracted correctly but his name on the PDF is written differently). This evaluation can easily be extended to include other state-of-the-art methods and can give researchers or platform developers valuable insights into which tool to use.

3.4.4 Implementation, APIs and integration

Different methods for extraction of additional information from PDF can be applied as enrichment services in the MOVING platform as depicted in Figure 17 (see Section 3.7). For example, one service can be used to enrich the metadata of a document by feeding a PDF file to a reference extractor and linking the output to another document already in the database. In general, all PDF extraction services will take as input a PDF file and return a complex object, i.e.:

- Full-text extraction: set of $(term, frequency)$ pairs.
- Header extraction: set of $(metadata\,field, value)$ pairs.
- Reference extraction: set of reference strings or set of $(referenceID, reference\,field, value)$ triples for parsed references.
- Author-specific extraction: set of $(mentionID, author\,field, value)$ triples.

These outputs can be fed through other services (i.e. for normalisation) until the final values are entered into the metadata entries. A parent control process can run on the database and call the specific services for each document.

[24] http://www.gesis.org/forschung/drittmittelprojekte/projektuebersicht-drittmittel/excite/, last accessed: 23/03/2017

[25] www.ssoar.info, last accessed: 28/03/2017

3.5 Adaptive index models for Linked Data retrieval

Finding relevant sources of data on the web for a given information need is crucial to the success of the Linked Open Data (LOD) idea. However, this task is not trivial since there is a vast amount of data available, which is distributed over various sources. Therefore, the success of finding data often depends on good recommendations, e.g. through social media or search on known data portals. However, recommendations are not generally accessible and search on data portals is often done over the metadata description. Furthermore, not necessarily all data sources are listed in those portals. In order to address this challenge, schema-level indices have been developed. A schema-level index abstracts from the actual data by analysing and aggregating the instances along common uses of, e. g. RDF types and properties. Such aggregations of instances are called schema elements. An example is illustrated in Figure 10. Thus, search systems like LODatio (Gottron,

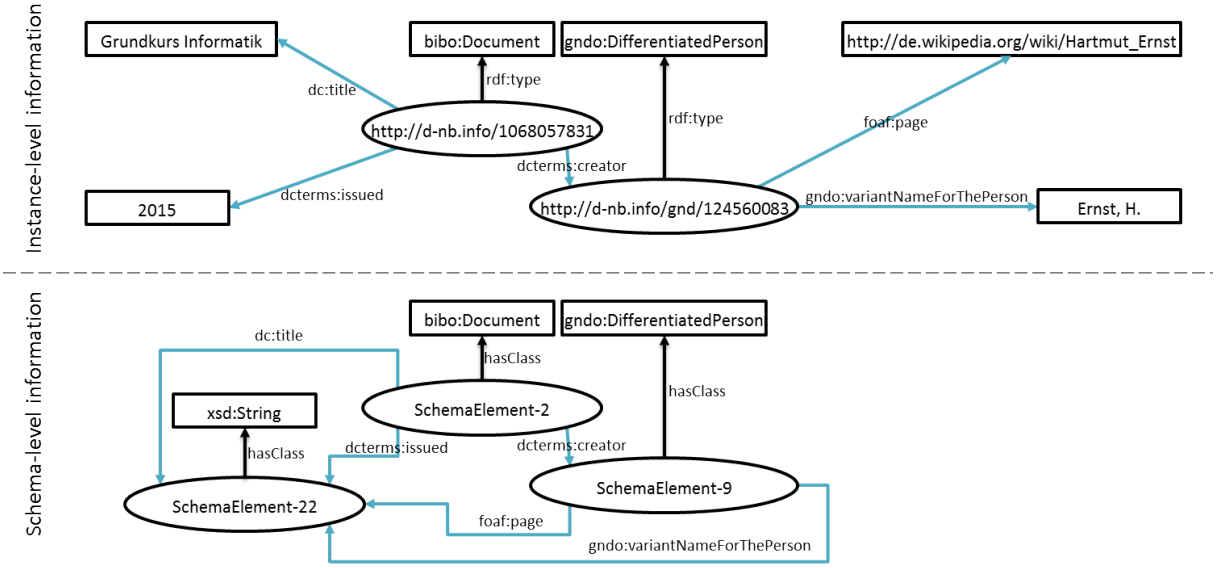


Figure 10: An example RDF graph with instance-level information and an example of a schema extracted from the instance data.

Scherp, Krayer, & Peters, 2013) that are using a schema-level index support users in finding LOD sources based on queries of combinations of different RDF types and/or properties. For example, one would search for data sources containing instances of type *bibo:Document*, connected over the property *dc:creator* to another instance of type *gndo:DifferentiatedPerson*. Such a schema pattern defines the relevancy of a data source, in this example bibliographic metadata which is linked to an unique representation of an author (modelled as "differentiated person"). However, there are various different possibilities and variants of how to model bibliographic metadata since Linked Open Data gets published in a decentralised fashion with no central authority. This of course does not apply only to bibliographic metadata, but rather to the whole LOD cloud, which includes structured data from various other domains, e.g. data about governments, organisation, products, cities or social media activities.

In the past, different definitions of schema-level indices have been developed which allow for capturing different schema patterns (Goldman & Widom, 1997; McHugh, Abiteboul, Goldman, Quass, & Widom, 1997; Neumann & Moerkotte, 2011; Ciglan, Nørvåg, & Hluchý, 2012; Konrath, Gottron, Staab, & Scherp, 2012; Benedetti, Bergamaschi, & Po, 2015; Spahiu, Porrini, Palmonari, Rula, & Maurino, 2016; Schaible, Gottron, & Scherp, 2016). Thus, they can be used to answer different information needs. Furthermore, the different indices entail different computational complexity and storage requirements, since they aggregate data on different levels of granularity. For example, when searching for certain bibliographic metadata representations it is not enough to query for directly attached properties and types. To define a good relevancy, sometimes multiple related instances have to taken into account and a schema-level index needs to reflect those complex structure in order to efficiently answer complex queries. Considering Figure 10, the presented instance-level information can be aggregated into three different schema elements according to the SchemEX index definition (Konrath et al., 2012). This means the query results for complex queries (taking RDF types and properties of multiple instances into account) are computed over multiple schema elements which leads to computational complexity and possibly to erroneously aggregated query results. Another important issue is that many different vocabularies

can be used to model the same attributes. Therefore inferring more information could ease the search. For example, if two libraries model the same book using different combination of vocabulary terms but include an *owl:sameAs* link, for both instances both schema descriptions can be used.

3.5.1 Problem statement

In order to develop, compare, and validate different variants of schema-level indices, a formal model for the parameterised specification of such indices is needed. We present a first parameterised model with five different parameters for a schema-level index. The formal model distinguishes between schema information and payload information. The schema information comprises schema elements, which are aggregations of RDF instances, and is used for answering queries against the data. The payload information comprises information about the data which is of interest for the use case. For example, for a search engine as described above the payload could contain data source URIs to memorise where on the LOD cloud instances with a certain combination of RDF types and properties can be found. Our formal model comprises instance and property related parameters as well as an extended triple space parameter and a parameter for the overall size of the considered subgraph per instance. These parameter allow among others then to summarise sets of instances connected over *owl:sameAs* (Ding, Shinavier, Shangguan, & McGuinness, 2010), define sets of ignored properties in the RDF graph (T. Tran, Ladwig, & Rudolph, 2013), and enable inferencing over RDF Schema such as *rdfs:domain*, *rdfs:range*, *rdfs:subClassOf* and *rdfs:subPropertyOf*. To evaluate the formal model we use it to implement different schema-level indices on top of our existing stream-based schema extraction tool SchemEX (Konrath et al., 2012). An important step of schema-level index computation is to extract and aggregate the information as defined by the schema elements from the data graph. Such a schema computation has to be highly efficient in order to be capable of indexing the whole LOD cloud. Stream-based approaches are known to sufficiently scale for schema computations over very large graphs (Konrath et al., 2012), although due too limited window sizes introduce an approximation error. We use our implementation to run evaluations on six different schema-level index configurations with respect to their individual approximation error in terms of precision and recall of the attached data sources.

3.5.2 Method description

Schema-level indices

A data graph DG is defined by a set of triples $DG \subset V_{RB} \times P \times (V_{RB} \cup L)$, where $V_{RB} = R \cup B$ denotes the set of resource R and blank nodes B, P the set of RDF properties, and L the set of literals. These triples are considered as statements about resources (represented by an URI) in the form of subject–predicate–object expressions (s, p, o). One common practice in the context of Linked Open Data (LOD) is to aggregate triples to entities via a common subject URI. Such an entity is represented by the subject URI, the so called instance URI. Thus, we consider instances $i \in V_{RB}$ to be such resources, that appear at least once as subject within a triple. The subset $V_C \subset V_{RB}$ contains all RDF classes. A resource $c \in V_C$ is considered a RDF class, if there exists a triple $(i, \text{rdf:type}, c) \in DG$. Using these basic notions, we can define a generic schema-level index. Any schema-level index partitions instances into disjoint subsets. Since all subsets are disjoint, they can be treated as equivalence classes and can therefore be defined using equivalance relations (European Mathematical Society, 2014). Each equivalence class is then represented by exactly one schema element and the set of all schema elements makes up the schema information. The schema information is the index search space and is fundamental to any schema-level index as it used to answer the queries. Furthermore, the index needs to be connected to the actual data. To this end, we reuse the notion of payload (Gottron et al., 2013). The payload comprises information about the actual data, e.g. all instances or only references to their datasource. Using these basic notions, we can define a simple generic schema-level index as 3-tuple of the data graph DG, a graph partitioning function EQR and a payload function PAY.

Definition (Schema-level index)**.** *A schema-level index can be described as 3-tuple* (DG, EQR, PAY)*, where* DG *is the data graph which is indexed,* EQR *is a data graph partitioning function, which assigns instances to exactly one schema element and* PAY *is a set of payload functions which define the extracted instance information assigned to a schema-element.*

Below, we introduce some subsets of properties P appearing in the data graph DG. We allow to exclude certain properties as suggested by T. Tran et al. (2013).

Definition (Property set P)**.** *The properties* P *can be divided into disjoint subsets* $P = P_{type} \dot\cup P_{rel} \dot\cup P_{ign}$*, where* P_{type} *contains all properties denoting type information and* P_{rel} *contains all properties denoting relationships between instances. The set* P_{ign} *can contain user defined properties which are explicitly ignored during index computation.*

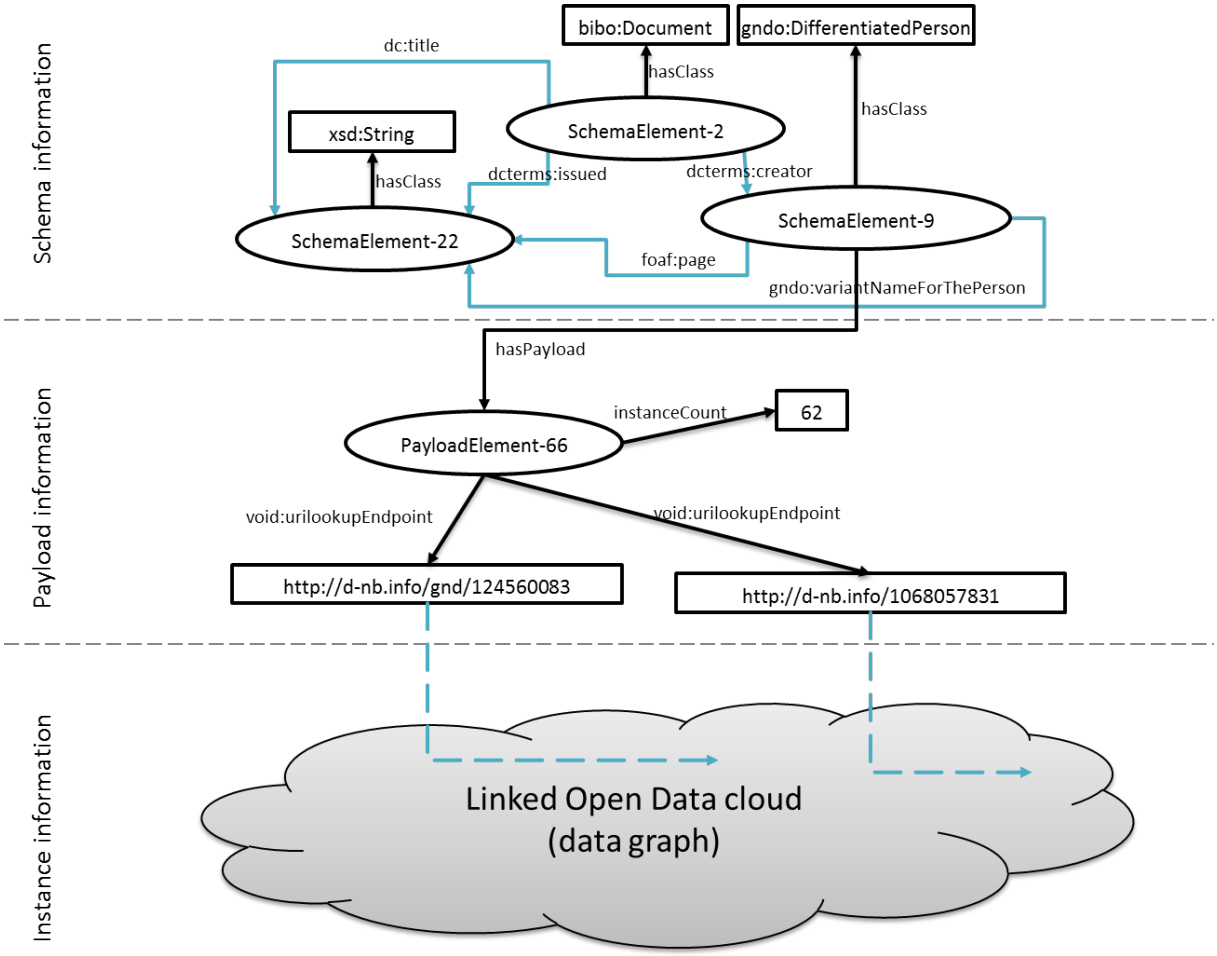


Figure 11: Simplified example of an index for a data graph distinguishing schema information and payload information. The payload information is stored locally and links to the original data source on the Linked Open Data (LOD) cloud.

Label parameterisation

All proposed schema elements in this work are defined using equivalence relations. Equivalence relations partition sets into disjoint subsets and have a projection that assigns items to the corresponding subset, the so called equivalence class (European Mathematical Society, 2014). To simplify the notation, we call equivalence classes schema elements. The first schema element is called Type Cluster (TC). A Type Cluster partitions the data graph by aggregating instances based on a common set of assigned types.

Definition (Type Cluster (TC)). *A type set $TS \in \mathscr{P}(V_C)$ is a set of RDF classes. The type set of an instance $i \in V_{RB}$ is defined by the type set $\Gamma(i) := \{c \mid (i_1, p_1, c_1) \in DG \ \wedge p_1 \in P_{type} \ \wedge \ c_1 \in V_C\}$. The Type Cluster aggregates all instances which have the same type set.*

Note, that the definition above qualifies as equivalence relation since it is a reflexive, symmetric and transitive relation. The Type Cluster depends on the property set P_{type}. With $P_{type} = \{\text{rdf:type}\}$ the Type Cluster includes all RDF classes of an instance. With $P_{type} = \emptyset$ there is no distinction between types and properties.

The next schema element is the Equivalence Class (EQC). It uses the Type Cluster definition and additionally considers the properties P_{rel}. A Equivalence Class contains all instances with an equivalent schema according to the equivalence relation. This means that all properties in $P_{ign} \subset P$ are ignored when computing the equivalence relation.

Definition (Equivalence Class (EQC)). *An Equivalence Class $EQC \in \mathscr{P}(V_{RB})$ is defined by the equivalence relation $\sim$ over instances for any $i_1, i_2 \in V_{RB}$:*

$$(i_1 \sim i_2) \Leftrightarrow \forall (i_1, p_1, o_1) \in DG : p_1 \in P_{rel} \ \wedge \ \exists (i_2, p_2, o_2) \in DG : \ p_2 \in P_{rel} \ \wedge \ p_1 = p_2 \ \wedge \\ \Gamma(i_1) = \Gamma(i_2) \ \wedge \ \Gamma(o_1) = \Gamma(o_2)$$

The equivalence relation partitions all instances based on a common set of types (Type Cluster) that have the same properties in P_{rel} which link to resources with the same Type Cluster. For example the SchemEX index uses $P_{type} = \{\text{rdf:type}\}$ and $P_{ign} = \emptyset$ (Konrath et al., 2012). With $P_{type} = \emptyset$ and $P_{ign} = \{\text{rdf:type}\}$, we can achieve an index structure like SemSets (Ciglan et al., 2012). Equivalence Classes partition instances regarding their outgoing properties P^+. So far incoming properties P^- have not been considered, but this extension can easily be integrated.

Instance sets

So far, we considered Type Cluster and Equivalence Classes only over instances. In the following, we propose to aggregate multiple instances that resemble the same real-world entity using *owl:sameAs* before indexing them. The commonly used owl:sameAs property links two instances, and states according to its W3C definition their equality [26]. This motivates to model the inclusion of properties with special semantics like owl:sameAs. We propose to include RDF types and properties from all connected resources instead of indexing the owl:sameAs property. To this end, we introduce the function $\sigma : V_{RB} \rightarrow \mathscr{P}(V_{RB})$, which for a given instance i returns a set of connected instances, called instance set. The super set of all instance sets is called $\mathscr{I}$.

Definition (SameAs Instance Set)**.** *For a given data graph DG, let $\sigma : V_{RB} \rightarrow \mathscr{P}(V_{RB})$ be a function, which for a given resource i_1 returns all resources $i_2 \in V_{RB}$, where there is a path in DG from i_1 to i_2 (independent of the edges direction), over all edges labelled owl:sameAs.*

It can easily be shown that the assignment of an instance to a SameAs Instance Set is unique, by reducing the problem to finding weakly connected components in a owl:sameAs-labelled subgraph of DG as is has been done by Ding et al. (2010). With the notion of σ, we can extend Definition 3.5.2 to consider instance sets instead of single instances.

Definition (Extended Type Cluster (eTC) over SameAs Instance Sets)**.** *The Extended Type Cluster of an SameAs Instance Set $\sigma(i_1)$ is defined by the type set over an instance set $\Gamma'(\sigma(i)) := \bigcup_{j \in \sigma(i)} \Gamma(j)$.*

Including the Extended Type Cluster Definition 3.5.2, we can also extend the Equivalence Class Definition 3.5.2 and apply it for instance sets.

Definition (Extended Equivalence Class (eEQC) with relation sets over instance sets)**.** *An Extended Equivalence Class $eEQC \in \mathscr{P}(V_{RB})$ is defined by the equivalence relation $\sim'$ over instance sets $\sigma(i_1)$ and $\sigma(i_2)$ for any $i_1, i_2 \in V_{RB}$:*

$$\begin{aligned}\sigma(i_1) \sim' \sigma(i_2) \Leftrightarrow \forall (i_v, p_v, o_v) \in DG : p_v \in P_{rel} \ \wedge \ \exists (i_w, p_w, o_w) \in DG : \ i_v \in \sigma(i_1) \ \wedge \\ i_w \in \sigma(i_2) \ \wedge \ p_w \in P_{rel} \ \wedge \ p_v = p_k \ \wedge \\ \Gamma'(\sigma(i_v)) = \Gamma'(\sigma(i_w)) \ \wedge \ \Gamma'(\sigma(o_v)) = \Gamma'(\sigma(o_w))\end{aligned}$$

Please note, that in the definition above the Extended Type Cluster $\Gamma'(\sigma(i_v))$ is the same as the Extended Type Cluster $\Gamma'(\sigma(i_1))$, since i_1 and i_v are part of the same SameAs Instance Set. An example graph is

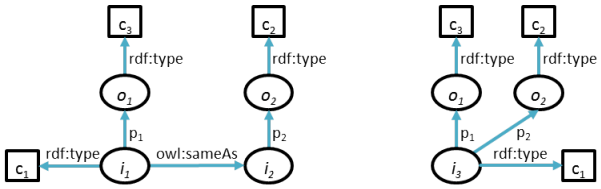


Figure 12: Sample data graph which can be aggregated to either three Equivalence Classes or one Extended Equivalence Class using SameAs instance sets

illustrated in Figure 12. According to the plain Equivalence Class definition i_1, i_2 and i_3 are not equivalent. However, aggregating i_1 and i_2 to a SameAs Instance Set leads the equivalence of all three instances according to the Extended Equivalence Class using σ.

[26] `https://www.w3.org/TR/owl-semantics/`, last accessed: 08/03/2017

RDF schema graph
The data graph DG introduced above contains triples from the LOD cloud such as assertion about individuals as well as assertions about RDF types and properties (De Giacomo & Lenzerini, 1996). For example, the data graph can contain the following three triples:

<Hans> <foaf:knows> <Frank> .
<foaf:knows> <rdfs:domain> <foaf:Person> .
<foaf:knows> <rdfs:range> <foaf:Person> .

For a schema-level index focussing on indexing schema patterns of individuals, e.g. bibliographic metadata, it is of no added value to index the assertions about the used vocabulary terms, such as *<foaf:knows> <rdfs:domain> <foaf:Person>*. Instead of indexing the assertions about the property *foaf:knows*, such a schema-level index would rather index the inferred statements about *Hans* and *Frank* derived form the statements about *foaf:knows*.

<Hans> <rdf:type> <foaf:Person> .
<Frank> <rdf:type> <foaf:Person> .

The schema summarisation tool ABSTAT (Spahiu et al., 2016) incorporates this by inferring triples based on a subtype schema graph, which is constructed a-priori by extracting the contained schema assertions. We extend ABSTAT by including RDFS subproperties in the schema graph. RDFS is commonly used in the context of Linked Data. We formally distinguish the two phases of (I) schema graph construction and (II) schema graph inferencing. Whether all triples are materialised at schema-computation time or on query time does not make a difference for our formal model.

