1 Background

In this section, we will give an overview of the needed theoretical background to understand this thesis.

1.1 Big Data

Todays society is witnessing an exponential increase in the amount of generated data. For instance, retailers, social medias, organizations in financial and healthcare sectors are all generating large volumes of data that need to be collected and analyzed to make the appropriate decisions. In this section, we will give an overview of Big Data characteristics and analytics methods.

1.1.1 Definition

The term Big Data applies to information that could not be processed using traditional processes or tools [3]. For instance, Edd Dumbill defined Big Data according to its processing aspect as follows:
”Big data is data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesnt fit the structures of your database architectures” [2].
Another definition of Big Data based on the three dimensions: *Volume*, *Variety* and *Velocity* has emerged. For instance, Gartner Inc. defined Big Data as follows:
”Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation.” [1].
Some other definitions add the term *Veracity* to the 3 V model to characterize Big Data [4].
In the remainder of this section, we will go through each of these Big Data aspects.

1. Volume:
   refers to the quantity of data which is reported in multiple terabytes and petabytes.

2. Variety:
   refers to the diverse types of data which can be:

structured data refers to information with a high degree of organization, such that inclusion in a relational database is seamless and readily searchable by simple, straightforward search engine algorithms or other search operations; whereas unstructured data is essentially the opposite. The lack of structure makes compilation a time and energy-consuming task. It would be beneficial to a company across all business strata to find a mechanism of data analysis to reduce the costs unstructured data adds to the organization. [?]

- structured data: refers to highly organized data that can be easily stored in a relational database. [?]

- unstructured data:. refers to data with no structural organization such as texts, images, audios and videos. [?]

- semi-structured data: refers to data that have some organizational properties, after some process it could be stored in a relational database. Examples of this type of data include XML and JSON documents. NoSQL databases are considered as semi structured. [?]

3. Velocity:
   it refers to the rate at which data are generated and the speed at which it should be analyzed and acted upon [?].

4. Veracity:
   it refers to the biases, noise and abnormality in data.

1.1.2 Big Data analytics

Big Data analytics aims to obtain meaningful results from big data by providing algorithms that include statistical models, machine learning, text analytics and other advanced data-mining techniques.

1.1.2.1 Text analytics

Text analytics deals with analyzing unstructured text, extracting relevant information, and transforming it into structured information that can then be leveraged in various ways. It uses techniques from Natural Language Processing (NLP), knowledge discovery, data mining, information retrieval, and statistics..

1. Natural Language Processing:
   NLP is a complex field that aims to develop tools and techniques to make computer systems understand and manipulate natural languages to perform the desired tasks. It generally makes use of linguistic concepts such as grammatical structures and parts of speech to determine who did what to whom, when, where, how, and why [20] [6].
   NLP performs analysis on text at different levels:

   ◇ Lexical/morphological: deals with understanding the meaning of words and parts of speech (such as noun, verb, adjective) in the context of the text provided. The lexical analyzer connects each word with its corresponding label in a dictionary. The morphological analyzer identifies words or phrases in one sentence and marks each word with a token symbol. The identified tokens are classified according to their grammatical class [?].

- ◇ Syntactic analysis aims to check that a sentence is well formed and to break it up into a structure that shows the syntactic relationships between the different words. This is done using a dictionary of word definitions and a set of grammar rules [?].
- ◇ Semantic analysis determines the possible meanings of a sentence. This can include examining word order and sentence structure and disambiguating words by relating the syntax found in the phrases, sentences, and paragraphs.
- ◇ Discourse-level analysis attempts to determine the meaning of text beyond the sentence level [20].

These techniques are generally combined with other statistical or linguistic techniques to automate the tagging and markup of text documents to extract the following informations [20]:

- ◇ Terms: these are the keywords.
- ◇ Entities: or named entities, these are specific examples of abstractions (tangible or intangible). Examples are names of persons, names of companies, geographical locations, contact information, dates, times, currencies, titles and positions, and so on.
- ◇ Facts: Also called relationships. They indicate the who/what/where relationships between two entities.
- ◇ Events: events usually contain a time dimension and often cause facts to change.
- ◇ Concepts: These are sets of words and phrases that indicate a particular idea or topic with which the user is concerned. This can be done manually or by using statistical, rule-based, or hybrid approaches to categorization. Concepts can be defined by users to suit their particular needs.
- ◇ Sentiments: Sentiment analysis is used to identify viewpoints or emotions in the underlying text. Some techniques do this by classifying text as, for example, subjective (opinion) or objective (fact), using machine learning or NLP techniques.

1.1.3 Big Data Processing Frameworks

The MapReduce [?] framework stayed for a long time the reference standard for big data processing, however, it is not suited for wide range of applications, such as real time and machine learning. In the last years, a variety of big data processing engines that tackled its shortcomings were developped. MapReduce revolutionized computation over huge data sets by offering a simple model for writing programs that could execute in parallel across hundreds to thousands of machines.

1.1.3.1 The difference between batch and stream processing

- batch processing: refers to a type of engines that process finite batch data sets as bounded data [5]. Batch processing systems are based on the ETL technology. ETL tools are used to transform the data into the format required by the data warehouse, which provide business users with a way to consolidate information across disparate sources (such as enterprise resource planning [ERP] and customer relationship management [CRM]) to analyze and report on data relevant to their specific business focus, by performing the following functions:

  - Extract: Read data from the source database.
  - Transform:Convert the format of the extracted data so that it conforms to the requirements of the target database. Transformation is done by using rules or merging data with other data.
  - Load: Write data to the target database.

  ETL tools extracted the data to an intermediary location to perform the transformation before loading the data to the data warehouse. Data warehouses provide business users with a way to consolidate information across disparate sources (such as enterprise resource planning [ERP] and customer relationship management [CRM]) to analyze and report on data relevant to their specific business focus.

- stream processing: According to [35], streaming data is data that is generated continuously by thousands of data sources, which typically send in the data records simultaneously, and in small sizes (order of Kilobytes). It can also be described as infinite or unbounded data [5]. Tylor Akidu, an engineer in the Google Cloud DataFlow team, defined stream processing as data processing engine that is designed with infinite data sets in mind [5]. For instance, an online gaming company that collects streaming data about player-game interactions, and feeds the data into its gaming platform. It then analyzes the data in real-time, offers incentives and dynamic experiences to engage its players.

In a batch-processing scenario, the transactions databases collect event data during the day, which is bulk loaded to a staging database at night. A large ETL process then executes to produce aggregations on this data and store these results to the data warehouse. The Business Intelligence applications then present the reports based on these aggregates to the business users the following day. Users have to wait an entire day to get insights into business operations.
Streaming systems operate differently. The transactions database will stream the logs in near real-time which are processed by a streaming system which will produce aggregations within intervals such as time. For example, calculate aggregates like total sales by region for every 5-minute interval. We will soon see that other types of intervals are also possible in Flink. These aggregates will be stored in the Data Warehouse which provide a more real-time insight into business operations to the business users. The intervals should not be too small (ex. 5-seconds) because Data Warehouses are typically designed for optimal reads (lots of indexes) which will slow down the writing of the granular aggregates and cause the overall system to have bottlenecks.