(I) Schema graph construction: The schema graph is a directed, edge-labelled multigraph, which depicts hierarchical dependencies of rdfs:subClassOf and rdfs:subPropertyOf in a tree structure with further cross connections regarding rdfs:range and rdfs:domain. Properties and RDF classes are a parent node of a node in the schema graph if they are in subject position in a triple with the aforementioned properties and the corresponding object. The property and RDF class names are the node labels in the schema graph.

Definition (Schema graph)**.** *Let $SG := (V_C \cup P, \mathbb{E})$ be an edge labelled directed multigraph and $\mathbb{E} \subseteq (V_C \cup P \times V_C \cup P)$. The set of nodes is the union of the set of RDF classes and the properties. The edge-label function $\phi : \mathbb{E} \to P$ assigns labels from a given set of possible properties P to all edges $e \in \mathbb{E}$.*

An example schema graph modelling RDFS vocabulary terms is depicted in Figure 13. Please note that multigraphs allow parallel edges between nodes, which allows modelling multiple relationships between nodes. Such a schema graph enables search for related types and properties. We construct the RDFS schema graph by extracting triples containing RDFS vocabulary terms, namely all properties

$$P_{RDFS} = \{\text{rdfs:subClassOf},\ \text{rdfs:subPropertyOf},\ \text{rdfs:range},\ \text{rdfs:domain}\}$$

and label the the schema graph respectively:

$$\phi_{RDFS}((u,v) \in \mathbb{E}) = \begin{cases} \text{rdfs:subClassOf} \mid \exists(u, \text{rdfs:subClassOf}, v) \in DG \\ \text{rdfs:subPropertyOf} \mid \exists(u, \text{rdfs:subPropertyOf}, v) \in DG \\ \text{rdfs:range} \mid \exists(u, \text{rdfs:range}, v) \in DG \\ \text{rdfs:domain} \mid \exists(u, \text{rdfs:domain}, v) \in DG \end{cases}$$

Cycles in the schema graph, e.g. two RDF classes that are a sub class of each other, do not restrict the forming of Equivalence Classes in this model. However it would be possible to regard this special case separately. In the following, we denote the schema graph constructed using ϕ_{RDFS} with SG_{RDFS}.

(II) Schema graph inferencing: Having the hierarchically dependencies of types and properties represented using a schema graph, additional triples can be inferred. Based on the schema graph we introduce the next parameterisation called extended triple set parameterisation Φ.

Definition (Extended triple set $\Phi(DG, SG)$)**.** *The extended triple set parameterisation takes any data graph DG and based on the entailment rules defined in the schema graph SG infers all additional triples.*

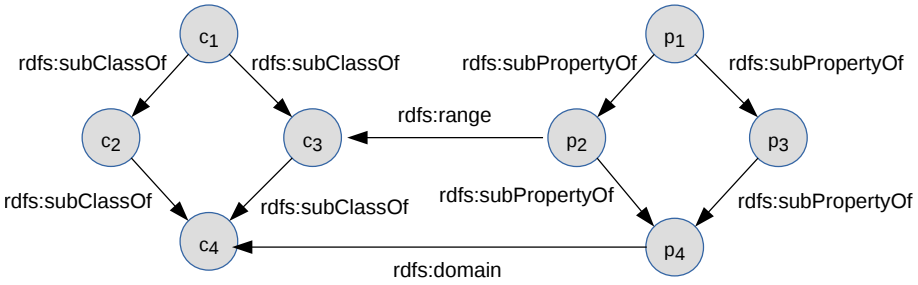


Figure 13: Example of a schema graph.

While the parameterisations introduced above where applied on the schema elements, this is a parameterisation applied on the data graph itself. This way, no additional changes on the schema elements are required.

Equivalence relation chaining

Moreover, it is possible to choose a length n of chained equivalence relations. When chaining n equivalence relations, not only the properties of the instance itself are considered, but also those properties within a n-neighbourhood.

The desired size of the matching subgraph structure can be formalised using a stratified n-bisimulation (Luo, Fletcher, Hidders, Wu, & Bra, 2013). It allows for scaling the size of the subgraph, which has to be equivalent, when assigning instances to equivalence classes. A relation $\sim_n$ for SameAs Instance Sets is defined recursively with n denoting the number of hops and $n \in \mathbb{N}$.

Definition (Chained n extended equivalence relations $\sim_n$ with relation set over instance sets with an extended triple set)**.**

$$
\begin{aligned}
&\textit{If } n = 0 : \sigma(i_1) \sim_0 \sigma(i_2) \Leftrightarrow \Gamma'(\sigma(i_1)) = \Gamma'(\sigma(i_2)).\\
&\textit{If } n > 0 : \sigma(i_1) \sim_n \sigma(i_2) \Leftrightarrow\\
&\qquad \forall (i_v, p_v, o_v) \in DG \cup \Phi(DG, SG_{RDFS}) : p_v \in P_{rel} \ \wedge\ \exists (i_w, p_w, o_w) \in DG \cup \Phi(DG, SG_{RDFS}) :\\
&\qquad p_w \in P_{rel} \ \wedge\ p_v = p_w \ \wedge\ \Gamma'(\sigma(i_v)) = \Gamma'(\sigma(i_w)) \ \wedge\ \Gamma'(\sigma(o_v)) = \Gamma'(\sigma(o_w))\\
&\qquad \wedge \sigma(o_v) \sim_{n-1} \sigma(o_w)
\end{aligned}
$$

Using the presented notion of (Extended) Type Cluster and (Extended) Equivalence Classes we can represent various schema-level indices in a unified way. Although these schema elements are the core of any schema-level index, it is also of certain interest which kind of information is attached to each schema element.

Payload

The payload characterises information of data sources connected to schema elements including, e.g. the contexts of the summarised instances or a certain number of concrete instances. The information stored as payload can be directly accessed in contrast to exclusively storing URIs linking to external content. When designing the payload, a certain trade off between fast access of information and size of the index is to be made. When indexing the LOD cloud storing too much data locally as payload requires a huge amount of disk space. For example one of the largest Linked Open Data dataset the LOD Laundromat dataset contains about 38 Billion RDF-quads, which sums up to more than 6.2 TB of raw data (Beek, Rietveld, Bazoobandi, Wielemaker, & Schlobach, 2014). Nevertheless, designing the payload is highly depending on the use case and we do not want to present general guidelines, but rather introduce a formal notion of the payload and present some possible payload elements.

In the formal model we map schema elements to the corresponding payload elements. Thus, payload elements mediate between the summarising schema elements and the actual data (see depiction Figure 11).

Definition (Payload)**.** *The Payload PAY is n-tuple of mapping functions, which map schema element to specific payload attributes.*

Data source (URI lookup endpoint): The data source function $s : V_{RB} \rightarrow \mathscr{P}(V_R)$ returns all data sources of an instance $i \in V_{RB}$. Alternatively, we could use the notion of (s, p, o, c)-quads, where the context c describes

the data source. Using this function s, we can map schema elements to data sources. As a reminder, schema elements are defined as equivalence classes using equivalence relations over a set of instances. Therefore, we can treat a schema element also as set of instances. All instances from such an equivalent class can then be mapped to the desired payload information.

Definition (Datasource)**.** *Any schema element can be mapped to the corresponding data sources using a function* $source : \mathscr{P}(V_{RB}) \rightarrow \mathscr{P}(D)$*, which takes an (Extended) Equivalence Class* EQC *containing equivalent instances, with*

$$source(EQC) := \bigcup_{i_1 \in EQC} s(i_1).$$

Snippets (human readable preview of the data): Snippets in the field of LOD are triples characterised by specific properties such as *rdfs:label* to further describe the given linked data instance. Snippets are marked by special properties that denote snippet information, e.g. *rdfs:label* or *rdfs:comment*. We call the set of all those properties $P_{snippet}$. The snippet function defined below takes a schema element and such a property as input and returns a set of literals.

Definition (Snippet)**.**

$$snippet(EQC, p_{snippet}) := \{l_1 | (i_1, p_{snippet}, l_1), i_1 \in EQC\}$$

In an practical environment it is also useful to limit the number of snippets per Equivalence Class. Since this limitation is straight forward, we will not formalise it here explicitly.

3.5.3 Experimental evaluation and comparison

The formal model allows to model various schema-level indices. As a proof of concept, we implement the four parameterised schema elements (Extended) Type Cluster and (Extended) Equivalence Class) in our stream-based schema-extraction tool SchemEX (Konrath et al., 2012). This stream-based approach allows us to tackle large scale snapshots of the Linked Open Data (LOD) cloud. Stream-based approaches compute an approximate index over a stream of statements using a window technique from stream databases (Garofalakis, Gehrke, & Rastogi, 2016). We assume a sliding window approach with First-in- First-out (FiFo) replacement strategy and refer to it as cache. In order to evaluate each schema-level index individually with respect to the approximation error, we compute indices with limited cache size and compare them to the gold standard. Each gold standard is computed using the same schema elements and parameters, but with unlimited cache size. We refer to the combination of schema elements and parameterisations as configurations. For the evaluation, we use two datasets with different characteristics, the TimBL11 dataset and the first DyLDO snapshot from May 2012. Although reasonably large, both datasets allow us to compute a gold standard. The TimBL11 dataset contains about 11 Million quads crawled starting from the FOAF profile of Tim Berners-Lee (Konrath et al., 2012). The crawl was conducted in 2011 using the Open Source Crawler LDSpider with a breadth first search (Isele, Umbrich, Bizer, & Harth, 2010). Regular snapshots from LOD cloud are provided by the Dynamic Linked Data Observatory (DyLDO). We use their first snapshot containing about 127 Million quads crawled from about $95,000$ seed URIs in May 2012. This crawl was done using the LDSpider with breadth first search, but limited to a crawling depth of two (Käfer, Abdelrahman, Umbrich, O'Byrne, & Hogan, 2013).

Procedure

We evaluate six different schema-level index configurations over the two mentioned datasets. We first define from the literature the schema-level indices CharacteristicSets and SchemEX. Additionally, we increase or decrease the indices' complexity by changing the chaining parameter n. We expect more complex configurations to have a greater approximation error since more data within the data graph DG needs to be present in the cache. The payload of each configuration comprises the datasource only, which is necessary for the evaluation.

The first configuration is the $CharacteristicSets$ configuration (Neumann & Moerkotte, 2011). Characteristic sets classify RDF resources by the co-occurrence of their incoming predicates P^- and outgoing predicates P^+. CharacteristicSets $:= (DG, EQC_n, (source))$, with P_{rel} containing all properties including rdf:type. Thus, no Type Cluster are considered in the Equivalent Class.

The second configuration is the $SchemEX$ configuration (Konrath et al., 2012). SchemEX computes equivalence relations over types and properties of the instances as well as the types of the instances connected via properties. SchemEX $:= (DG, EQC_n, (source, snippet))$, with P_{type} containing rdf:type and P_{rel} containing all remaining properties. SchemEX uses the snippets $P_{snippet}$ containing *rdfs:label*.

The third configuration $SchemEX + RDFS + SameAs$ extends the $SchemEX$ configuration by means of the SameAs instance set parameterisation and an extended triple set parameterisation using the RDFS schema

graph. SchemEX+RDFS+SameAs := $(\Phi(DG, SG_{RDFS})), eEQC_n, (source))$, with P_{type} containing rdf:type, P_{ign} containing all $RDFS$ properties as well as the owl:sameAs property.

We evaluate if the datasource is assigned to the correct schema element during the index computation. This can be done by executing a set of queries on the index and measure precision and recall of the returned datasources. These queries are designed to address types or properties or a combination of both. It is possible to generate a full list of all possible queries by extracting types, properties and their combination from the gold standard and then aggregate them to queries. For this evaluation, we focused on two kind of queries, the type query (TQ) queries for the directly attached type information (focused on Type Cluster) and the complex query (CQ) additionally queries for properties linking to other instances and their type information (focused on Equivalence Classes). The approximation error is measured by comparing the returned datasources of a query q with regard to the precision, recall and F1-measure. $Precision(q) = \frac{|D_{gold} \cap D_{eval}|}{|D_{eval}|}$ and the recall by $Recall(q) = \frac{|D_{gold} \cap D_{eval}|}{|D_{gold}|}$. Knowing precision and recall, we can compute the commonly known F1-score $F1(q) = 2 \cdot \frac{Precision(q) \cdot Recall(q)}{Precision(q) + Recall(q)}$.

Results

Table 18 shows the results of the experimental evaluation of the above mentioned configurations. For each configuration, we present the results with regard to the chaining parameter n. The F1-score for the TQ and the CQ results are presented in each row. Each configuration is evaluated with cache sizes of 100k instances and 200k instances. Since the evaluation showed very similar precision and recall values, we only report the F1-score in Table 18. From the results of our experiments, we can state that consistently over all indices

Table 18: F1-scores for two kind of queries Q, the type queries (TQ) and the complex queries (CQ), for each configuration with changed equivalence chaining parameter n for different cache sizes (100k, 200k) on the two datasets TimBL11 and the first DyLDO snapshot from 2012.

			CharacteristicSets		*SchemEX*		*SchemEX* +RDFS+SameAs	
	n	**Q**	**100k**	**200k**	**100k**	**200k**	**100k**	**200k**
TimBL11	0	TQ	1	1	.97	.98	.91	.93
	0	CQ	1	1	.98	.98	.92	.94
	1	TQ	1	1	.75	.76	.70	.71
	1	CQ	.77	.78	.44	.46	.42	.45
DyLDO 2012 first	0	TQ	1	1	.57	.58	.47	.47
	0	CQ	1	1	.59	.60	.48	.49
	1	TQ	1	1	.49	.50	.43	.43
	1	CQ	.57	.71	.17	.18	.13	.14

and datasets, a larger cache size improves the accuracy of the approximative schema-level indices. However, after a certain threshold even doubling the cache size does not have a huge impact on the quality any more. Furthermore, for an increasing number of chained equivalence relations we observe that it strongly reduces the evaluation results. We observe about 10% more accuracy loss when using inference with RDF Schema and SameAs Instance Sets. One explanation is that more statements need to be taken into account and thus there are more possible errors. Additionally, those triples may be even more distributed within the dataset than instance related triples. Furthermore, there seems to be a strong influence from the characteristics of the crawled dataset. The configurations performed by an average of about 35% worse on the DyLDO dataset compared to the TimBl11 dataset. Since the TimBL11 dataset was crawled starting from only one seed URI, this might lead to closely related triples in the beginning of the crawl also stored next to each other. The DyLDO dataset was crawled with more than 90,000 seed URIs which were selected to crawl a representative snapshot of the LOD cloud. Therefore the relevant triples are not necessarily stored close to each other. Thus, even doubling the cache size cannot compensate this.

In general simple queries (SQ) have higher F1-scores than complex queries (CQ). This suggests that triples sharing a common subject appear together more often than triples connected over a common intermediate instance. This hypothesis can be supported by the fact that CharacteristicSets, although have a simple schema structure, can only achieve a F1 score of 57%. CharacteristicSets consider incoming properties as well, which

means triples not sharing a common subject. Please note that CharacteristicSets do not distinguish between types and properties, which makes the F1-measure for the type query not comparable to the others and their complex query only considers all directly connected properties (the connected object is ignored by definition of the CharacteristicSet). Therefore the complex query for CharacteristicSet returns comparable results to the type query for the other configurations.

3.5.4 Implementation, APIs and integration

The implementation is done in Java and can be run as stand-alone application. Indexing large collections of Linked Open Data can be considered the first step in integrating this data into MOVING. As depicted in Figure 14, it is one essential component. The "Harvesting Linked Data" component uses the index to retrieve a list of datasources. For this task, a user has to formulate a SPARQL[27] query stating his or her information need. For example the query from Listing 4 is used to find bibliographic metadata in MOVING.

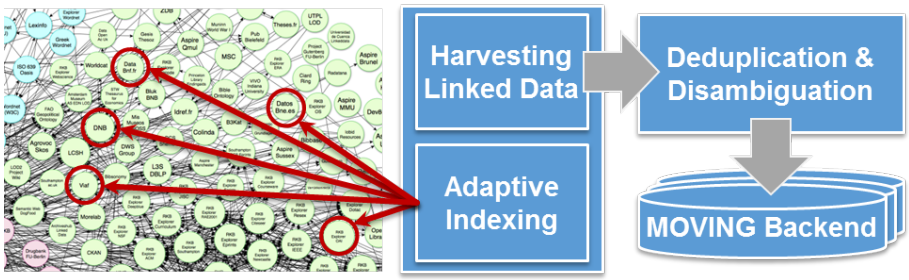


Figure 14: Concept for integration of Linked Open Data into MOVING.

Listing 4: SPARQL schema level query to search for bibliographic metadata on LODatio.

```
SELECT ?x
WHERE {
    ?x rdf:type bibo:Document .
    ?x dcterms:title [] .
    ?x dcterms:description [].
    ?x dcterms:creator [] .
}
```

Each datasource is then harvested individually and all relevant instance information is collected. As a final step, the user needs to provide a mapping configuration. Such a mapping defines which instance information is used for which attribute within our common data model (Section 2). An example mapping is given below:

Listing 5: Example mapping file used to transform bibliographic metadata modelled as Linked Data into MOVING's common data model

```
{  "BibItemMapping":{
      "title":["http://purl.org/dc/terms/title"],
      "abstract":["http://purl.org/dc/terms/description",
                  "http://swrc.ontoware.org/ontology#abstract"],
      "author":["http://purl.org/dc/terms/creator",
                "http://swrc.ontoware.org/ontology#author"],
      "startDate":["http://purl.org/dc/terms/date"],
      "endDate":["http://purl.org/dc/terms/date"],
      "venue":["http://purl.org/dc/terms/isPartOf",
               "http://www.bl.uk/schemas/bibliographic/blterms#publication"
                 ],
      "language":["http://purl.org/dc/terms/language"],
      "keyword":["http://purl.org/dc/terms/subject"]
   },
   "AuthItemMapping":{
      "name":["http://www.w3.org/2000/01/rdf-schema#label",
```

[27] https://www.w3.org/TR/rdf-sparql-query/, last accessed: 15/03/2017

```
                "http://xmlns.com/foaf/0.1/name"]
    }
}
```

Thus, the harvesting component is able to retrieve bibliographic metadata modelled as Linked Open Data and transform it into our common data model. Subsequently, the data is ingested into our common database (Elasticsearch), which is then to be de-duplicated and disambiguated for further use.

3.6 Video processing

3.6.1 Problem statement

Given a lecture video and a set of non-lecture videos, such as videos from YouTube, our target is to retrieve the most relevant non-lecture videos. Since lecture videos typically depict a lecturer in front of a relatively static background, the visual channel of such videos does not provide any useful information concerning the content of the lecture. For this reason, we choose to use only the transcripts of the lecture videos and process them using text-based techniques, as discussed below. Also, videos found in the Web usually provide useful visual information about their semantic content, while textual information is often not available or of no particular interest (e.g. might be in various languages or might not provide important hints about the content of the video). For this, we choose to use solely the visual channel of those videos and process them as discussed below. Our goal is to semantically correlate lecture video transcripts with the visual content of the non-lecture videos. Consequently, our method includes the following steps: temporal fragmentation of the non-lecture videos into shots, concept-based annotation of the generated shots, and transcript analysis on lecture videos in order to translate text transcripts into a set of related visual high-level concepts.

3.6.2 Method description

Video fragmentation

Video shot segmentation aims to partition the video into groups of consecutive frames captured without interruption by a single camera. These elementary structural units, which are called shots, by definition demonstrate a certain degree of temporal and visual affinity, thus constituting a self-contained visual entity. Shot segmentation can be seen as the foundation of most high-level video analysis approaches, being a prerequisite for tasks such as video semantic analysis and fine-grained classification, indexing and retrieval.

A non-lecture video is decomposed into elementary temporal segments by applying shot segmentation. The shots of the video are detected using a variation of the algorithm described in (Apostolidis & Mezaris, 2014). According to the utilised method, the visual content of each video frame is represented by extracting an HSV (Hue Saturation Value) histogram and a set of ORB (Oriented FAST and Rotated BRIEF) descriptors (Rublee, Rabaud, Konolige, & Bradski, 2011), allowing the algorithm to detect differences between a pair of frames, both in colour distribution and at a more fine-grained structure level. An image matching strategy is then applied using the extracted descriptors for assessing the visual similarity between successive or neighbouring frames of the video. The computed similarity scores and their pattern over short sequences of frames are then compared against experimentally pre-specified thresholds and models that indicate the existence of abrupt and gradual shot transitions. The defined transitions are re-evaluated with the help of a flash detector which removes erroneously detected abrupt transitions due to camera flashes, and a pair of dissolve and wipe detectors (based on the methods from (Su, Liao, Tyan, Fan, & Chen, 2005) and (Seo, Park, & Jung, 2009) respectively) that filter out wrongly identified gradual transitions due to camera and/or object movement. Finally, the union of the resulting sets of detected abrupt and gradual transitions forms the output of the applied technique.

Concept detection

Each non-lecture video shot is annotated with a set of high-level visual concepts and for that two different approaches are followed. First, we performed video concept detection in order to annotate video shots based on 1000 ImageNet and 346 TRECVID SIN concepts (e.g. water, aircraft, etc.). To obtain scores for the 1000 ImageNet concepts, we applied five pre-trained ImageNet deep convolutional neural networks (CNNs) on the test keyframes. The output of these networks is averaged in terms of arithmetic mean to obtain a single score for each of the 1000 concepts. To obtain the scores for the 346 TRECVID concepts we fine-tuned (FT) two of the above pre-trained ImageNet networks on the 346 concepts using the TRECVID 2013 SIN development dataset. We experimented with many FT strategies and we selected the ensemble of FT networks that reaches the best accuracy. We used concept detection on two different experimental settings: video "annotation" and "indexing". The latter two terms refer to the way we evaluate the results of concept detection: at the

video level (where we want the top concepts detected for each video separately to match the visual content of that video), and at the video collection level, where we evaluate concept detection by formulating it as a concept-based video retrieval problem. With respect to the video annotation problem, the output of the FT networks was fused in terms of arithmetic mean in order to return a single score for each concept. Regarding the video indexing problem, the last fully-connected layer of each FT network was used as feature to train SVM (Support Vector Machines) classifiers separately for each FT network and each concept. Then, the SVM classifiers were applied on the test keyframes and the prediction scores of the SVMs for the same concept were fused in terms of arithmetic mean in order to return a single score for each concept. We used different methodologies for the two problems because we did not find a single one performing the best for both of the problems.

Furthermore, for the video annotation, we also investigated the case of learning with uncertainty in input. Uncertainty is ubiquitous in many visual understanding problems. For instance, in problems such as those of video concept-based or event-based annotation, measurement inaccuracies or artefacts of the feature extraction process may contaminate the training examples with noise that can affect the discriminative power of the feature representation scheme. We tried to deal with such noisy training examples by introducing input uncertainty in the standard linear and kernel SVM paradigm using the hinge loss function. In this SVM extension, which we call SVM with Gaussian Sample Uncertainty (SVM-GSU), input uncertainty is modeled using the multivariate (anisotropic) Gaussian distribution. That is, each training example is treated as a random vector, normally distributed with given mean and covariance matrix, which are either estimated simultaneously with the feature extraction process, or modeled by a process by which new data are generated for each distribution. We validated SVM-GSU in various visual understanding problems, such as image classification (Tzelepis, Mezaris, & Patras, 2015), video event detection (Tzelepis, Mezaris, & Patras, 2016), video aesthetic quality assessment (Tzelepis, Mavridaki, Mezaris, & Patras, 2016), achieving promising results in terms of classification and retrieval performance.

We also started to investigate the introduction of input uncertainty in the CNN framework, where we have tried to extend the work made in the SVM's case. More specifically, we tried to modify a well-known loss function usually used at the top of a CNN, the multi-class hinge loss, so that input data are treated as multivariate normal vectors with given means and variances, inspired by the respective modification of the loss of SVM-GSU. First, we directed our efforts towards developing various "uncertainty-aware" building boxes, typically used in CNNs, i.e. convolution and fully-connected layers, pooling layers, etc. However, inserting uncertainty and propagating it through the network proved to be inefficient in terms of complexity and training time. For this reason, we decided to work towards a data-augmentation-based approach, in which each input image will be transformed using a set of typical geometric transformations, such as rotation, translation, mirroring, etc, so that each "attacked" version of each input datum to be used in order to estimate a measure of uncertainty of the respective training example. This uncertainty measure, in the form of a covariance matrix will be subsequently used at the top of the network in a modified hinge loss function, similarly to this proposed in the SVM-GSU case, as discussed above. However, this is work in progress and its effectiveness needs to be evaluated in the coming months.

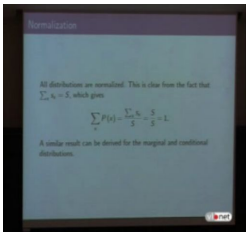
Transcript analysis

In order to translate the transcript of a lecture video into a set of pre-defined visual concepts we adopt a variation of the method presented in (C. Tzelepis & Patras, 2016). The transcript of a video is translated into a set of high-level concepts and finally every video is represented by a set of the k most related concepts. Each concept is assigned with a score that indicates the degree the concept is related to the video transcript.

A Transcript Language Model (TLM) and Concept Language Models (CLM) are built similarly to (C. Tzelepis & Patras, 2016). A TLM is a set of N keywords that are extracted from the corresponding transcripts of a video shot. Each transcript is transformed in a Bag-of-Words (BoW) representation and the N most frequent keywords are selected. Similarly to TLM, a CLM is a set of M words or phrases that are extracted with respect to a specific concept definition. A CLM is built for each concept using the top articles in Wikipedia. The retrieved articles are transformed in a BoW representation from which the top-M words, which are the most characteristic words of this particular visual concept, are kept. For example, the top retrieved words for the concept "*palace*" are "*palace*","*crystal*", "*theatre*", "*season*", "*west*", "*east*", "*spanish*", "*gates*", "*hotel*". After building the TLM and CLMs, we calculate a single value per concept that denotes the semantic relation of this concept with the TLM. For each CLM we calculate a $N \times M$ distance matrix W. Each element of the matrix contains the semantic relatedness between pairs of words appearing in the TLM and CLM. The Explicit Semantic Analysis (ESA) measure (Gabrilovich & Markovitch, 2007) was used to calculate the semantic relatedness of two words or phrases. Given the matrix W, a single score is calculated by applying to W the Hausdorff distance, defined as $D_{\mathscr{H}}(\text{EML}, \text{CLM}) = \text{median}_p\left(\max_p \|w_i - w_j\|\right)$. A single value is calculated per

concept, by repeating the above process for every CLM and the k concepts with the higher value are selected for representing the video transcript. In Table 19 two examples are presented in order to illustrate the results of the above procedure.

Table 19: Representative lecture videos along with transcripts samples and their concept representation.

Video Title	Keyframe	Transcript Sample	Keywords	Top Concepts
Learning with Probabilities		..similar looking things as you throw away more data they start looking less similar, so that's the only thirty percent of the data, and that fifty percent of the data being thrown away it's probabilistic approach helps you..	data things think covariance function noise way	analog_computer .054 hard_disc .045 computer .042 Kernel .037 measuring_instrument .036 EntleBucher .035 memory_device .033 portable_computer .033
A Mosque in Munich Nazis, the CIA, and the Rise of the Muslim Brotherhood in the West		..western base for the muslim brotherhood and for many years was a key centre for this group the muslim brotherhood you may have heard of may be familiar with but it is these were islamic group if you think of political islam as a tree the muslim brotherhood might be..	world ramadan people brotherhood war union mosque muslim muslims	muslims 0.555 3_or_more_people 0.521 old_people 0.521 two_people 0.52 mosques 0.5097 dark skinned_people 0.46 people_marching 0.4056 asian_people 0.244

3.6.3 Experimental evaluation and comparison

Video fragmentation

The shot segmentation method is evaluated on the same dataset used in (Apostolidis & Mezaris, 2014). The updated and improved method achieves higher precision and recall scores in comparison with the original method (Table 20). Moreover, the new orb-based multi-threaded method conducts the processing at least two times faster than the previous SURF-based GPU-accelerated method.

Table 20: Comparison of the shot segmentation experimental results.

Method	Precision	Recall	F-score	Processing time (% of video duration)
Original (Apostolidis & Mezaris, 2014)	0.887	0.917	0.902	30%
MOVING updated version	**0.942**	**0.941**	**0.942**	**13%**

Concept detection

We evaluate the concept detection method by generating scores for the 346 TRECVID SIN concepts. We experimented with four pre-trained ImageNet networks (ResNet-50, two variations of GoogLeNet, AlexNet). Each of these networks was fine-tuned on these concepts using the TRECVID SIN 2013 development dataset. Specifically, each pre-trained network was extended by one fully-connected layer. The size of the extension layer was examined for 7 different dimensions: 64, 128, 256, 512, 1024, 2048, 4096. Furthermore, the classification layer of the pre-trained networks was replaced by a 346-element fully-connected layer. This process resulted to 28 fine-tuned networks. Instead of using all them, we kept the best ensemble of networks that consists of a maximum of N of them. The size of the ensemble, (N), was evaluated for: 1, 2, 5 and 14 networks. We evaluated two different sets of concept scores for the TRECVID SIN concepts: The direct output of the N fine-tuned networks was fused in terms of arithmetic mean in order to return a single score for each concept. The last fully-connected layer was used as feature to train SVM classifiers separately for each fine-tuned

network and each concept. Subsequently, the SVM classifiers were applied on the TRECVID SIN test dataset and the prediction scores of the SVMs for the same concept were fused in terms of arithmetic mean in order to return a single score for each concept. We evaluated the two different sets of concepts on the TRECVID SIN 2013 test set that consists of 112,677 representative keyframes and 38 semantic concepts in terms of MXInfAP. Extended infAP combines the simplicity of random sampling with the efficiency of stratification and thus it is simple and easy to compute while, at the same time, it is much more efficient than infAP in terms of reducing the judgement effort. The indexing problem was examined; that is, given a concept, return the 2,000 test keyframes that are more likely to represent it. The MXInfAP of each set of concept scores for different ensembles of FT networks (i.e., different values of N) is presented in Table 21.

Table 21: Concept detection experimental results.

#N of fine-tuned networks (fused in terms of arithmetic mean)	**Direct output**	**SVM-based detectors**
14-best	29.37	33.91
5-best	29.08	35.27
2-best	**30.04**	**35.81**
1-best	29.99	33.01

According to Table 21 the best result in terms of MXInfAP was reached by an ensemble of 2 FT models. The results indicate that increasing the number of the detectors that will be combined for the same concept does not increase the overall accuracy, i.e., combining 14 detectors is worse than combining 2 detectors. This indicates the importance of selecting the most suitable subset of detectors that are available for a concept. In addition, combining only 2 detectors is more efficient in terms of execution time.

Combined video and textual analysis

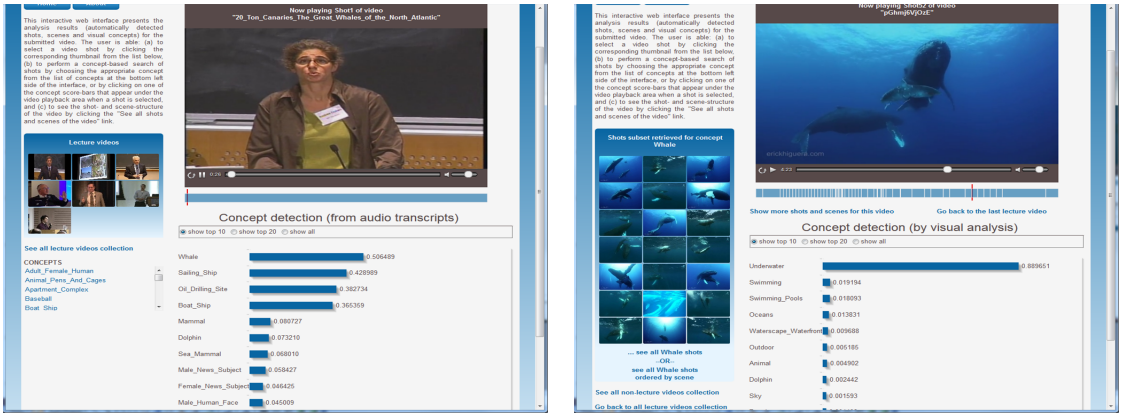
The evaluation of transcript analysis method and its combination with the video analysis was performed on the TRECVID AVS 2016 (AVS16) (Awad et al., 2016) and Video Search 2008 (VS08) (Over et al., 2008) datasets. For that, the experimental setup of the Trecvid Ad-hoc video search task were followed. The objective of AVS (Ad-hoc Video Search) task is, given a text query, to retrieve the 1,000 videos shots that are most related with it, where no annotated training video samples are available. AVS text queries can be considered as lecture transcripts equivalents, while the video shots of the non-lecture videos are the equivalent of the videos that need to be retrieved in AVS. Furthermore, we analyse our results in terms of mean extended inferred average precision (MXInfAP), which is an approximation of the mean average precision suitable for the partial ground-truth that accompanies the TRECVID dataset (Yilmaz, Kanoulas, & Aslam, 2008).

We compared our method with seven different literature methods. The top part of the Table 22 refers to those methods that were re-implemented in order to be adapted for this problem and datasets, whereas in the lower part we introduce the results of the top-four finalists in the AVS16 and the VS08 tasks. Overall, for both datasets, our proposed method performs very well in this challenging task, compared to the other methods. Specifically, it outperforms all the compared methods, achieving an MXinfAP of 6.35% and 9.11% for AVS16 and VS08, respectively.

Table 22: Mean Extended Inferred Average Precision (MXInfAP) for different compared AVS methods. A higher MXInfAP percentage is better.

Methods	**AVS16**		**VS08**	
(a) Literature methods				
	(Ueki et al., 2016)	5.65	(C. Tzelepis & Patras, 2016)	8.27
	(C. Tzelepis & Patras, 2016)	4.16	(Norouzi et al., 2013)	7.30
	(Norouzi et al., 2013)	3.14	(Ueki et al., 2016)	7.24
(b) Top-4 TRECVID finalists				
Top-1	(Duy-Dinh et al., 2016)	5.4	(Juan et al., 2008)	6.7
Top-2	(Markatopoulou et al., 2016)	5.1	(C.G.M. et al., 2008)	5.4
Top-3	(Junwei et al., 2016)	4.0	(Ngo et al., 2008)	4.2
Top-4	(Zhangy et al., 2016)	3.8	(Mei et al., 2008)	4.1
Proposed		**6.35**		**9.11**

3.6.4 Implementation, APIs and integration



(a) (b)

Figure 15: (a) The list of related concepts extracted from the lecture video's transcripts is shown beneath the playback, (b) related non-lecture video fragments after clicking on the concept "whales".

CERTH video analysis REST service

For the visual analysis of the videos, CERTH hosts a REST web service that performs shot segmentation and concept detection utilising the methods analysed above. The service can handle videos hosted on file servers, or download from some video hosting platforms, such as YouTube. The videos are temporarily downloaded, for legal reasons, and only the metadata extracted are stored after the processing. Communication between the user and the service is done via HTTP POST and GET calls, and the processing results can be retrieved in XML and JSON formats. The results include the temporal segmentation of the video, the shots' visual concept annotation and keyframes of the video. This service is intended to be an external component of the MOVING platform and will be accessed via its REST API, so that its results will be communicated automatically to and be displayed in the MOVING platform.

For example, to perform shot segmentation and concept detection on a video, we issue the following POST request:

```
POST http://multimedia2.iti.gr:8080/shot-scene-concept
{
'video_url': <url>,
'user_key': <key>,
}
```

The attribute 'video_url' in the request body, contains the url of the video to be processed, while 'user_key' is used for authentication

Obtaining the concept detection processing results of a video in XML format, can be done by issuing the following GET request

```
GET http://multimedia2.iti.gr:8080/result/<video_name>_concepts
```

Lecture video linking demo

To demonstrate the whole process of linking lecture with non-lecture videos, CERTH created a web demo[28] for the MOVING project. This interactive web interface links lecture videos, using general purpose concepts that were produced from textual analysis of their transcripts, with non-lecture videos, using their visual analysis results such as automatically detected shots and visual concepts. The user is able to select a lecture video by clicking its corresponding thumbnail. During the playback of the selected lecture video, the automatically detected concepts that characterise it are listed on the screen as shown in Figure 15a. By clicking on any one of the concepts, additional temporal segments of non-lecture videos that are related to the selected concept are presented to the user (Figure 15b).

[28] http://multimedia2.iti.gr/moving-project/lecture-video-linking-demo/results.html, last accessed: 28/03/2017

3.7 Author alignment

3.7.1 Problem statement

Documents have authors. This information is almost always available on a document and in the document's metadata. However, it is crucial to distinguish an author name mentioned on a specific document from the real-world author. It is not very often the case that author identifiers from authority files are assigned to author names on a document. Usually, the author is referred to by a string of characters that is given with the document. This concept introduces two types of ambiguity:

1. The same author may be referred to by different author name strings (synonymy). This can be a result of misspelling, language-specificity, different conventions for first, middle and last names, etc.
2. The same author name string might refer to different authors (homonymy). If a name is not disambiguated, we have the problem to map a single author name mentioned to the right author. With the size of the document collection, the chance increases that two different authors in the collection have the same name. A generalising normalisation (i.e. use only initials) makes this even more likely.

The first kind of ambiguity can in many cases be addressed by a normalisation schema that makes sure that different versions of the same name are assigned the same generalisation: in a first step, the string referring to an author is parsed into different subfields; in a second step, these fields are merged again, with possible generalisations on their content (i.e. initials). While this is not always trivial, we focus on the second kind of ambiguity for now.

Formal problem definition

With respect to the second kind of ambiguity, namely authorship disambiguation or author name disambiguation decides for a set of author mentions with the same name, which of them belong to the same author and which do not (Smalheiser & Torvik, 2009). This is a clustering problem over the author mentions (see Figure 16). Each cluster is considered an author. More formally:

- For each collection, there is a set $\mathcal{N}$ of names. Each $name \in \mathcal{N}$ is a string representing of an author.
- For each name $name \in \mathcal{N}$, there is a set $\mathscr{C}$ (also referred to as a clustering) of authors $C \in \mathscr{C}$ (also referred to as a cluster) and a set X of mentions $x \in X$. A mention is a name instance in a document of the collection. So, each name can be considered as a tuple of (authors, mentions).
- Each author C is a set of mentions $x \in C$ and a subset $C \subseteq X$. The task of author alignment is to determine which mention x belongs to which author C.
- For each mention x, there is a bag of features $f \in F(x)$, each with a frequency $\#(f,x)$ of occurrence with x.

If a name is not disambiguated, this state can either be expressed by putting all mentions of the name under study in the same cluster $C = X$ such that $\mathscr{C} = \{\{x \mid x \in X\}\}$, or by considering each mention x as an own cluster $C = \{x\}$, such that $\mathscr{C} = \{\{x\} \mid x \in X\}$. The task of author disambiguation is then to suggest a system clustering $\mathscr{C}_{sys}$ that is as close to the correct clustering $\mathscr{C}_{cor}$ as possible. In the training/tuning and evaluation case, we have both $\mathscr{C}_{sys}$ and $\mathscr{C}_{cor}$ present and optimise some evaluation score $eval(\mathscr{C}_{sys}, \mathscr{C}_{cor})$. We can safely assume that this score measures the similarity between the system clustering and the correct clustering. Please note that when disambiguating a name $name$, we do not need to consider any other names $name' \in \mathcal{N}$. In practice, this strongly reduces the complexity of the problem, as we can disambiguate one name at a time. This procedure is referred to as blocking, where each name defines one block.

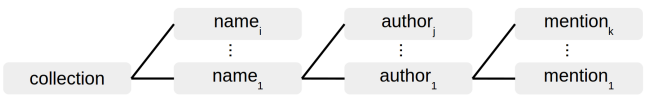


Figure 16: Tree representing the structure of the author disambiguation problem.

Table 23: Overview of the literature regarding the research field of author disambiguation.

overv.		(Ferreira et al., 2012)
		(Smalheiser & Torvik, 2009)
problem an.	***impact***	(Harzing, 2015)
		(Strotmann & Zhao, 2012)
	ground truth	(Krämer et al., 2017)
author disambiguation method	***general***	(Gurney et al., 2012)
	specific	(Culotta et al., 2007)
		(Tan et al., 2006)
		(L. Tang & Walsh, 2010)
	neural network	(H. N. Tran et al., 2014)
	probabilistic	(Han et al., 2005)
		(J. Tang et al., 2012)
		(Torvik et al., 2005)
		(Torvik & Smalheiser, 2009)
	topic-based	(Song et al., 2007)
		(K.-H. Yang et al., 2008)
	baseline	(Milojević, 2013)
	co- or ref-authors	(Kang et al., 2009)
		(F. H. Levin & Heuser, 2010)
		(G.-C. Li et al., 2014)
		(Schulz et al., 2014)
	focus on training	(Ferreira et al., 2010)
		(M. Levin et al., 2012)
		(Qian et al., 2011)
	dynamic	(Khabsa et al., 2015)
		(Qian et al., 2015)

Related work

As shown in Table 23, the literature on author disambiguation can be coarsely classified into (1) overview papers, (2) papers analysing the problem as such and (3) papers presenting a method to disambiguate author names. While Ferreira et al. (2012) give a short overview of the field including a classification of different problem setups and methods, Smalheiser and Torvik (2009) go into more detail.

We consider two types of contributions regarding the analysis of the author ambiguity problem: (2.a) papers assessing the impact that author name ambiguity (Harzing, 2015) vs. author name disambiguation (Strotmann & Zhao, 2012) have on further processing of the respective data; as well as (2.b) papers analysing the availability of a ground truth that can be used to train and evaluate methods of author disambiguation (Krämer, Momeni, & Mayr, 2017).

Furthermore, we distinguish the following methods of author disambiguation: (3.a) general solutions that do not have a specific methodological focus (Gurney, Horlings, & Van Den Besselaar, 2012); on the other hand (3.b) papers that present a specific method to solve the ambiguity problem, i.e. using ranking loss (Culotta, Kanani, Hall, Wick, & McCallum, 2007) or other techniques (Tan, Kan, & Lee, 2006; L. Tang & Walsh, 2010); (3.c) solutions based on neural networks (H. N. Tran, Huynh, & Do, 2014); (3.d) probabilistic solutions like ours (Han, Xu, Zha, & Giles, 2005; J. Tang, Fong, Wang, & Zhang, 2012; Torvik & Smalheiser, 2009; Torvik, Weeber, Swanson, & Smalheiser, 2005) – here we note that our approach varies considerably from these approaches; (3.e) solutions using topic modeling (Song, Huang, Councill, Li, & Giles, 2007; K.-H. Yang, Peng, Jiang, Lee, & Ho, 2008); also (3.f) baseline approaches focusing on very simple methods (Milojević, 2013). Furthermore, a number of papers focus on (3.g) the exploitation of features related to co- or referenced authors using different methods (Kang et al., 2009; F. H. Levin & Heuser, 2010; G.-C. Li et al., 2014; Schulz, Mazloumian, Petersen, Penner, & Helbing, 2014). Some papers focus on (3.h) specific ways of training, i.e. semi-supervised learning (M. Levin, Krawczyk, Bethard, & Jurafsky, 2012; Ferreira, Veloso, Gonçalves, & Laender, 2010) or human-controlled learning (Qian, Hu, Cui, Zheng, & Nie, 2011). Also, some recent papers explore (3.i) options for developing and integrating methods that can cope with dynamically growing collections (Khabsa, Treeratpituk, & Giles, 2015; Qian, Zheng, Sakai, Ye, & Liu, 2015).

We classify our approach as probabilistic (3.d). We focus to some extend on the training aspect (3.h), as we found that our approach requires almost no training data. We also discuss certain (natural) baselines (3.f), measure the importance of co- and reference authors for our model (3.g) and provide some insight into the distribution of ground truth labels (2.b) in the Web of Science.

3.7.2 Method description

Using the blocking paradigm, our method disambiguates one name at a time, i.e. it clusters all mentions of that name based on the features that can be extracted from the collection for that mention. Features are extracted in a preprocessing step. In this context, we note that for each mention x, there is exactly one document $d(x)$ on which this mention appears. However, for each document, there can be multiple mentions that appear on it.

Features

Features $F(x)$ assigned to a mention x can be extracted from document d in general or from information given specifically for x on d. We will distinguish between feature types that are specific for x (such as affiliation) and feature types which address the document as a whole (such as keywords). The following information is used:

1. $F_{term}(x)$: Feature-type terms: A bag of words that contains all words considered relevant in the text fields of the document and their frequency of occurrence in $d(x)$.
2. $F_{aff}(x)$: Feature-type affiliations (specific): A bag of affiliations given for x on $d(x)$. This is usually just a single affiliation with a frequency of 1.
3. $F_{cat}(x)$: Feature-type categories: A bag of categories assigned to $d(x)$, where we consider categories to be relatively general terms that are picked w.r.t. a relatively small vocabulary of classifications. The frequency of one category for a document d is usually 1.
4. $F_{key}(x)$: Feature-type keywords: A bag of keywords assigned to $d(x)$, where we consider keywords to be relatively specific terms that are picked w.r.t. a relatively large vocabulary or only with respect to the current document. The frequency of one keyword for a document d is usually 1.
5. $F_{co}(x)$: Feature-type coauthornames (specific): A bag of names of the coauthors of x on $d(x)$. Unless more than one author of d have the same name, the frequencies are 1. This feature-type is specific only in that the author mention x is not repeated as its own coauthor.
6. $F_{ref}(x)$: Feature-type refauthornames: A bag of names given as authors of all documents d' referenced by d. Frequencies larger than 1 will occur if multiple documents by the same author are referenced.
7. $F_{email}(x)$: Feature-type emails (specific): A bag of email addresses given for x on $d(x)$. This is usually just a single email address with a frequency of 1.
8. $F_{year}(x)$: Feature-type years: The year given for $d(x)$, e.g. the publication year with a frequency of 1.

While there are many details related to the question of which and how features are extracted and normalised, the focus of our research was not to investigate specific features but to develop a method that can provide satisfying results independent of the exact set of features and feature-types. This independence will be proven in the evaluation section.

Agglomerative clustering

We apply a method of agglomerative clustering presented in Algorithm 1. This means that we start with the initial state mentioned in the previous section where each mention x of a name is considered as an own cluster $C = \{x\}$. Then, pairs (C, C') of clusters are merged. If no stopping criterion is applied, this will ultimately result in a state where all mentions are in the same cluster $C = X$. If merging is random, this will not result in satisfying clusters either. For this reason, we need to compute the score $score(C, C')$ of a pair (C, C') of clusters to be merged. Furthermore, we apply a quality threshold (limit) l, that tells us whether the score can be considered good or not. In our approach, $score(C, C')$ is not dependent on the score of any other pair of clusters. Neither is the quality limit. This means that in each iteration of the clustering process, we merge all pairs (C, C'), such that the scores of all pairs of C and C' with a third cluster C'' are not larger than $score(C, C')$, i.e. (1) $\forall C'' \in \mathscr{C} : score(C'', C) \leq score(C, C') \wedge score(C'', C') \leq score(C, C')$ and at the same time, (2) $score(C, C') > l$. In other words, we evaluate all pairs $(C, C') \in \mathscr{C} \times \mathscr{C}$; for each of these pairs, we check whether (1) and (2) hold true. If yes, the pair is saved for merging. At the end of each iteration, all saved pairs are merged. A new system clustering is obtained and the next iteration begins. This process converges if no pairs are saved for merging. For evaluation purposes, we can continue to merge with moves that are below the threshold. We will elaborate on this in the experiments section.

Probabilistic scores

The main contribution of our approach is the similarity used to define $score(C, C')$. We define the score of a pair of clusters as the conditional probability $p(C|C')$, which is basically the normalised ratio of feature frequencies related to a mention $x \in C$. In more detail we define:

$$p(C|C') = \sum_{x \in C} p(C|x) \cdot p(x|C') \tag{1}$$

$$p(C|x) = \begin{cases} 1 & \text{if } x \in C \\ 0 & \text{else} \end{cases} \tag{2a}$$

$$p(x|C') = \sum_{x' \in C'} p(x|x') \cdot p(x'|C') \tag{2b}$$

Algorithm 1 Our agglomerative clustering approach (without evaluation)

Given *name* with X and $\mathscr{C}$ as well as $N, \#(f), \lambda, \varepsilon, l$

while $|\mathscr{C}| > 1$:

  foreach $(C, C') \in \mathscr{C} \times \mathscr{C}$:
    set $scores(C, C') := \sum_{ftype} \lambda_{ftype} \cdot p_{ftype}(C|C')$;
  init $Marked4Merge := \emptyset$;
  foreach $(C, C') \in \mathscr{C} \times \mathscr{C}$:
    if $\forall C'' \in \mathscr{C} : score(C'', C) \leq score(C, C') \wedge score(C'', C') \leq score(C, C')$ **and** $score(C, C') > l$:
      $Marked4Merge \leftarrow (C, C')$;
  if $Marked4Merge$ is empty:
    break
  foreach $(C, C') \in Marked4Merge$:
    merge $C + C'$;

$$p(x|x') = \sum_{f \in F(x)} \frac{\#(f,x) \cdot \#(f,x')}{\#(f) \cdot \#(x')} \tag{3a}$$

$$p(x'|C') = \frac{[x' \in C'] \cdot \#(x')}{\#(C')} \tag{3b}$$

$$\#(f) = \sum_{x' \in X} \#(f, x') \tag{4a}$$

$$\#(C') = \sum_{x' \in C'} \#(x') \tag{4b}$$

$$\#(x') = \sum_{f' \in F(x)} \#(f', x') \tag{4c}$$

Please note that $\#(f,x)$ denotes the frequency of f in $F(x)$ and consider $\#(f,x) = 0$ if $f \notin F(x)$. To prevent division by zero, we apply $add\varepsilon$ smoothing. This slightly modifies $p(x|x')$ and $p(x'|C')$.

Feature-type weights

All the probabilities shown above are obtained separately for each feature-type. Consider that each probability p should actually be denoted p_{ftype} where $ftype$ is either `term`, `aff`, `cat`, `key`, `co`, `ref`, `email` or `year`. For better readability, we drop the subscript where it is not necessary. We perform a simple linear combination with feature-type weights to obtain the final score:

$$score(C, C') = \sum_{ftype} \lambda_{ftype} \cdot p_{ftype}(C|C') \tag{5}$$

Feature-type weights λ are trained on the training portion of our data. For this, we sample pairs (x, C) and (x, C') such that $x \in C \wedge x \notin C' \wedge |C| = |C'| \wedge C \cup C' \subseteq X$, where X is the set of mentions for a single name. All possible values for $|C| = |C'|$ are considered in order to create a more or less realistic binary classification scenario, where we are asked to assign x to a correct cluster C or an incorrect cluster C'. The classifier receives probabilities $p_{ftype}(x|C)$ and $p_{ftype}(x|C')$ for each $ftype$ together with the class 'correct' or 'incorrect'. It then learns feature-type weights λ_{ftype} in order to optimise the classification outcome.

Convergence

Above, we have introduced a quality threshold l on the scores for merges during clustering. In order to account for different clustering sizes (and corresponding smaller probabilities), we define l as follows:

$$l = \alpha + |X| \cdot \beta$$

where $|X|$ is the number of mentions for the current name. While our results are w.r.t. the above form, we recommend to use $l = \alpha \cdot (1 + |X| \cdot \beta)$ where α can be used as a scaling factor for different collections or choices of hyperparameters. Fortunately, when they are normalised such that $\sum_{ftype} \lambda_{ftype} = 1$, the stopping criterion is relatively independent of the feature-type weights λ.

3.7.3 Experimental evaluation and comparison

In the following, we describe the experimental setup used to train, tune and test our approach.

Data

To train and test our approach we use the Web of Science (WoS)[29] as a major source of scientific documents since it contains annotated authorship information, in particular unique author IDs. In a preprocessing step, we extract features for the feature-types mentioned in the previous section. The exact fields in the WoS data are given in Table 24. We normalise author names as "LASTNAME, INITS", where INITS are all the initials for each first name of the author mention. Thereby we do not use all available information for disambiguation, which allows us to challenge our method with a harder setup. We extract terms and their frequency from the title and abstract of the metadata. Title terms are weighted three times higher than abstract terms. Some stop words are omitted and all words are lowercased. Furthermore some basic lemmatisation from the Natural Language Toolkit[30] is applied. Affiliations are already normalised in the WoS. Categories and keywords are taken as they appear in the WoS. Note that we use three different WoS fields for our categories (headings, subheadings, subjects, see Table 24), but do not distinguish between them at any later time. The co- and referenced author names are normalised in the same way as the names that form name blocks. For co-authors, they can be extracted directly from the document of the mention. For referenced authors, we first have to look them up in the referenced document. Emails are not normalised, but we plan to lowercase them in the future. From the publication date, we only pick the year which constitutes the only integer feature of our data.

Table 24: The exact Web of Science (WoS) fields from which the features are extracted, by feature-type. There are potentially multiple instances of the subtree under the main branch.

Field	Main branch	Subtree
names	summary/names/	name/wos_standard
coauthor link	summary/names /name/	seq_no
refauthor link	fullrecord_metadata/references/	reference/uid
term	summary/titles/ fullrecord_metadata/abstracts/abstract/abstract_text/	title p
aff	fullrecord_metadata/addresses/	address_name/address_spec /organisations/organisation
author link	fullrecord_metadata/addresses/ address_name/address_spec	addr_no
cat	fullrecord_metadata/category_info/headings/ fullrecord_metadata/category_info/subheadings/ fullrecord_metadata/category_info/subjects/	heading subheading subject
key	fullrecord_metadata/keywords/	keyword
coauthor link	summary/names /name/	seq_no
refauthor link	fullrecord_metadata/references/	reference/uid
email	summary/names/	name/email_addr
year	summary/ pub_info/	pubyear

From the entire WoS with more than 100 million documents we extract features from all documents where a persistent author ID is assigned to at least one author. In WoS authors can register their ID by self-registration, either as subscribers of WoS or at the ResearcherID website[31]. By adding publications to their profile authors establish the link between their ResearcherID and the publication metadata in WoS (Krämer et al., 2017). However, in WoS just a very small fraction of authors are annotated by a unique researcher ID. This is up-to-now the case for approximately 290,000 authors (cf. (Krämer et al., 2017)). We use this fraction as our test collection. This corpus contains about 250,000 different author names (normalised versions, as

[29] `https://webofknowledge.com/`, last accessed: 28/03/2017
[30] `www.nltk.org`, last accessed: 28/03/2017
[31] `http://www.researcherid.com/`, last accessed: 28/03/2017

described above). Thus, we are concerned with about 40,000 real cases of name ambiguity. We create a database containing the features related to each single author name mention. For each name, we use all the mentions that are given a researcher ID and we use all the authors that contain at least one such mention. As stated earlier, we consider each researcher ID a distinct author. Names are ordered randomly and separated into training and testing portions. As name blocks are considered separate clustering problems, they constitute the units distributed over training and testing data.

Test environment

We are specifically interested in evaluating the performance of our model in relation to the size of the document collection. In MOVING, this is of particular interest since we integrate large collections from various sources. Thus, we have a high probability that a name refers to a number of different real-world authors. We compare a maximum of $1,000$ names for all clusters $|\mathscr{C}_{cor}| \in \{1..10\}$. From this selection 25% is used for training. The clustering sizes are distributed according to Zipf's law. For the sizes 1 to 4, more than $1,000$ names are available (but only a random sample of $\sim 1,000$ used). See Table 25 for exact number of names for each clustering size.

Table 25: The number of names found or used with a clustering size $|\mathscr{C}| \in \{1..10\}$ in the Web of Science (WoS) data.

$\lvert\mathscr{C}\rvert$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	$\sum$
train	255	250	250	250	215	139	80	65	43	35	1582
test	767	751	750	750	645	418	242	195	131	106	4749
train+test	1022	1001	1000	1000	860	557	322	260	174	141	6331
overall	229653	14108	3630	1657	860	557	322	260	174	141	251362
% used	0.45	7.10	27.55	60.35	100	100	100	100	100	100	2.52

We evaluate our approach for each size separately. Furthermore, in order to view the behaviour of our method, we monitor the development of precision, recall and F1 measure with each iteration of the clustering process. We also monitor how the clustering would have continued if there had not been a threshold l on the score of possible merges. Thus, for each iteration in the clustering process, we record the following information:

1. Precision of current system clustering.
2. Recall of current system clustering.
3. Current number of clusters in system clustering.
4. Whether the current iteration is before or after convergence.

Evaluation measures

In order to evaluate the performance of our approach, we use two popular evaluation measures for clustering: (1) pairwise F1 (pairF1) and (2) bCube. Both measures define precision (Pr) and recall (Re) when comparing two clusterings $\mathscr{C}_{sys}$ and $\mathscr{C}_{cor}$. F1 is defined as usual as $2 \cdot \frac{Pr \cdot Re}{Pr+Re}$. According to Levin et. al. (2012) we define:

$$Pr_{pairF1} = \frac{pairs(\mathscr{C}_{cor}) \cap pairs(\mathscr{C}_{sys})}{pairs(\mathscr{C}_{sys})} \quad (6a) \qquad Pr_{bCube} = \frac{1}{|X|} \cdot \sum_{x \in X} \frac{|C_{sys}(x) \cap C_{cor}(x)|}{|C_{sys}(x)|} \quad (7a)$$

$$Re_{pairF1} = \frac{pairs(\mathscr{C}_{cor}) \cap pairs(\mathscr{C}_{sys})}{pairs(\mathscr{C}_{cor})} \quad (6b) \qquad Re_{bCube} = \frac{1}{|X|} \cdot \sum_{x \in X} \frac{|C_{sys}(x) \cap C_{cor}(x)|}{|C_{cor}(x)|} \quad (7b)$$

where $pairs(\mathscr{C}) = \bigcup_{C \in \mathscr{C}} \{\{x,x'\} \mid x,x' \in C \wedge x \neq x'\}$, $C_{sys} \in \mathscr{C}_{sys}$ and $C_{cor} \in \mathscr{C}_{cor}$.

One important question with regard to these evaluation measures is on which subset of the problem they are applied. It is understood from the above formula that there is a distinct precision and recall value for each clustering problem that is for each name. However, one could also consider the pairs to be taken over the entire test data, such that $pairs_{cor}(\mathscr{N}) = \bigcup_{name \in \mathscr{N}} \bigcup_{C \in \mathscr{C}_{cor}^{name}} \{\{x,x'\} \mid x,x' \in C \wedge x \neq x'\}$. In that case one would calculate one value of precision and recall over all correct and incorrect pairs in the test data. From these two values, F1 could be calculated. If we calculate precision and recall for each name separately, we have to average over all results (in doing so, each name is weighted equally, independent of the number of mentions

with that name). We can then calculate F1 from the average precision and average recall over all names. We use this approach to obtain a final score since we aim to establish a precise evaluation w.r.t. different cases of name ambiguity. In Table 25 we report the number of names that were found in the WoS data for each $|\mathscr{C}_{cor}|$, from which one can approximate the performance over the whole collection.

Experiments

In our experiments, we use the test environment and the evaluation measures described above in combination with different parameters, hyperparameters and variants. Our model has the following variants:

1. within vs. overall: $\#(f)$ only within *name* vs. over the entire collection
2. size vs. freq: $p(x|C) = \frac{[x \in C]}{|C|}$ vs. $p(x|C) = \frac{[x \in C] \cdot \#(x)}{\#(C)}$
3. prob vs. max: sum-of-products vs. maximum-of-products
4. pc_on vs. pc_off: $p(C) = \sum_x p(C|x) \cdot p(x)$ vs. $p(C) = 1$

Based on first experiments on the training data we choose the following setup : (1) overall, (2) freq, (3) prob, (4) pc_off. Our model has the following parameters and hyperparameters:

1. Smoothing hyperparameter ε
2. Feature-type weights λ
3. Threshold parameters α and β

So far, we have tried only one smoothing parameter $\varepsilon = .0001$. The effect of the smoothing parameter is yet to be studied. We train the feature-type weights over the union of training portions for all clustering sizes and approximate the results as shown in Table 26. The same table also shows all other feature-type weights

Table 26: Feature-type weights considered in our experiments.

	trained	*opposed*	*uniform*	*select*	*leaving-one-out*	*authors*	*docfeats*	
term	.15	.12	.125	1 0 0 0 0 0 0 0	0 .143 .143 .143 .143 .143 .143 .143	0 0 0 0 .5 .5 0 0	.2 0 0 .2 .2 .2 0 .2	*term*
aff	.2	.03	.125	0 0 0 0 0 0 0 0	.143 .143 .143 .143 .143 .143 .143 .143	0 0 0 0 .5 .5 0 0	.2 0 0 .2 .2 .2 0 .2	*aff*
cat	.18	.1	.125	0 0 0 0 0 0 0 0	.143 .143 .143 .143 .143 .143 .143 .143	0 0 0 0 .5 .5 0 0	.2 0 0 .2 .2 .2 0 .2	*cat*
key	.03	.2	.125	0 0 0 0 0 0 0 0	.143 .143 .143 .143 .143 .143 .143 .143	0 0 0 0 .5 .5 0 0	.2 0 0 .2 .2 .2 0 .2	*key*
co	.2	.02	.125	0 0 0 0 0 0 0 0	.143 .143 .143 .143 .143 .143 .143 .143	0 0 0 0 .5 .5 0 0	.2 0 0 .2 .2 .2 0 .2	*co*
ref	.12	.15	.125	0 0 0 0 0 0 0 0	.143 .143 .143 .143 .143 .143 .143 .143	0 0 0 0 .5 .5 0 0	.2 0 0 .2 .2 .2 0 .2	*ref*
email	.1	.18	.125	0 0 0 0 0 0 0 0	.143 .143 .143 .143 .143 .143 .143 .143	0 0 0 0 .5 .5 0 0	.2 0 0 .2 .2 .2 0 .2	*email*
year	.02	.2	.125	0 0 0 0 0 0 0 1	.143 .143 .143 .143 .143 .143 .143 0	0 0 0 0 .5 .5 0 0	.2 0 0 .2 .2 .2 0 .2	*year*

examined. We tuned the threshold parameters on the training portion and found that a good choice for the max variant is to set $\alpha = .0005, \beta = 0$ and for the prob variant results were best with $\alpha = 0, \beta = .000075$. Table 27 displays the results of the evaluation of our author name disambiguation approach on the base of the test corpus as regards recall, precision and F-measure. The evaluation shows high F-scores.

Main findings

In the following, we briefly summarise the main findings of our research: The best variant is using $p(C|C')$ with $p(x|C) = \frac{[x' \in C'] \cdot \#(x')}{\#(C')}$ and $\#(f)$ over the whole collection. When choosing the right variant and tuning an appropriate stopping criterion, our method is able to deliver results competitive with the state-of-the-art (cf. (M. Levin et al., 2012; F. H. Levin & Heuser, 2010; Khabsa et al., 2015; Kang et al., 2009; Qian et al., 2015)). However, more research is certainly needed here. In particular, the maximum-of-products is worth to be investigated more deeply. There are some hints that it might be even more precise ('max prec.' is slightly higher). As a next step, we intend to fully investigate potential gains of fine-tuning this variant. Even though feature weights learned in the classification scenario (logistic regression performed well for this task) are quite heterogeneous, in our model they do not perform better than uniformly distributed weights. We consider this as a benefit of our model, as the score works well independent from any particular training. We hypothesise that the probabilities filter out unspecific features, thereby implicitly controlling feature weighting without any discriminative training. The stopping criterion needs to be tuned on some training set, but it is only a single variable per variant that needs to be fitted – and there is considerable tolerance regarding its exact value. Thus, the quality threshold works remarkably well at finding an appropriate number of clusters.

Table 27: Recall and precision values of author name disambiguation on the test corpus, using bCube and pairF1 measure.

			1	2	3	4	5	6	7	8	9	10	$\lvert\mathscr{C}\rvert$				$\lvert\mathscr{C}\rvert$	1	2	3	4	5	6	7	8	9	10	
bCube	prob	$\alpha=0,\beta=.000075$	*100*	*95*	*95*	*96*	*97*	*96*	*94*	*96*	*95*	*94*	**Pr**	trained			**Pr**	*100*	*95*	*95*	*96*	*96*	*96*	*94*	*95*	*94*	*93*	pairF1
			98	***93***	***93***	***92***	***91***	***90***	***91***	***91***	***90***	***89***	**F1**				**F1**	***99***	***94***	***93***	***92***	***92***	***91***	***92***	***92***	***91***	*90*	
			97	*91*	*90*	*88*	*86*	*85*	*87*	*87*	*86*	*84*	**Re**				**Re**	*97*	*92*	*92*	*89*	*88*	*87*	*90*	*89*	*88*	*87*	
			100	*95*	*95*	*97*	*97*	*97*	*95*	*96*	*96*	*95*	**Pr**	uniform			**Pr**	*100*	*95*	*95*	*96*	*97*	*96*	*94*	*96*	*94*	*94*	
			98	***93***	***93***	***92***	*90*	***90***	*90*	*90*	*89*	***89***	**F1**				**F1**	***99***	***94***	***93***	***92***	*91*	*91*	*91*	*91*	*90*	***91***	
			97	*91*	*90*	*87*	*84*	*84*	*85*	*85*	*83*	*84*	**Re**				**Re**	*97*	*92*	*91*	*89*	*86*	*85*	*88*	*87*	*85*	*87*	
	max	$\alpha=.0005,\beta=0$	*100*	*96*	*96*	*97*	*96*	*95*	*92*	*92*	*91*	*88*	**Pr**	trained			**Pr**	*100*	*96*	*96*	*97*	*96*	*95*	*92*	*90*	*89*	*87*	
			96	*92*	*91*	*91*	*90*	***90***	*89*	*89*	*89*	*88*	**F1**				**F1**	*97*	*92*	*92*	*92*	*90*	*90*	*89*	*88*	*88*	*88*	
			93	*88*	*87*	*86*	*84*	*85*	*87*	*86*	*87*	*88*	**Re**				**Re**	*94*	*89*	*88*	*87*	*84*	*85*	*87*	*86*	*86*	*89*	
			100	*96*	*96*	*97*	*96*	*95*	*92*	*92*	*91*	*88*	**Pr**	uniform			**Pr**	*100*	*96*	*96*	*97*	*96*	*94*	*91*	*91*	*89*	*87*	
			96	*92*	*91*	*91*	*89*	***90***	*89*	*89*	*88*	*88*	**F1**				**F1**	*97*	*92*	*92*	*91*	*89*	*89*	*89*	*85*	*87*	*88*	
			92	*88*	*87*	*86*	*83*	*85*	*86*	*86*	*85*	*88*	**Re**				**Re**	*93*	*89*	*87*	*86*	*83*	*85*	*87*	*88*	*84*	*89*	

Another pleasant finding was that it is sufficient to tune one parameter depending on whether we use the sum-of-products or the maximum-of-products variant. In the first case, we tune β and in the second α. Also, the exact value of these parameters is not particularly important. It is sufficient to find the right order of magnitude. Deviations in the range of $\pm 50\%$ do not affect the performance much.

Leaving out one feature-type at a time in the clustering score shows that co-author names are the most important features as they are basically the only feature-type that cannot be taken out without deteriorating the performance. Disregarding this exception, our method's performance is not dependent on the presence of specific feature-types. For example, a weighting where only the co-author names and the referenced author names are used performs very well.

Recording the results for the system clustering sizes $|\mathscr{C}_{sys}|$ allows to precisely monitor the behaviour of the clustering process in order to tune precision and recall values for different problem cases. This might be particularly interesting for digital libraries, as they might want to focus on precision over recall. Separate evaluation for each correct clustering size $|\mathscr{C}_{cor}|$ also shows how high the baseline of putting all mentions in a single cluster is for the frequent cases of $|\mathscr{C}_{cor}| = 1$ or $|\mathscr{C}_{cor}| = 2$. If the larger problem cases are not considered separately, an approach could easily be considered satisfying even though it only approximates this primitive baseline.

The performance of our approach does not vary to any relevant extend between using pairF1 or bCube. Moreover, our evaluation features details not available in other publications and shows that our method can cope with all problem sizes. It also turned out that any performance figures reported in the literature have to be treated with caution if they disregard the baseline mentioned above.

Our research supports previous findings regarding the importance of co-authors and citation features (cp. (Kang et al., 2009; F. H. Levin & Heuser, 2010; G.-C. Li et al., 2014; Schulz et al., 2014)), but also suggests that other features like terms from the document should be used as well if available. In conclusion, we contribute a conceptually simple method, which has high F-score values independent of the exact set of feature-types used, without requiring any training phase and with only very little data, effort and accuracy required for tuning the stopping criterion.

3.7.4 Implementation, APIs and integration

In this section, we give a brief insight into efficient implementation of the approach for author disambiguation described on a more abstract level in the previous sections. In addition, we sketch how the method can be used as an integrated service in the context of a platform for literature search. So far, a fully operational prototype of author name disambiguation has been implemented and applied to the Web of Science corpus as described above. The next step will be to apply and evaluate the prototype w.r.t. MOVING use cases.

Implementation details

Our approach as presented in previous sections has been implemented in an efficient way by means of matrix multiplication. In a first step, we calculate $p(x|x')$ for all pairs $(x,x') \in X \times X$ and $p(x')$ for all $x' \in X$. This is based on the $|X| \times |F|$ count matrix $\#(x,f)$ (where $F = \bigcup_{x \in X} F(x)$) and the count vector $\#(f)$ which in the overall variant contains the feature counts over all names:

$$p(x|x') = \frac{\#(x,f) \cdot \#(x,f)^T + \frac{\varepsilon}{N}}{\#(x)^T + \varepsilon} \qquad \#(x) = \#(x,f) \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \qquad p(x') = \frac{\#(x) + \varepsilon}{\#(f) \cdot \langle 1 \dots 1 \rangle + N\varepsilon}$$

Here, $\cdot$ denotes matrix multiplication (matrix- or dot product). Next, during each iteration, we calculate the $|\mathscr{C}_{sys}| \times |\mathscr{C}_{sys}|$ matrix $p(C|C')$ from $p(x|x')$ and the current clustering $\mathscr{C}_{sys}$ represented as a $X \cdot |\mathscr{C}_{sys}|$ matrix:

$$p(x'|C') = \left((\#(x) + \varepsilon) \cdot \frac{1}{\#(C') + |\mathscr{C}_{sys}|\varepsilon} \right) \circ \mathscr{C}_{sys} \qquad \#(C) = \#(x)^T \cdot \mathscr{C}_{sys}$$

$$p(C|C') = p(C|x) \cdot \left(p(x|x') \cdot p(x'|C') \right) \qquad p(C|x) = \mathscr{C}_{sys}^T$$

Here, $\circ$ denotes component-wise multiplication (Hadamard product). This product ensures that $p(x'|C') = 0$ if $x' \notin C'$. For the max variant, all sums in the matrix products are replaced by a maximum function. As both the sum and the maximum take a set of input values and return a single output value, these can both be seen as special cases of the same generalisation.

Author name disambiguation as an integrated service

Our approach of author name disambiguation is designed to be run as an integrated service on the MOVING platform. As shown in Figure 17, the disambiguate service takes as input a (correctly normalised) $name \in \mathscr{N}$ and returns pairs (x,C) of mention IDs and author IDs. Note that the author ID is related to the name. The enrichment by disambiguation process is a parent process that takes a name and a mention ID from a metadata entry and completes it with the respective author ID returned by the disambiguation service. The process iterates over all normalised names and all author IDs for a given name in the collection and inserts the resulting author IDs into the respective mentions' metadata. Furthermore, the integration of the disambiguation service will be designed in such a way that it runs as a permanent background programme and continuously replies to name queries by the control process. In this way, it does not have to rebuild all necessary information in the memory just for a single request. Using a common module for normalisation should ensure that both the names in the collection metadata and the names in the author disambiguation database are normalised in the same way. This normalisation module might also implement a solution for the synonymy problem, i.e. that one author might be referred to by different name strings.

The MOVING platform will run an Elasticsearch index (ES) and an SQLite database for author disambiguation in parallel, which have to be correctly aligned. Our approach of representing, loading and using data of the MOVING collection is that we adapt a Python script that extracts the required information from the respective data source. The script creates a single SQLite file which indexes all the required information. For each dataset we derive another SQLite file from its SQLite representation which stores the author disambiguation features, in relation to the author mention (mentionID). This database is for disambiguation purposes only, but it is important that the names and mentionIDs are exactly the same as in the ES. In order to integrate the author disambiguation service, the following is necessary:

1. Each document in ES has a list of author-objects.
2. Each author object has definitely the following fields:
   (a) mentionID (docID+numberOfAuthorOnPaper).
   (b) authorID (disambiguated real-world author).
   (c) Name (normalised in the same way as with author disambiguation).

While the presented approach for authorship disambiguation has been evaluated on the Web of Science data (because this data comes with annotated author IDs), it will be integrated in the platform using the feature-types present in the different data sources that form the collection of the platform. In our case, this set of available feature-types corresponds to the basic feature-type weighting which was tested and which has shown satisfying results. Therefore, we are confident that the method can be applied successfully in this context.

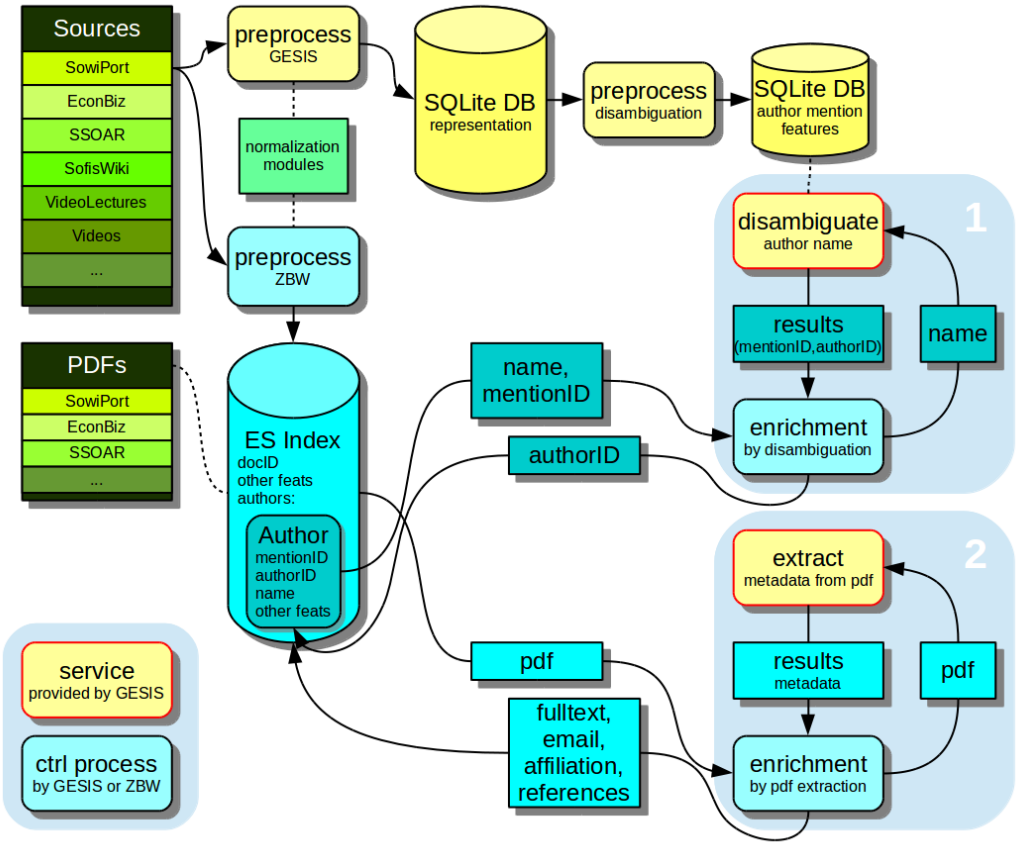


Figure 17: Integration of author disambiguation in the MOVING platform.

4 User logging and data analysis dashboard

4.1 Logging of user interaction data

4.1.1 Problem statement

We collect user interaction data with the MOVING platform in order to acquire insights about how the platform is being used. Thus, we are able to validate the requirements and to explore whether there is a relationship between interaction patterns and knowledge acquisition.

Usability studies in the laboratory, which are a well-established practice, have disadvantages in terms of data collection. While laboratory studies allow to control research variables do not account for unpredicted factors including the computer set up and the influence caused by the presence of an observer (i.e. the Guinea Pig effect). The users are aware they are being observed, therefore making them behave in a different way. Since users' day-to-day environments differ greatly from the ones found in laboratories the external validity of the outcomes is therefore a problem in these settings. The human-computer interaction community has acknowledged these problems and has started exploring *in-situ* observation techniques. These observations take place in users' own environments and since studies are run remotely more participants can get involved.

Remote unobtrusive observations tackle some of the above mentioned issues with laboratory studies. Having said this, the weakness of the laboratory studies is at the same time a strength in that being able to control variables prevents the uncertainty inherent in remote observations. Consequently a combination of remote observations with laboratory studies provides a solid approach to understanding how users employ the MOVING platform and identify possible issues while assuring that the platform's internal and external validity is assessed. For example, problems identified in remote observations could be recreated in laboratory studies in order to collect feedback from users.

Techniques to collect data from remote settings include Web logs, giving an unobtrusive view of users (Zaiane, Xin, & Han, 1998). The use of proxies to get the same information as Web server logs has been exploited to show click-through paths and to measure the time spent at each Web page (Cugini & Scholtz, 1999). There

have been visualisation approaches that give easily understandable insight into the users' behaviour performing a series of tasks (Carta, Paternò, & Santana, 2011). Other methods avoid predefining the interaction by extracting the tasks to be evaluated from data generated by users´ interacting freely (Vargas, Weffers, & da Rocha, 2010). Although Web logs can give interesting information like users' click-through, they make it impossible to analyse complex detailed interaction data from users.

An alternative to get interaction data unobtrusively is by instrumenting the Web application so particular events are recorded. Selected events to be recorded can be semantically relevant for the particular Web application instead of generic interaction events, simplifying the analysis. This approach requires an extensive modification of the Web application, hence being cumbersome and not easily scalable. Alternatively, additional code can be included on the Web pages that are being tracked. This approach has been commonly used in commercial approaches such as Google Analytics as well as in research into user behaviour (Cugini & Scholtz, 1999; Etgen & Cantor, 1999; Paganelli & Paternò, 2002; de Santana & Baranauskas, 2010; Atterer, Wnuk, & Schmidt, 2006). This approach removes the overhead of modifying the Web application, without requiring the installation of additional software on the client side, and can provide fine-grained interaction data.

4.1.2 Method description

UCIVIT[32] (Apaolaza, Harper, & Jay, 2013) has been chosen as a way to include additional code to unobtrusively capture interaction data from the MOVING platform. In order to collect such data it just requires adding a piece of JavaScript code to all the Web pages of the site resulting in a scalable solution. A wide range of low-level interaction events, such as mouse movement and window events, are captured for later analysis. Section "User interaction tracking" in Deliverable 4.1 provides detailed information about the capture. Permission from users is requested before any interaction capture takes place and users can opt out from it at anytime.

4.1.3 Experimental evaluation and comparison

UCIVIT has been successfully employed to capture fine-grained interaction data from real Web sites with high volume of visitors. UCIVIT has been deployed in the Web site of the School of computer science from the University of Manchester for over 16 months, capturing millions of events from over 18,000 recurrent users. Analysis of part of this data provided insight into evolving aspects of Web interaction (Apaolaza, Harper, & Jay, 2015).

4.1.4 Implementation, APIs and integration

The interaction capture platform has been deployed on the MOVING platform. The first time a user logs in, a permission prompt is shown requesting for their permission to collect data. If permission is granted, interaction data starts being captured and stored in a MongoDB database. All interaction is immediately available for processing and analysis, making the access to the interaction data near real-time.

4.2 Analysis of user interaction data

4.2.1 Problem statement

Analysis of the interaction of the MOVING platform serves various purposes. Firstly, it will support that the requirements of the use cases correspond to the actual usage of the platform. Secondly, it will provide detailed information of how particular users interact with the various features provided by MOVING. Finally, it will help designers to find interaction patterns, providing further understanding on the usage of the platform. Implemented analysis tools will enable designers/observers to understand how users employ the platform, highlighting issues and opportunities for improvements. Common metrics such as efficiency and time spent on the user interface can be extracted from the captured interaction data, and will feed the Adaptive Training Support system.

The analysis tool will also take advantage of the fine-granularity of the captured interaction events. Analysis of fine-grained interaction data has been found useful in the past to signal particular behaviours. For instance, mouse interaction has been successfully employed to identify scrolling and reading behaviours (Arroyo, Selker, & Wei, 2006). Combining mouse trajectory (including speed and acceleration) and past click-through information has also been found useful to predict future clicks (Guo, Agichtein, Clarke, & Ashkan, 2009). Special actions can be extracted from the interaction data, and used as indicators of usability problems. For example, undo and erase actions in 3D design tools have been successfully employed as a way to detect interaction problems

[32] `https://github.com/aapaolaza/UCIVIT-WebIntCap`, last accessed: 28/03/2017

that would otherwise remain unreported (Akers, Simpson, Jeffries, & Winograd, 2009). The isolation of these instances helps recreating them so that they can be repaired.

Other indicators can be more complex and entail looking for particular sequences of user interface events. For example, repeated actions on the interface can be used as indicators of users not being able to complete a task (W. Li, Harrold, & Görg, 2010). Known problematic behaviours can be programmatically coded as algorithms that keep track of particular sequences of user interface events (Vigo & Harper, 2017). When these behaviours are detected during the interaction, practical interventions can be put in place. Others have suggested to keep track of a set of behaviours over time in order to obtain insights into the evolution of users' behaviour (Apaolaza et al., 2015).

4.2.2 Method description

An interactive dashboard has been implemented to facilitate the analysis of interaction log data. WevQuery, which stands for Web Event Query tool, allows designers, who may not necessarily have database and programming skills, to build queries to retrieve information about behaviours exhibited on the Web. These queries are represented as sequences of events that are defined using interactive drag-and-drop functionalities on a Web application.

The analysis platform will include tools to extract patterns from interaction data. These patterns will support the validation of the requirements of the use cases of the MOVING platform. Additionally, they will also serve to create learning scenarios on the real use of the interface. These scenarios are workflows or sequences of events that indicate a learning activity.

A RESTful service will be implemented to allow other project partners access to the captured interaction data and patterns found. This service will support the Adaptive Training Support system, providing descriptive statistics about usage and information about learning activities.

4.2.3 Experimental evaluation and comparison

User studies will be carried out to measure the effectiveness and the perceived complexity of WevQuery. Continuous feedback from users of the RESTful service will help to improve the service and provide continuous support to their needs. In order to support the task of extracting interaction patterns from the use of the MOVING platform, appropriate algorithms for pattern matching need to selected. A performance comparison between the algorithms will be obtained by carrying out a benchmark with the following datasets:

- MSNBC.com anonymous Web dataset[33] containing 989,818 sequences of page visits. As described in the dataset: "the data comes from Internet Information Server (IIS) logs for msnbc.com and news-related portions of msn.com for the entire day of September, 28, 1999 (Pacific Standard Time). Each sequence in the dataset corresponds to page views of a user during that twenty-four hour period."[33]
- University of Manchester, School of Computer Science Sequences of page visits from December 2013 to March 2014, from 5,454 users, who created 30,023 sequences.
- Synthetic data generated by the IBM data generator. The IBM data generator, by IBM Research-Almaden[34], generates synthetic sequential data but is no longer officially supported. Data generated using this tool is still commonly employed and still available[35]. Two datasets containing 100,000 sequences of varying nature will be employed.
- Click-stream data of a Hungarian on-line news portal Dataset containing 88,162 sequences of page visits to a Hungarian on-line news portal[35].

4.2.4 Implementation, APIs and integration

WevQuery

WevQuery is a scalable system to query user interaction logs that allows designers to test their hypotheses about users' behaviour. WevQuery supports for this purpose a graphical notation to define the interaction event sequences and patterns. Designers can create complex queries adding temporal relations between user interface events in the Moving platform. This way WevQuery provides designers effortless access to users' interaction patterns, removing the complexity of low-level interaction data analysis.

[33] http://archive.ics.uci.edu/ml/datasets/MSNBC.com+Anonymous+Web+Data, last accessed: 27/03/2017
[34] http://www.research.ibm.com/labs/almaden/, last accessed: 30/03/2017
[35] http://fimi.ua.ac.be/data/, last accessed: 27/03/2017

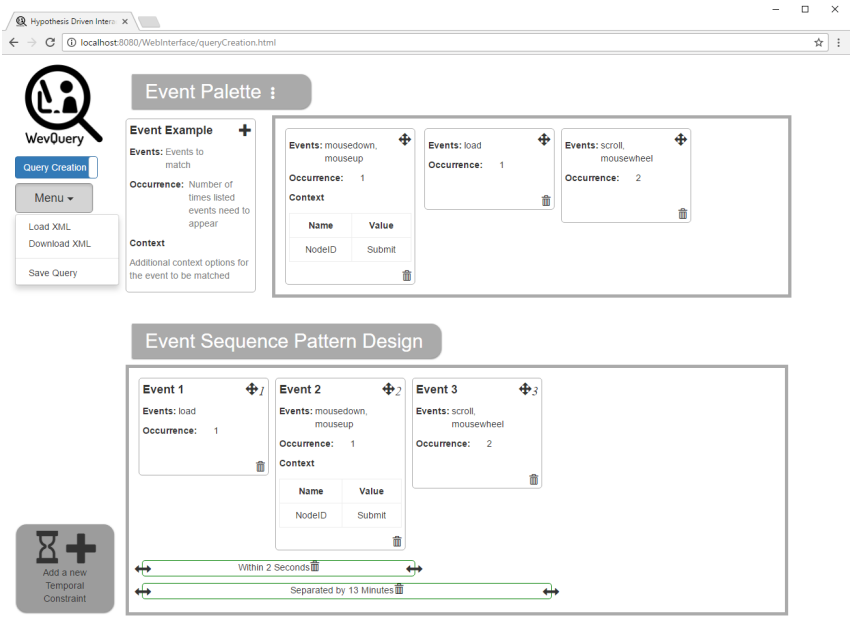


Figure 18: Screenshot of the query creation interface of the WevQuery web application.

WevQuery is comprised of two components: the first component shown in Figure 18 allows designers to formulate hypotheses of use by defining sequences of events and adding temporal restrictions between them. This way, the minimum, or maximum, time difference between the events in the sequence can be defined. The sequences serve as queries that are stored as XML files and can be saved, imported and exported. The second component depicted in Figure 19 allows designers to manage saved queries/hypotheses, and execute the queries against the captured interaction database. Designed queries in XML format are transformed into MapReduce (Dean & Ghemawat, 2008) queries for the underlying MongoDB system. The analysis panel of the interface provides insight into the results of these queries via interactive visualisations. In Figure 19, a Sankey diagram is shown, representing the volume of transitions between the events in a given hypothesis.

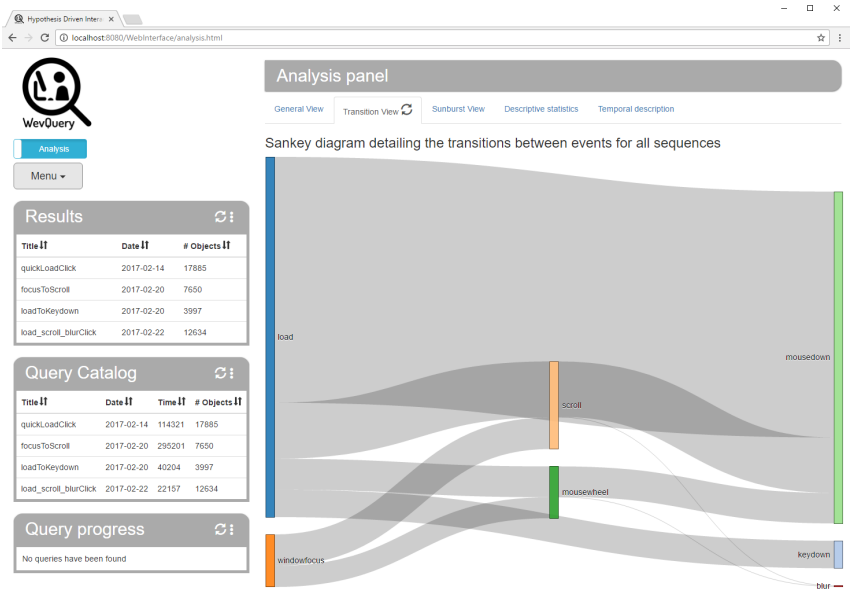


Figure 19: Screenshot of the query analysis interface of the WevQuery web application.

RESTful service
A RESTful service will provide access to user interaction data and it will serve as a layer of abstraction to be consumed by the Adaptive Training Support functionality. As such, its implementation will focus on supporting the necessary queries about users' interaction. For example, these queries could be whether particular elements of the user interface are being used or the duration of search tasks.

Pattern mining algorithms
The analysis dashboard will support pattern extraction functionalities. The objective is to support further extraction of patterns, combining queries created through WevQuery, to extract patterns that inform workflows and learning activities. The pattern extraction techniques we will explore for this analysis are:

- Frequent itemset mining involves finding itemsets where the support (frequency) is high enough. It can be used to detect events commonly found to take place together. For example, it could be found that users who click on X also access URL Y and scroll on Z. It can be useful to identify features commonly used together.
- Association rule mining is a method for discovering interesting relations between variables in large databases. An example of association rule is "A,B $\rightarrow$ C", meaning that C tends to occur when A and B occur. It first looks for frequent itemsets and then postprocess them into rules. These rules indicate relations between frequent itemsets. For example, it could be found that when users click on X and access Y, they then tend to scroll on Z. A measurement of a rule's confidence can be computed, indicating how often that rule has been found to be true.
- Sequential pattern mining looks for statistically relevant patterns where the values are delivered in a sequence. For example a frequent sequence can be users clicking on X, then clicking on Y, to then loading Z. It can be useful to identify common interaction sequences.
- Sequential rule mining takes into account the probability that a pattern will be followed. Mining sequential rules indicates the likelihood that a certain item will be found after a particular set of items is found. For example, it could be found that users tend to load Z after clicking on X, and then clicking on Y.

5 Visualisation technologies

5.1 Interactive network-visualisation framework

5.1.1 Problem statement

Users which use the Elasticsearch of the MOVING platform retrieve a list of search results. "Browsing through a long list of documents and then reading parts of the content to locate the needed information can be a mentally exhausting task" (Chau, 2011). Hence, data visualisation concepts can help in finding valuable information in search results easier as "the human visual system has enormous power to perceive information from visualized data" (Ware, 2012).

Graphs or node-link-diagrams are used to represent different entities and relations between them. Entities are visualised as nodes which are connected by links representing relations. Each graph is represented by a specific visual layout, which specifies the positions of the nodes (e.g. force-directed placement algorithms like Fruchterman and Reingold (Fruchterman & Reingold, 1991)) and the geometry of the links (such as edge bundling methods (Holten & van Wijk, 2009)). Different types of entities and relations as well as metadata can be visualised through different visual variables (see Section 5.2 for our proposed visual encoding concepts).

We contribute to the MOVING project by integrating our Graph Visualisation Framework (GVF[36]) into the MOVING web application, which is currently under development. GVF is a web-based framework designed to support interactive analysis of large, complex networks which may consist of documents, topical concepts, authors, venues, locations and other named entities as well as relationships which arise from co-occurrences, hierarchies, discourses, reading orders etc. To ensure scalability and provide smooth animated transitions, GVF is implemented using WebGL[37]-based rendering.

[36] `https://github.com/PeterHasitschka/gvf_core`, last accessed: 27/03/2017
[37] `https://www.khronos.org/registry/webgl/specs/latest/`, last accessed: 24/03/2017

Special focus is put on visual representation of metadata and novel graph aggregation metaphors (see (Kienreich, Wozelka, Sabol, & Seifert, 2012)) conveying relevant properties of nodes and relations in subgraphs. These metaphors are currently in the design phase, with draft designs introduced in the following subsections. Building on these visual metaphors, we plan to introduce powerful interaction models for explorative navigation (see (Rauch, Wozelka, Veas, & Sabol, 2014) for early ideas), filtering (such as in (Hasitschka & Sabol, 2017)) and visual querying of the graph data.

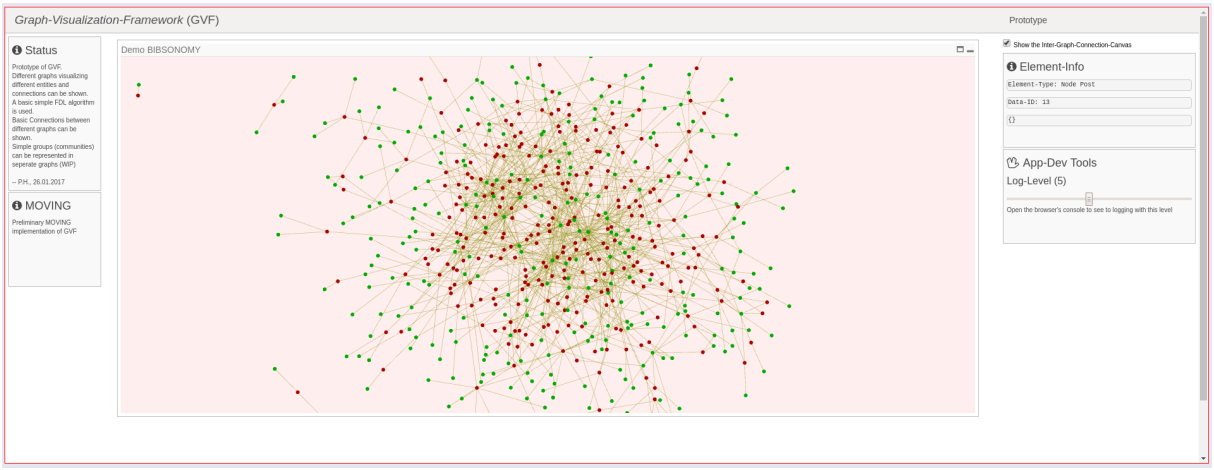


Figure 20: The graph visualisation framework (GVF): interactive visualisation of a network containing posts and tags.

5.1.2 Method description

Current state of the implementation

GVF runs as a web application inside all modern web browsers. It consists of a main window (see Figure 20), which can contain one or multiple graph visualisations simultaneously. Optional sidebars on the left and right side allow users to gain information on hovered nodes or to interact with the graphs. Graph windows can be freely resized. Zooming in and out is already supported. Panning is currently being implemented.

Figure 21 illustrates how GVF can be used in the current state of development. In the domain of analysing learning environments, it can be used to visually analyse different community types of learners. The visualisation is fully interactive and reacts on hovering over nodes or groups. In addition, it is possible to link connecting nodes in different windows is possible, e.g. to show connections between a person (which is shown in the learner graph, right) and the documents (shown in the resource graph, left) that this person has already accessed.

Visualised graphs may be very large. Combined with the fact that GVF can visualise multiple graphs at once and that it runs as a HTML5/JavaScript web application, the rendering performance needed particular attention. Thus, we decided to use a GPU-supported WebGL[37] rendering technology, allowing us to scale with the number of nodes and edges.

5.1.3 Implementation, API and integration

Programming language and libraries

GVF is written as an Angular2[38] web application using TypeScript[39] as programming language. This combination allows us to extend the framework and implement it in a fully object-oriented manner, which is typically limited when using common JavaScript. Node, edge and graph classes, as well as layout algorithms are implemented within a sophisticated class hierarchy which reduces redundant code and improves the maintainability of the application. To use it inside the MOVING project, the TypeScript code needs to be transpiled to pure JavaScript, since browsers cannot interpret it directly. During development, a built-in lightweight server is used to detect changes in the code and to transpile the files to a destination folder which is then used inside the MOVING platform. We use the Three.js[40] library, which provides functionality simplifying WebGL coding.

[38] `https://angularjs.org/`
[39] `https://www.typescriptlang.org/`, last accessed: 27/03/2017
[40] `https://threejs.org/`, last accessed: 27/03/2017

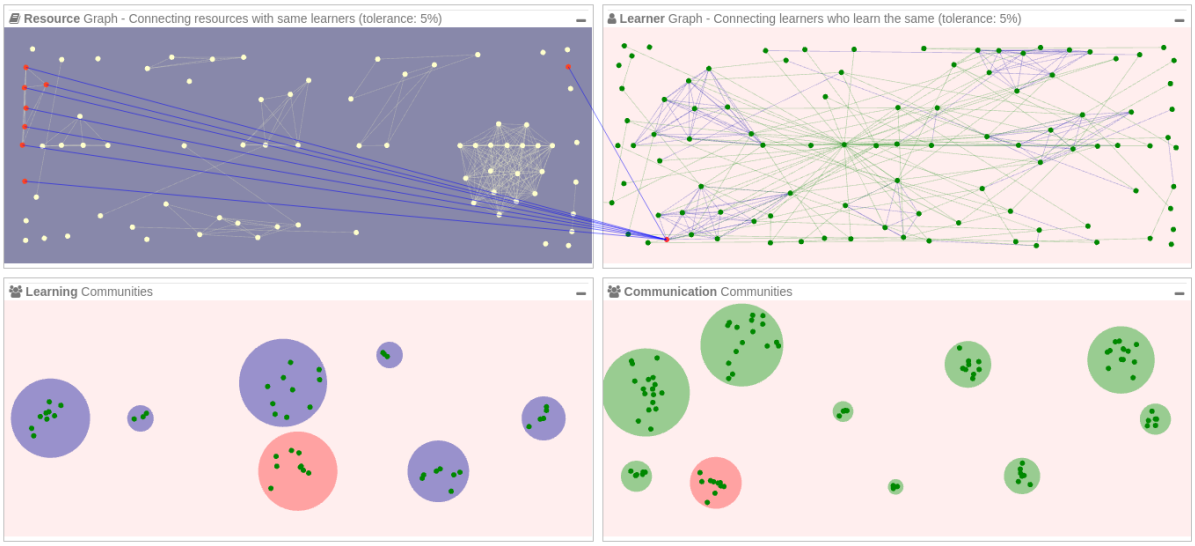


Figure 21: GVF used to visualise resources (up-left), persons (up-right) and extracted communities (based on similarity of learned resources on left and based on communication patterns on right) within a learning environment. Artificially created test data is used in this example.

Integration and API

GVF can be used either as standalone application or it can be integrated into other applications, which succeeds easily due to the provided integration API. The framework can be loaded inside an HTML-*iFrame*, which has the advantage that no conflicts between libraries used in the MOVING platform and those of GVF can occur. However, the communication between the two platforms becomes harder in this case, since the objects of GVF cannot be accessed through the *iFrame* and vice versa. For this purpose, we provide a simple API which is integrated into the MOVING platform. It sends and receives events to and from GVF facilitating the communication between the two applications.

5.2 Visual encoding of nodes and edges

5.2.1 Problem statement

The common data model (see Section 2) contains various data types, which we want to represent in our graph visualisation. Therefore it is necessary to define (a) which information should be shown as nodes, which as edges and what represents descriptive metadata, and (b) a visual encoding concept that ensures the user can perceive the data correctly. Please note that the proposed visual encodings and metaphors illustrate how these might look in the future GVF version. Their exact visual appearance is still a topic of discussion.

5.2.2 Method description

Definition of nodes and metadata properties

Table 28 shows relevant data listed in the current common data model, including a categorisation which of them should be represented as nodes, and which are metadata represented as visual properties of the nodes. We currently subdivide the metadata into categorical and non-categorical, depending on whether different category values can be distinguished visually using a limited number of icons or colours. Except for the dates (`startDate`, `endDate`), no numerical, continuous data is contained in the model at the moment.

Visual encoding of metadata in nodes

Considering the work of Carpendale (2003) regarding the usage of visual variables in information visualisation, we propose a visual encoding for nodes as listed in Table 29. The samples in the last column illustrate how we plan to use these visual variables in a default setting. We also plan to explore the possibility of letting users select how data properties are mapped onto visual properties.

Visual encoding of relations (edges)

The user does not only need to identify node types and their properties but also differentiate the relations

between them. Relations can be of different types or may carry metadata. Thus, we also need to visually encode relation information. We propose the following default settings:

- Thickness: Strength of a relation or the number of relations connecting two nodes.
- Texture: Uncertainty (e.g. dotted vs. full line)
- Colour: Relation type

Use of further visual variables like shape (straight, curved, angular) or opacity are still being discussed.

5.3 Navigation in large graphs

5.3.1 Problem statement

The graph we want to visualise contains multiple types of nodes connected by different types of links. Additionally, this graph might grow large and complex since a lot of hits may be received. The resulting visualisation is likely too cluttered due to many link crossings and overlapping nodes, leading to an overload of the user. We propose a concept that enables the user who focuses on particular information to visualise this information in detail. To avoid clutter, the rest of the graph is summarised in a way that allows users to identify and explore other potentially relevant graph areas.

To avoid information overload, the user shall initiate the exploration of the graph beginning from a small set of selected nodes, such as the most relevant results and the named entities extracted from these results.

Table 28: Default visual representation of the MOVING common data model.

Attribute	Shown as	Annotation
`document`	Node	
`author`	Node	
`affiliation`	Node	
`location`	Node	
`date`	Metadata	Timestamp
`venue`	Node	
`source`	Metadata	Categorical
`license`	Metadata	Categorical
`docType`	Metadata	Categorical
`language`	Metadata	Categorical
`concept`	Node	
`keyword`	Metadata	Categorical

Table 29: Proposed visual encoding of metadata in graph nodes and graph properties.

Graph property	Graph node	Visual Encoding	Mockup
Node-type	All	Icon plus Background-Colour	
Doc-type	Document	Sub-icon	
Language	Document+	Sub-icon (flag)	
Source	Document	Logo-icon	
License	Document	Pattern / texture	
StartDate, EndDate (distance from user-selected time interval)	Document, Venue	Opacity / intensity	

The user can explore the rest of the graph by clicking on nodes of interest, which triggers the expansion of the visible portion of the graph by showing nodes and relations which surround the current node(s) of interest.

Graph regions which are out of the user's focus are aggregated visually and represented by a less complex visual summary. This summary provides information on what the user can expect to find, if she or he decides to explore that particular area of the graph. However, identifying relevant graph areas and finding nodes containing interesting information might be hard due to a mass of nodes and edges which are characterised by rich metadata. Thus, our concept focuses on supporting explorative navigation of the graph, by providing means for interest-driven, selective expansion of the visible graph areas.

To summarise our goal is to empower the user to easily find relevant entities such as documents, authors, venues etc. and to discover relationships between them, by providing sophisticated visual aggregation metaphors. Based on these aggregation metaphors, we offer powerful interaction techniques for navigating and filtering the graph. Please note that the following visual concepts are in the early development stages.

5.3.2 Method description

Visualising a node's context

In our first scenario, we follow the goal of summarising a node's context - which is the network surrounding it - and supporting users in exploring the graph beginning from a particular node of interest. The concept is presented using the mockup shown in Figure 22.

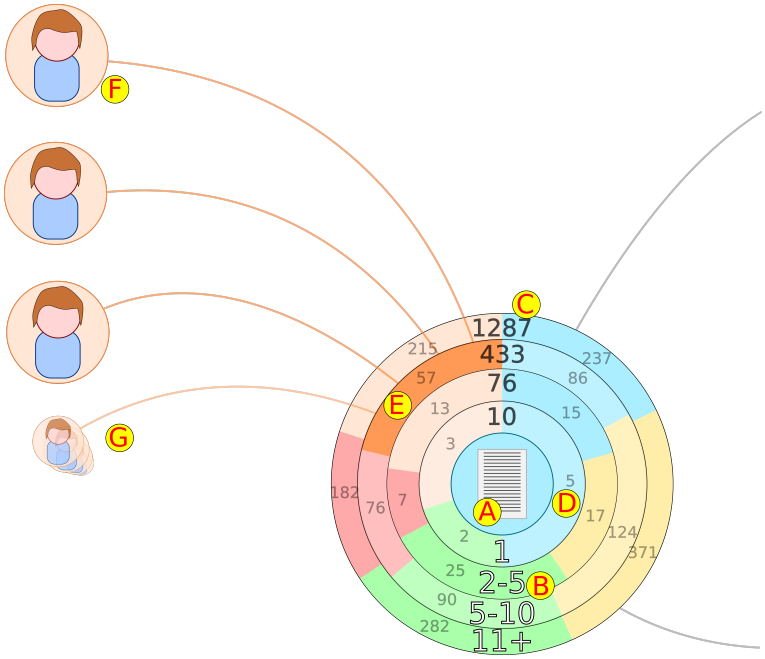


Figure 22: Concept of visualising the aggregated network around the focused node: (A): A document node is the current focus of interest of the user. Only a few other connections are visible (grey edges). (B): Concentric rings surround the node, each representing distance from the focus-node. For example the first ring (1) represents the immediate neighbours, while the second ring (2-5) summarises nodes which have a shortest path to the focus-node between two and five hops. The last ring represents additional, potentially relevant nodes in the graph, which are further away than 10 hops. (C): The number of nodes represented by each ring. (D): Each ring has segments. Each of them represents a different node type. They are colour encoded, thus the user can identify the type. For example the "5" in the first ring indicates that 5 documents (blue) are directly connected to the focus-node. "3" in the same rings indicates that three persons (beige) are mentioned in that document. (E): Interactive elements allow the user to navigate. Hovering over a segment (dark beige in the third ring) shows a handful nodes which correspond to the segment (in the sample: persons which can be reached by following five to ten hops). (F): The example shows that 57 persons meet the distance restrictions. Since showing all of them might overload the user, only the three most relevant are shown. This means, that a ranking depending on parameters like the distance (five might be more interesting than an author that is ten nodes away) or similarities between those nodes and, e.g. the currently focused one. (G): The user can also expand all the other authors which are collapsed in a further meta node.

The node itself is shown in the center while the visual neighbourhood summary appears when the user clicks on the node. The summary allows users to identify related nodes depending on their properties (such as the type or descriptive metadata) and their distance from the original node. By placing a particular focus

within of a large data set and then displaying what is surrounding the focused node, we follow the well-known focus-plus-context principle (Card, Mackinlay, & Shneiderman, 1999).

The caption of Figure 22 describes the idea in detail along a short example. To summarise, by combining the selective expand of the aggregated graph which surrounds the node (see *E* in Figure 22), and by showing only the most interesting nodes (*F*,*G*), calculated by a Degree-Of-Interest Function (van Ham & Perer, 2009), we provide interactive mechanisms for selective, context-based exploration of the graph starting from a particular user-selected node.

Visualising aggregated nodes and edges

In our second scenario the user may have already expanded a portion of the graph using the techniques described above. However, now he or she wishes to gain more understanding on what the whole graph is globally about. Areas of the graph, which are not closely related to the currently expanded subset may also provide valuable information which should not remain inaccessible to the user. Thus, we introduce a concept for aggregating major parts of a graph (e.g. using graph clustering algorithms) and visualising summaries of the computed aggregations. These visual summaries provide insights into what information is contained in the corresponding sub-graph. Additionally, the representation shall allow the user to easily identify sub-graphs, which share similar information.

Illustrated in Figure 23, we introduce a concept of visualising metadata distribution of aggregated graph regions (clusters) in a combination of multiple diagrams. The caption of the figure describes in detail the idea. To summarise, the resulting visual metaphor resembles a flower with leaves of different colours and length used to encode information. The visual summary not only shows the distribution of different node types (e.g. affiliations/countries, authors/persons etc.) and their instances (e.g. DE, EN, AT), but also allows comparison of sub-graphs by visually assessing the similarity of their leaf-structure.

We also plan to make this concept interactive by allowing the user a selective expand of nodes which meet particular criteria by clicking on corresponding leaves. Also, users could manually select and collapse large areas of the graph they are currently not focusing on to reduce the complexity of the entire graph visualisation.

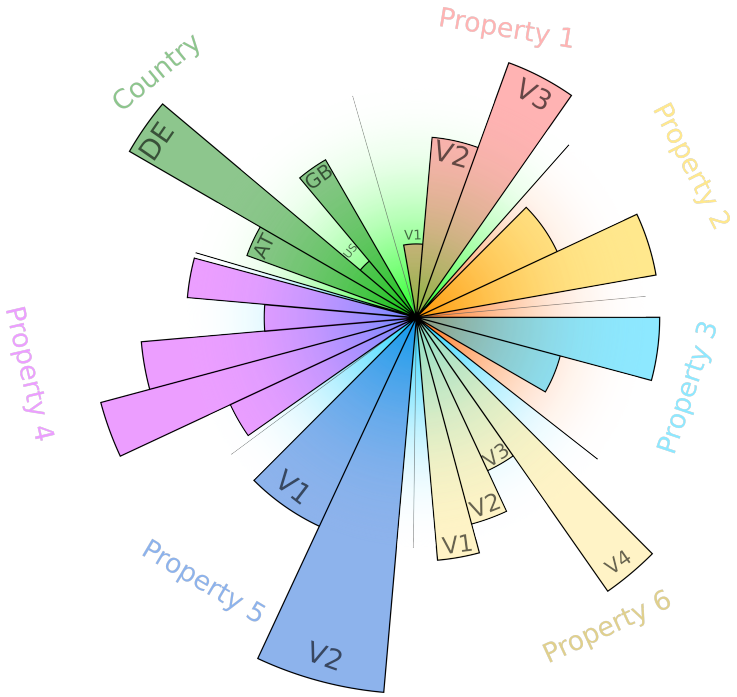


Figure 23: Concept of representing a summary of a graph cluster (a sub-graph): Different colours help to distinguish aggregated node types, for example blue for documents, green for affiliations/countries, orange for authors/persons etc. The distribution of particular instances for a type (e.g. DE, AT, GB, etc. for country) is shown as a radial bar chart. Bars are grouped into segments depending on node type, with the ratios between the segment areas (angles) representing the distribution of different node types. Using the radial bar charts to show the distribution of properties other then the node type will also be considered.

6 Conclusion

We presented our initial set of techniques for data acquisition, data processing, data visualisation and user logging. The common data model is able to represent full-texts, metadata, HTML content and video data in a unified way. This model will be refined after all functional requirements for the use cases (D1.1, M12) are known. Regarding data acquisition, we collect HTML content from three different sources using our three crawlers. The Social Stream Manager performs crawling in the Social Media, while the Search-Engine-based web Crawler exploits the Google search API capabilities to crawl the web for certain topics. We also employ the Focused web-Domain Crawler to crawl specific websites.

Regarding data processing, we investigated several techniques. For the ranking task, we showed that word embeddings can be successfully employed in a practical information retrieval setting. The cosine distance of aggregated IDF re-weighted word vectors is competitive to the TF-IDF baseline and even outperforms it on our news dataset with a relative percentage of 15%. The limitation of complete vocabulary mismatch (due to the necessary binary term-level matching operation even in case of word vectors) could be tackled by an embedding-based, semantic query expansion. For text extraction from scholarly figures, we systematically compared different configurations from the literature and showed that the best performing configuration is (BS15). This pipeline does not make any assumptions about the figures, e.g. figure type, font or colour. Thus performing better on heterogeneous datasets. As next step the extracted text needs to be integrated into a information retrieval engine to verify its practicability. Regarding the metadata extraction from PDF documents, we further investigate state-of-the-art tools and integrate an extraction framework into the MOVING project. The adaptive schema-level index for Linked Open Data allows capturing different representations of bibliographic metadata and allows to efficiently harvest additional metadata from the Linked Open Data cloud. However, this process needs to be optimised in terms of finding more data with less human effort to ease the integration process into MOVING. We employed text analysis techniques to the lecture videos' transcripts to extract concepts. We also performed visual analysis including temporal fragmentation and concept detection to non-lecture videos to semantically correlate them with the lecture videos. Furthermore, we developed a method for author disambiguation and alignment using agglomerative clustering, which is able to group mentions of the same name on different documents and to assign each author name a unique identifier that represents the real-world author. Despite the solid results in terms of precision and recall using the Web of Science dataset, the method still has to be evaluated within the MOVING platform to demonstrate its applicability for our use cases.

The UCIVIT framework is used to capture and log user interaction data on the MOVING platform. It adds JavaScript code that unobtrusively captures all interaction data in a scalable way, providing data from the actual use of the MOVING Web application. However, in a subsequent step, the extraction of interaction events from complex visualisations needs to addressed. The analysis of the captured user interactions serves various purposes. First, it will support that the predicted use cases correspond to the real usage of the platform. Second, it will provide detailed information of how particular users interact with the various features provided by MOVING. Finally, it will support designers to find interaction patterns, providing further understanding on the usage of the platform and helping check whether the requirements of use cases are satisfied. To allow access for the adaptive training support component to the dashboard, a RESTful API and pattern matching functionalities still need to be fully integrated.

Finally, we have introduced concepts and mockups to visualise and explore large graphs representing rich heterogeneous data. Using a combination of context summaries and node and edge aggregations, we are able to efficiently navigate through large graphs. The visualisations are implemented by extending our scalable Web-based Graph Visualisation Framework. The basic framework is set up and an initial test demonstrates its scalability.

References

Ahmed, K. M., & Al Dhubaib, B. (2011). Zotero: A bibliographic assistant to researcher. *Journal of Pharmacology and Pharmacotherapeutics*, *2*(4), 303. doi: 10.4103/0976-500x.85940

Akers, D., Simpson, M., Jeffries, R., & Winograd, T. (2009). Undo and erase events as indicators of usability problems. In *Proceedings of the 27th international conference on human factors in computing systems* (pp. 659–668). Boston, MA, USA: ACM. doi: 10.1145/1518701.1518804

Amati, G., & van Rijsbergen, C. J. (2002). Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, *20*(4), 357–389. doi: 10.1145/582415.582416

Apaolaza, A., Harper, S., & Jay, C. (2013). Understanding users in the wild. In *Proceedings of the 10th international cross-disciplinary conference on web accessibility* (pp. 13:1–13:4). Rio de Janeiro, Brazil: ACM. doi: 10.1145/2461121.2461133

Apaolaza, A., Harper, S., & Jay, C. (2015). Longitudinal analysis of low-level web interaction through micro behaviours. In *Proceedings of the 26th ACM conference on hypertext & social media* (pp. 337–340). Guzelyurt, Northern Cyprus: ACM. doi: 10.1145/2700171.2804453

Apostolidis, E., & Mezaris, V. (2014). Fast shot segmentation combining global and local visual descriptors. In *Proceedings of the 2014 IEEE international conference on acoustics, speech and signal processing* (pp. 6583–6587). Florence, Italy: IEEE. doi: 10.1109/ICASSP.2014.6854873

Arroyo, E., Selker, T., & Wei, W. (2006). Usability tool for analysis of web designs using mouse tracks. In *CHI '06 extended abstracts on human factors in computing systems* (pp. 484–489). Montréal, Québec, Canada: ACM. doi: 10.1145/1125451.1125557

Atterer, R., Wnuk, M., & Schmidt, A. (2006). Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In *Proceedings of the 15th international conference on world wide web* (pp. 203–212). Edinburgh, Scotlan: ACM. doi: 10.1145/1135777.1135811

Awad, G., Fiscus, J., Joy, D., Michel, M., Smeaton, A. F., Kraaij, W., … Larson, M. (2016). TRECVID 2016: Evaluating vdeo search, video event detection, localization, and hyperlinking. In *Proceedings of the 20th International Workshop on Video Retrieval Evaluation.* Gaithersburg, MD, USA: NIST. Retrieved from `http://www.eurecom.fr/en/publication/5054/download/data-publi-5054.pdf`

Balikas, G., & Amini, M. (2016). An empirical study on large scale text classification with skip-gram embeddings. *CoRR*, *abs/1606.06623*. Retrieved from `http://arxiv.org/abs/1606.06623`

Beeferman, D., Berger, A. L., & Lafferty, J. D. (1999). Statistical models for text segmentation. *Machine Learning*, *34*(1-3), 177–210. doi: 10.1023/A:1007506220214

Beek, W., Rietveld, L., Bazoobandi, H. R., Wielemaker, J., & Schlobach, S. (2014). LOD laundromat: A uniform way of publishing other people's dirty data. In *Proceedings of the 13th international semantic web conference* (Vol. 8796, pp. 213–228). Riva del Garda, Italy: Springer. doi: 10.1007/978-3-319-11964-9_14

Benedetti, F., Bergamaschi, S., & Po, L. (2015). Exposing the underlying schema of LOD sources. In *Proceedings of the international conference on web intelligence and intelligent agent technology* (pp. 301–304). Singapore. doi: 10.1109/WI-IAT.2015.99

Bengio, Y., Courville, A. C., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, *35*(8), 1798–1828. doi: 10.1109/TPAMI.2013.50

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*, 1137–1155. Retrieved from `http://www.jmlr.org/papers/v3/bengio03a.html`

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022. Retrieved from `http://www.jmlr.org/papers/v3/blei03a.html`

Böschen, F., & Scherp, A. (2015a). Formalization and preliminary evaluation of a pipeline for text extraction from infographics. In *Proceedings of the LWA 2015 workshops: KDML, FGWM, IR and FGDB* (Vol. 1458, pp. 20–31). Trier, Germany: CEUR-WS.org. Retrieved from `http://ceur-ws.org/Vol-1458/D03_CRC13_Boeschen.pdf`

Böschen, F., & Scherp, A. (2015b). Multi-oriented text extraction from information graphics. In *Proceedings of the 2015 ACM symposium on document engineering* (pp. 35–38). Lausanne, Switzerland: ACM. doi: 10.1145/2682571.2797092

Böschen, F., & Scherp, A. (2016). *A systematic comparison of different approaches for unsupervised extraction of text from scholarly figures [extended report]* (Tech. Rep. No. 1607). Christian-Albrechts-Universität zu Kiel. Retrieved from `http://www.uni-kiel.de/journals/receive/jportal_jparticle_00000290`

Böschen, F., & Scherp, A. (2017). A comparison of approaches for automated text extraction from scholarly figures. In *Proceedings of the 23nd international conference on multimedia modeling* (Vol. 10132, pp. 15–27). Reykjavik, Iceland: Springer. doi: 10.1007/978-3-319-51811-4_2

Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). *Readings in information visualization: using vision to think*. Morgan Kaufmann.

Carpendale, M. (2003). Considering visual variables as a basis for information visualisation. *Cartographica: International Journal for Geographic Information and Geovisualization*, *43*, 175–188. Retrieved from `http://prism.ucalgary.ca/bitstream/1880/45758/2/2001-693-16.pdf`

Carta, T., Paternò, F., & Santana, V. (2011). Support for remote usability evaluation of web mobile applications. In *Proceedings of the 29th ACM international conference on design of communication* (pp. 129–136). Pisa, Italy: AMC. doi: 10.1145/2038476.2038502

C.G.M., S., K.E.A., v. d. S., O., d. R., B., H., J.C., v. G., J.R.R., U., … D.C., K. (2008). The mediamill TRECVID 2008 semantic video search engine. In *Proceedings of the TRECVID workshop.* Gaithersburg, MD, USA: NIST. Retrieved from `http://www-nlpir.nist.gov/projects/tvpubs/tv8.papers/mediamill.pdf`

Chakrabarti, S., van den Berg, M., & Dom, B. (1999). Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, *31*(11–16), 1623–1640. doi: 10.1016/S1389-1286(99)00052-3

Chandrika Jayant, D. W., Matthew Renzelmann, Krisnandi, S., Ladner, R. E., & Comden, D. (2007). Automated tactile graphics translation: in the field. In *Proceedings of the 9th international ACM SIGACCESS conference on computers and accessibility* (pp. 75–82). Tempe, Arizona, USA: ACM. doi: 10.1145/1296843.1296858

Chau, M. (2011). Visualizing web search results using glyphs: Design and evaluation of a flower metaphor. *ACM Transactions on Management Information Systems*, *2*(1), 2. doi: 10.1145/1929916.1929918

Chiang, Y., & Knoblock, C. A. (2013). A general approach for extracting road vector data from raster maps. *IJDAR*, *16*(1), 55–81. doi: 10.1007/s10032-011-0177-1

Chiang, Y., & Knoblock, C. A. (2015). Recognizing text in raster maps. *GeoInformatica*, *19*(1), 1–27. doi: 10.1007/s10707-014-0203-9

Choudhury, S. R., & Giles, C. L. (2015). An architecture for information extraction from figures in digital libraries. In *Proceedings of the 24th international conference on world wide web companion* (pp. 667–672). Florence, Italy: ACM. doi: 10.1145/2740908.2741712

Ciglan, M., Nørvåg, K., & Hluchý, L. (2012). The semsets model for ad-hoc semantic list search. In *Proceedings of the 21st world wide web conference* (pp. 131–140). Lyon, France: ACM. doi: 10.1145/2187836.2187855

Clinchant, S., & Perronnin, F. (2013). Textual similarity with a bag-of-embedded-words model. In *Proceedings of the 2013 international conference on the theory of information retrieval* (p. 25). Copenhagen, Denmark: ACM. doi: 10.1145/2499178.2499180

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the twenty-fifth international conference on machine learning* (Vol. 307, pp. 160–167). Helsinki, Finland: ACM. doi: 10.1145/1390156.1390177

C. Tzelepis, V. M., D. Galanopoulos, & Patras, I. (2016). Learning to detect video events from zero or very few video examples. *Image and vision Computing*, *53*, 35–44. doi: 10.1016/j.imavis.2015.09.005

Cugini, J., & Scholtz, J. (1999). VISVIP: 3d visualization of paths through web sites. In *Proceedings. of the tenth international workshop on database and expert systems applications* (pp. 259 –263). Florence, Italy. doi: 10.1109/DEXA.1999.795175

Culotta, A., Kanani, P., Hall, R., Wick, M., & McCallum, A. (2007). Author disambiguation using error-driven machine learning with a ranking loss function. In *Proceedings of the sixth international workshop on information integration on the web.* Vancouver, Canada: AAAI. Retrieved from `http://www.aaai.org/Papers/Workshops/2007/WS-07-14/WS07-14-006.pdf`

De Giacomo, G., & Lenzerini, M. (1996). Tbox and abox reasoning in expressive description logics. In *Proceedings of the fifth international conference on principles of knowledge representation and reasoning* (pp. 316–327). Cambridge, Massachusetts, USA: Morgan Kaufmann. Retrieved from `http://www.aaai.org/Papers/Workshops/1996/WS-96-05/WS96-05-004.pdf`

Dean, J., & Ghemawat, S. (2008). Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, *51*(1), 107–113. doi: 10.1145/1327452.1327492

de Santana, V. F., & Baranauskas, M. C. C. (2010). Summarizing observational client-side data to reveal web usage patterns. In *Proceedings of the 2010 ACM symposium on applied computing* (pp. 1219–1223). Sierre, Switzerland: ACM. doi: 10.1145/1774088.1774344

Ding, L., Shinavier, J., Shangguan, Z., & McGuinness, D. L. (2010). Sameas networks and beyond: Analyzing

deployment status and implications of owl:sameas in linked data. In *Proceedings of the 9th international semantic web conference* (Vol. 6496, pp. 145–160). Shanghai, China: Springer. doi: 10.1007/978-3-642-17746-0_10

Duy-Dinh, L., Sang, P., Vinh-Tiep, N., Benjamin, R., Tuan A., N., Van-Nam, H., … Shin'ichi, S. (2016). NII-HITACHI-UIT at TRECVID 2016. In *TRECVID 2016 workshop.* Gaithersburg, MD, USA: NIST. Retrieved from `http://www-nlpir.nist.gov/projects/tvpubs/tv16.papers/nii-hitachi-uit.pdf`

Etgen, M., & Cantor, J. (1999). What does getting WET (web event-logging tool) mean for web usability. In *Proceedings of the fifth conference on human factors & the web.* Austin, Texas. Retrieved from `http://zing.ncsl.nist.gov/hfweb/proceedings/etgen-cantor/index.html`

European Mathematical Society. (2014). *Equivalence relation.* Springer. Retrieved from `http://www.encyclopediaofmath.org/index.php?title=Equivalence_relation&oldid=35990`

Fagan, J. L. (1987). Automatic phrase indexing for document retrieval: An examination of syntactic and non-syntactic methods. In *Proceedings of the tenth annual international ACM SIGIR conference on research and development in information retrieval* (pp. 91–101). New Orleans, Louisiana, USA: ACM. doi: 10.1145/42005.42016

Ferreira, A. A., Gonçalves, M. A., & Laender, A. H. (2012). A brief survey of automatic methods for author name disambiguation. *ACM Sigmod Record*, *41*(2), 15–26. doi: 10.1145/2350036.2350040

Ferreira, A. A., Veloso, A., Gonçalves, M. A., & Laender, A. H. (2010). Effective self-training author name disambiguation in scholarly digital libraries. In *Proceedings of the 10th annual joint conference on digital libraries* (pp. 39–48). Gold Coast, Queensland, Australia: ACM. doi: 10.1145/1816123.1816130

Fruchterman, T. M., & Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and experience*, *21*(11), 1129–1164. doi: 10.1002/spe.4380211102

Furnas, G. W., Deerwester, S. C., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L. A., & Lochbaum, K. E. (1988). Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 465–480). Grenoble, France: ACM. doi: 10.1145/62437.62487

Gabrilovich, E., & Markovitch, S. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on artifical intelligence* (Vol. 7, pp. 1606–1611). Hyderabad, India: Morgan Kaufmann. Retrieved from `http://www.aaai.org/Papers/IJCAI/2007/IJCAI07-259.pdf`

Garofalakis, M. N., Gehrke, J., & Rastogi, R. (Eds.). (2016). *Data stream management - processing high-speed data streams*. Springer. doi: 10.1007/978-3-540-28608-0

Goldman, R., & Widom, J. (1997). Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proceedings of 23rd international conference on very large data bases* (p. 436-445). Athens, Greece: Morgan Kaufmann. Retrieved from `http://www.vldb.org/conf/1997/P436.PDF`

Goth, G. (2016). Deep or shallow, NLP is breaking out. *Commun. ACM*, *59*(3), 13–16. doi: 10.1145/2874915

Gottron, T., Scherp, A., Krayer, B., & Peters, A. (2013). LODatio: using a schema-level index to support users infinding relevant sources of linked data. In *Proceedings of the 7th international conference on knowledge capture* (pp. 105–108). Banff, Canada: ACM. doi: 10.1145/2479832.2479841

Guo, Q., Agichtein, E., Clarke, C. L. A., & Ashkan, A. (2009). In the mood to click? towards inferring receptiveness to search advertising. In *Proceedings of the 2009 IEEE/WIC/ACM international joint conference on web intelligence and intelligent agent technology* (pp. 319–324). Washington, DC, USA: IEEE Computer Society. doi: 10.1109/WI-IAT.2009.368

Gurney, T., Horlings, E., & Van Den Besselaar, P. (2012). Author disambiguation using multi-aspect similarity indicators. *Scientometrics*, *91*(2), 435–449. doi: 10.1007/s11192-011-0589-1

Han, H., Xu, W., Zha, H., & Giles, C. L. (2005). A hierarchical naive bayes mixture model for name disambiguation in author citations. In *Proceedings of the 2005 ACM symposium on applied computing* (pp. 1065–1069). ACM. doi: 10.1145/1066677.1066920

Harzing, A.-W. (2015). Health warning: might contain multiple personalities - the problem of homonyms in thomson reuters essential science indicators. *Scientometrics*, *105*(3), 2259–2270. doi: 10.1007/s11192-015-1699-y

Hasitschka, P., & Sabol, V. (2017). Visual exploration and analysis of recommender histories: A web-based approach using webgl. In *Proceedings of the 2017 ACM workshop on exploratory search and interactive data analytics* (pp. 33–40). Limassol, Cyprus: ACM. doi: 10.1145/3038462.3038467

Hiemstra, D. (1998). A linguistically motivated probabilistic model of information retrieval. In *Proceedings of the european conference on research and advanced technology on digital libraries* (Vol. 1513, pp.

569–584). Heraklion, Crete, Greece: Springer. doi: 10.1007/3-540-49653-X_34

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 50–57). Berkeley, CA, USA: ACM. doi: 10.1145/312624.312649

Holten, D., & van Wijk, J. J. (2009). Force-directed edge bundling for graph visualization. In *Proceedings of the 11th eurographics/IEEE-VGTC conference on visualization* (pp. 983–998). Berlin, Germany: The Eurographs Association; John Wiley; Sons, Ltd. doi: 10.1111/j.1467-8659.2009.01450.x

Isele, R., Umbrich, J., Bizer, C., & Harth, A. (2010). LDspider: An open-source crawling framework for the web of linked data. In *Proceedings of the ISWC 2010 posters & demonstrations track: Collected abstracts* (Vol. 658). Shanghai, China: CEUR-WS.org. Retrieved from `http://ceur-ws.org/Vol-658/paper495.pdf`

Jiuzhou, Z. (2006). *Creation of synthetic chart image database with ground truth* (Honors Year Project Report). National University of Singapore. Retrieved from `https://www.comp.nus.edu.sg/~tancl/ChartImageDatabase/Report_Zhaojiuzhou.pdf`

Juan, C., Yong-Dong, Z., Bai-Lan, F., Xiu-Feng, H., Lei, B., Xu, Z., & Jin-Tao, L. (2008). TRECVID 2008 search task by MCG-ICT-CAS. In *Proceedings of the TRECVID workshop*. Gaithersburg, MD, USA: NIST. Retrieved from `http://www-nlpir.nist.gov/projects/tvpubs/tv8.papers/mcg-ict-cas.pdf`

Junwei, L., Jia, C., Poyao, H., Xuanchong, L., Lu, J., Zhenzhong, L., … Alexandar, H. (2016). Informedia @ Trecvid 2016. In *TRECVID 2016 workshop*. Gaithersburg, MD, USA: NIST. Retrieved from `http://www-nlpir.nist.gov/projects/tvpubs/tv16.papers/inf.pdf`

Käfer, T., Abdelrahman, A., Umbrich, J., O'Byrne, P., & Hogan, A. (2013). Observing linked data dynamics. In *The semantic web: Semantics and big data, 10th international conference, ESWC 2013* (Vol. 7882, pp. 213–227). Montpellier, France: Springer. doi: 10.1007/978-3-642-38288-8_15

Kang, I.-S., Na, S.-H., Lee, S., Jung, H., Kim, P., Sung, W.-K., & Lee, J.-H. (2009). On co-authorship for author disambiguation. *Information Processing & Management*, *45*(1), 84–97. doi: 10.1016/j.ipm.2008.06.006

Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S. K., Bagdanov, A. D., Iwamura, M., … Valveny, E. (2015). ICDAR 2015 competition on robust reading. In *Proceedings of the 13th international conference on document analysis and recognition* (pp. 1156–1160). Nancy, France: IEEE Computer Society. doi: 10.1109/ICDAR.2015.7333942

Khabsa, M., Treeratpituk, P., & Giles, C. L. (2015). Online person name disambiguation with constraints. In *Proceedings of the 15th ACM/IEEE-CS joint conference on digital libraries* (pp. 37–46). Knoxville, Tennessee, USA: ACM. doi: 10.1145/2756406.2756915

Khurshid, K., Siddiqi, I., Faure, C., & Vincent, N. (2009). Comparison of niblack inspired binarization methods for ancient documents. In *Document recognition and retrieval xvi* (Vol. 7247, pp. 1–10). San Jose, CA, USA: SPIE. doi: 10.1117/12.805827

Kienreich, W., Wozelka, R., Sabol, V., & Seifert, C. (2012). Graph visualization using hierarchical edge routing and bundling. In *Proceedings of the 3rd international eurovis workshop on visual analytics* (pp. 97–101). Vienna, Austria: The Eurographics Association. doi: 10.2312/PE/EuroVAST/EuroVA12/097-101

Konrath, M., Gottron, T., Staab, S., & Scherp, A. (2012). SchemEX - efficient construction of a data catalogue by stream-based indexing of linked data. *J. Web Sem.*, *16*, 52–58. doi: 10.1016/j.websem.2012.06.002

Krämer, T., Momeni, F., & Mayr, P. (2017). Coverage of author identifiers in web of science and scopus. *ArXiv e-prints*. Retrieved from `https://arxiv.org/abs/1703.01319`

Kusner, M. J., Sun, Y., Kolkin, N. I., & Weinberger, K. Q. (2015). From word embeddings to document distances. In *Proceedings of the 32nd international conference on machine learning* (Vol. 37, pp. 957–966). Lille, France: JMLR.org. Retrieved from `http://jmlr.org/proceedings/papers/v37/kusnerb15.html`

Le, Q. V., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31th international conference on machine learning* (Vol. 32, pp. 1188–1196). Beijing, China: JMLR.org. Retrieved from `http://jmlr.org/proceedings/papers/v32/le14.html`

Levin, F. H., & Heuser, C. A. (2010). Evaluating the use of social networks in author name disambiguation in digital libraries. *Journal of Information and Data Management*, *1*(2), 183. Retrieved from `https://seer.ufmg.br/index.php/jidm/article/viewFile/35/23`

Levin, M., Krawczyk, S., Bethard, S., & Jurafsky, D. (2012). Citation-based bootstrapping for large-scale author disambiguation. *Journal of the American Society for Information Science and Technology*, *63*(5), 1030–1047. doi: 10.1002/asi.22621

Li, G.-C., Lai, R., D'Amour, A., Doolin, D. M., Sun, Y., Torvik, V. I., … Fleming, L. (2014). Disambiguation

and co-authorship networks of the us patent inventor database (1975–2010). *Research Policy*, *43*(6), 941–955. doi: 10.1016/j.respol.2014.01.012

Li, W., Harrold, M. J., & Görg, C. (2010). Detecting user-visible failures in AJAX web applications by analyzing users' interaction behaviors. In *Proceedings of the IEEE/ACM international conference on automated software engineering* (pp. 155–158). Antwerp, Belgium: ACM. doi: 10.1145/1858996.1859025

Lipinski, M., Yao, K., Breitinger, C., Beel, J., & Gipp, B. (2013). Evaluation of header metadata extraction approaches and tools for scientific pdf documents. In *Proceedings of the 13th ACM/IEEE-CS joint conference on digital libraries* (pp. 385–386). Indianapolis, Indiana, USA: ACM. doi: 10.1145/2467696.2467753

Lopez, P. (2009). GROBID: Combining automatic bibliographic data recognition and term extraction for scholarship publications. In *Proceedings of the international conference on theory and practice of digital libraries* (pp. 473–474). Corfu, Greece: Springer. doi: 10.1007/978-3-642-04346-8_62

Lu, X., Kataria, S., Brouwer, W. J., Wang, J. Z., Mitra, P., & Giles, C. L. (2009). Automated analysis of images in documents for intelligent document search. *IJDAR*, *12*(2), 65–81. doi: 10.1007/s10032-009-0081-0

Luo, Y., Fletcher, G. H. L., Hidders, J., Wu, Y., & Bra, P. D. (2013). External memory k-bisimulation reduction of big graphs. In *Proceedings of the 22nd ACM conference on information and knowledge management* (pp. 919–928). San Francisco, CA, USA. Retrieved from `https://arxiv.org/pdf/1210.0748`

Markatopoulou, F., Moumtzidou, A., Galanopoulos, D., Theodoros, M., Vagia, K., Anastasia, I., ... Ioannis, P. (2016). ITI-CERTH participation to TRECVID 2016. In *Proceedings of the TRECVID workshop*. Gaithersburg, MD, USA: NIST. Retrieved from `http://www-nlpir.nist.gov/projects/tvpubs/tv16.papers/iti-certh.pdf`

McHugh, J., Abiteboul, S., Goldman, R., Quass, D., & Widom, J. (1997). Lore: a database management system for semistructured data. *SIGMOD Record*, *26*(3), 54–66. Retrieved from `http://ilpubs.stanford.edu:8090/261/1/1997-48.pdf`

Mei, T., Zha, Z.-J., Liu, Y., Wang, M., Qi, G.-J., Tian, X., ... Hua, X.-S. (2008). MSRA at TRECVID 2008: High-level feature extraction and automatic search. In *Proceedings of the TRECVID workshop*. Gaithersburg, MD, USA: NIST. Retrieved from `http://www-nlpir.nist.gov/projects/tvpubs/tv8.papers/msra.pdf`

Menczer, F. (1997). ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery. In *Proceedings of the 14th international conference on machine learning* (pp. 227–235). San Francisco, CA, USA. Retrieved from `http://carl.cs.indiana.edu/fil/Papers/ICML.ps`

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th international conference on neural information processing systems* (pp. 3111–3119). Lake Tahoe, Nevada, USA. Retrieved from `http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality`

Miller, D. R., Leek, T., & Schwartz, R. M. (1999). A hidden markov model information retrieval system. In *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 214–221). Berkeley, California, USA: ACM. doi: 10.1145/312624.312680

Milojević, S. (2013). Accuracy of simple, initials-based methods for author name disambiguation. *Journal of Informetrics*, *7*(4), 767–773. Retrieved from `https://arxiv.org/pdf/1308.0749`

Mnih, A., Yuecheng, Z., & Hinton, G. E. (2009). Improving a statistical language model through non-linear prediction. *Neurocomputing*, *72*(7-9), 1414–1418. doi: 10.1016/j.neucom.2008.12.025

Mu, J., Bhat, S., & Viswanath, P. (2017). All-but-the-top: Simple and effective postprocessing for word representations. *CoRR*, *abs/1702.01417*. Retrieved from `http://arxiv.org/abs/1702.01417`

Muhammad Fraz, M. S. S., & Edirisinghe, E. A. (2015). Exploiting colour information for better scene text detection and recognition. *IJDAR*, *18*(2), 153–167. doi: 10.1007/s10032-015-0239-x

Neumann, T., & Moerkotte, G. (2011). Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In *Proceedings of the 27th international conference on data engineering* (p. 984-994). Hannover, Germany: IEEE Computer Society. doi: 10.1109/ICDE.2011.5767868

Ngo, C.-W., Jiang, Y.-G., Wei, X.-Y., Zhao, W., Wang, F., Wu, X., & Tan, H.-K. (2008). Beyond semantic search: What you observe may not be what you think. In *Proceedings of the TRECVID workshop*. Gaithersburg, MD, USA: NIST. Retrieved from `http://www-nlpir.nist.gov/projects/tvpubs/tv8.papers/vireo_cityuhk.pdf`

Norouzi, M., Mikolov, T., & S. Bengio, e. a. (2013). Zero-shot learning by convex combination of semantic embeddings. *CoRR*, *abs/1312.5650*. Retrieved from `https://arxiv.org/pdf/1312.5650`

Otsu, N. (1979). A threshold selection method from gray-level histograms. *Systems, Man and Cybernetics, IEEE Transactions on*, *9*(1), 62–66. doi: 10.1109/TSMC.1979.4310076

Over, P., Awad, G., Rose, T., Fiscus, J., Kraaij, W., & Smeaton-Alan, A. (2008). TRECVID 2008–Goals, Tasks, Data, Evaluation Mechanisms and Metrics. In *Trecvid 2008 workshop*. Gaithersburg, MD, USA: NIST. Retrieved from `http://www-nlpir.nist.gov/projects/tvpubs/tv8.papers/tv8overview.pdf`

Paganelli, L., & Paternò, F. (2002). Intelligent analysis of user interactions with web applications. In *Proceedings of the 7th international conference on intelligent user interfaces* (pp. 111–118). San Francisco, CA, USA: ACM. doi: 10.1145/502716.502735

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing* (pp. 1532–1543). Doha, Qatar: ACL. Retrieved from `http://aclweb.org/anthology/D/D14/D14-1162.pdf`

Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 275–281). Melbourne, Australia: ACM. doi: 10.1145/290941.291008

Qian, Y., Hu, Y., Cui, J., Zheng, Q., & Nie, Z. (2011). Combining machine learning and human judgment in author disambiguation. In *Proceedings of the 20th acm international conference on information and knowledge management* (pp. 1241–1246). Glasgow, Scotland, UK: ACM. doi: 10.1145/2063576.2063756

Qian, Y., Zheng, Q., Sakai, T., Ye, J., & Liu, J. (2015). Dynamic author name disambiguation for growing digital libraries. *Information Retrieval Journal*, *18*(5), 379–412. doi: 10.1007/s10791-015-9261-3

Rauch, M., Wozelka, R., Veas, E., & Sabol, V. (2014). Semantic blossom graph: A new approach for visual graph exploration. In (p. 234-240). Los Alamitos, CA, USA: IEEE Computer Society. doi: 10.1109/IV.2014.36

Robertson, S. E., Walker, S., Hancock-Beaulieu, M., Gatford, M., & Payne, A. (1995). Okapi at TREC-4. In *Proceedings of the fourth text retrieval conference* (Vol. Special Publication 500-236). Gaithersburg, Maryland, USA: NIST. Retrieved from `http://trec.nist.gov/pubs/trec4/papers/city.ps.gz`

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the 2011 international conference on computer vision* (pp. 2564–2571). Barcelona, Spain: IEEE. doi: 10.1109/ICCV.2011.6126544

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, *24*(5), 513–523. doi: 10.1016/0306-4573(88)90021-0

Salton, G., Fox, E. A., & Wu, H. (1983). Extended boolean information retrieval. *Commun. ACM*, *26*(11), 1022–1036. doi: 10.1145/182.358466

Samet, H., & Tamminen, M. (1988). Efficient component labeling of images of arbitrary dimension represented by linear bintrees. *IEEE TPAMI*, *10*(4), 579–586. doi: 10.1109/34.3918

Sandra Carberry, S. E., & Demir, S. (2006). Information graphics: an untapped resource for digital libraries. In *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 581–588). Seattle, Washington, USA: ACM. doi: 10.1145/1148170.1148270

Sas, J., & Zolnierek, A. (2013). Three-stage method of text region extraction from diagram raster images. In *Proceedings of the 8th international conference on computer recognition systems* (Vol. 226, pp. 527–538). Milkow, Poland: Springer. doi: 10.1007/978-3-319-00969-8_52

Savva, M., Kong, N., Chhajta, A., Li, F., Agrawala, M., & Heer, J. (2011). Revision: automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on user interface software and technology* (pp. 393–402). Santa Barbara, CA, USA: ACM. doi: 10.1145/2047196.2047247

Schaible, J., Gottron, T., & Scherp, A. (2016). TermPicker: Enabling the reuse of vocabulary terms by exploiting data from the linked open data cloud. In *Proceedings of the 13th european semantic web conference* (pp. 101–117). Heraklion, Crete, Greece. doi: 10.1007/978-3-319-34129-3_7

Schulz, C., Mazloumian, A., Petersen, A. M., Penner, O., & Helbing, D. (2014). Exploiting citation networks for large-scale author name disambiguation. *EPJ Data Science*, *3*(1), 11. doi: 10.1140/epjds/s13688-014-0011-3

Seo, K.-d., Park, S. J., & Jung, S.-h. (2009). Wipe scene-change detector based on visual rhythm spectrum. *IEEE Transactions on Consumer Electronics*, *55*(2). doi: 10.1109/TCE.2009.5174462

Smalheiser, N. R., & Torvik, V. I. (2009). Author name disambiguation. *Annual review of information science and technology*, *43*(1), 1–43. doi: 10.1002/aris.2009.1440430113

Song, Y., Huang, J., Councill, I. G., Li, J., & Giles, C. L. (2007). Efficient topic-based unsupervised name disambiguation. In *Proceedings of the 7th ACM/IEEE-CS joint conference on digital libraries* (pp. 342–351). ACM. doi: 10.1145/1255175.1255243

Spahiu, B., Porrini, R., Palmonari, M., Rula, A., & Maurino, A. (2016). ABSTAT: ontology-driven linked data summaries with pattern minimalization. In *The semantic web - ESWC 2016 satellite events, revised*

selected papers (pp. 381–395). Heraklion, Crete, Greece. doi: 10.1007/978-3-319-47602-5_51

Strotmann, A., & Zhao, D. (2012). Author name disambiguation: What difference does it make in author-based citation analysis? *Journal of the American Society for Information Science and Technology*, *63*(9), 1820–1833. doi: 10.1002/asi.22695

Su, C.-W., Liao, H.-Y., Tyan, H.-R., Fan, K.-C., & Chen, L.-H. (2005). A motion-tolerant dissolve detection algorithm. *IEEE transactions on multimedia*, *7*(6), 1106–1113. doi: 10.1109/ICME.2002.1035555

Tan, Y. F., Kan, M. Y., & Lee, D. (2006). Search engine driven author disambiguation. In *Proceedings of the 6th ACM/IEEE-CS joint conference on digital libraries* (pp. 314–315). Chapel Hill, NC, USA: ACM. doi: 10.1145/1141753.1141826

Tang, J., Fong, A. C., Wang, B., & Zhang, J. (2012). A unified probabilistic framework for name disambiguation in digital library. *IEEE Transactions on Knowledge and Data Engineering*, *24*(6), 975–987. doi: 10.1109/TKDE.2011.13

Tang, L., & Walsh, J. P. (2010). Bibliometric fingerprints: name disambiguation based on approximate structure equivalence of cognitive maps. *Scientometrics*, *84*(3), 763–784. doi: 10.1007/s11192-010-0196-6

Tkaczyk, D., Szostek, P., Fedoryszak, M., Dendek, P. J., & Bolikowski, Ł. (2015). CERMINE: automatic extraction of structured metadata from scientific literature. *International Journal on Document Analysis and Recognition*, *18*(4), 317–335. doi: 10.1007/s10032-015-0249-8

Torvik, V. I., & Smalheiser, N. R. (2009). Author name disambiguation in MEDLINE. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *3*(3), 11. doi: 10.1145/1552303.1552304

Torvik, V. I., Weeber, M., Swanson, D. R., & Smalheiser, N. R. (2005). A probabilistic similarity metric for medline records: A model for author name disambiguation. *Journal of the American Society for information science and technology*, *56*(2), 140–158. doi: 10.1002/asi.20105

Tran, H. N., Huynh, T., & Do, T. (2014). Author name disambiguation by using deep neural network. In *Proceedings of the asian conference on intelligent information and database systems* (pp. 123–132). Bangkok, Thailand: Springer. doi: 10.1007/978-3-319-05476-6_13

Tran, T., Ladwig, G., & Rudolph, S. (2013). Managing structured and semi-structured RDF data using structure indexes. *IEEE Transactions on Knowledge and Data Engineering*, *25*(9), 2076-2089. doi: 10.1109/TKDE.2012.134

Turian, J. P., Ratinov, L., & Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 384–394). Uppsala, Sweden: The Association for Computer Linguistics. Retrieved from `http://www.aclweb.org/anthology/P10-1040`

Tzelepis, C., Mavridaki, E., Mezaris, V., & Patras, I. (2016). Video aesthetic quality assessment using kernel support vector machine with isotropic gaussian sample uncertainty (KSVM-IGSU). In *Proceedings of the 2016 IEEE international conference on image processing* (pp. 2410–2414). Phoenix, Arizona, USA: IEEE. doi: 10.1109/ICIP.2016.7532791

Tzelepis, C., Mezaris, V., & Patras, I. (2015). Linear maximum margin classifier for learning from uncertain data. *arXiv preprint*. Retrieved from `https://arxiv.org/abs/1504.03892`

Tzelepis, C., Mezaris, V., & Patras, I. (2016). Video event detection using kernel support vector machine with isotropic gaussian sample uncertainty (KSVM-iGSU). In *Proceedings of the 22nd international conference on multimedia modeling* (pp. 3–15). Miami, FL, USA: Springer. Retrieved from `https://www.iti.gr/~bmezaris/publications/mmm16_1_preprint.pdf`

Ueki, K., Kikuchi, K., & Kobayashi, T. (2016). Waseda at TRECVID 2016: Ad-hoc Video Search. In *TRECVID 2016 workshop*. Gaithersburg, MD, USA: NIST. Retrieved from `http://www-nlpir.nist.gov/projects/tvpubs/tv16.papers/waseda.pdf`

van Ham, F., & Perer, A. (2009). "Search, Show Context, Expand on Demand": Supporting Large Graph Exploration with Degree-of-Interest. *IEEE Transactions on Visualization and Computer Graphics*, *15*(6), 953–960. doi: 10.1109/TVCG.2009.108

Vargas, A., Weffers, H., & da Rocha, H. V. (2010). A method for remote and semi-automatic usability evaluation of web-based applications through users behavior analysis. In *Proceedings of the 7th international conference on methods and techniques in behavioral research* (pp. 19:1–19:5). Eindhoven, Netherlands: ACM. doi: 10.1145/1931344.1931363

Vigo, M., & Harper, S. (2017). Real-time detection of navigation problems on the world 'wild' web. *International Journal of Human-Computer Studies*, *101*, 1–9. doi: 10.1016/j.ijhcs.2016.12.002

Ware, C. (2012). *Information visualization: perception for design*. Elsevier.

Weihua Huang, C. L. T., & Leow, W. K. (2005). Associating text and graphics for scientific chart understanding. In *Proceedings of the eighth international conference on document analysis and recognition*

(pp. 580–584). Seoul, Korea: IEEE Computer Society. doi: 10.1109/ICDAR.2005.54

Xu, S., & Krauthammer, M. (2010). A new pivoting and iterative text detection algorithm for biomedical images. *Journal of Biomedical Informatics*, *43*, 924-931. doi: 10.1016/j.jbi.2010.09.006

Yang, K.-H., Peng, H.-T., Jiang, J.-Y., Lee, H.-M., & Ho, J.-M. (2008). Author name disambiguation for citations using topic and web correlation. In *Proceedings of the international conference on theory and practice of digital libraries* (pp. 185–196). Aarhus, Denmark: Springer. doi: 10.1007/978-3-540-87599-4_19

Yang, L., Huang, W., & Tan, C. L. (2006). Semi-automatic ground truth generation for chart image recognition. In *Document analysis systems vii, 7th international workshop* (Vol. 3872, pp. 324–335). Nelson, New Zealand: Springer. doi: 10.1007/11669487_29

Yilmaz, E., Kanoulas, E., & Aslam, J. A. (2008). A simple and efficient sampling method for estimating AP and NDCG. In *Proceedings of the 31th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 603–610). Singapore: ACM. doi: 10.1145/1390334.1390437

Zaiane, O. R., Xin, M., & Han, J. (1998). Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In *Proceedings. ieee international forum on research and technology advances in digital libraries* (pp. 19–29). Santa Barbara, CA, USA: IEEE. doi: 10.1109/ADL.1998.670376

Zamani, H., & Croft, W. B. (2016). Embedding-based query language models. In *Proceedings of the 2016 international conference on the theory of information retrieval* (pp. 147–156). Newark, DE, USA: ACM. doi: 10.1145/2970398.2970405

Zhai, C. (2008). Statistical language models for information retrieval: A critical review. *Foundations and Trends in Information Retrieval*, *2*(3), 137–213. doi: 10.1561/1500000008

Zhangy, H., Pangy, L., & Y.-J. Luy, e. a. (2016). VIREO @ TRECVID 2016: Multimedia Event Detection, Ad-hoc Video Search, Video to Text Description. In *TRECVID 2016 workshop*. Gaithersburg, MD, USA: NIST. Retrieved from `http://www-nlpir.nist.gov/projects/tvpubs/tv16.papers/vireo.pdf`

Zheng, G., & Callan, J. (2015). Learning to reweight terms with distributed representations. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval* (pp. 575–584). Santiago, Chile: ACM. doi: 10.1145/2766462.2767700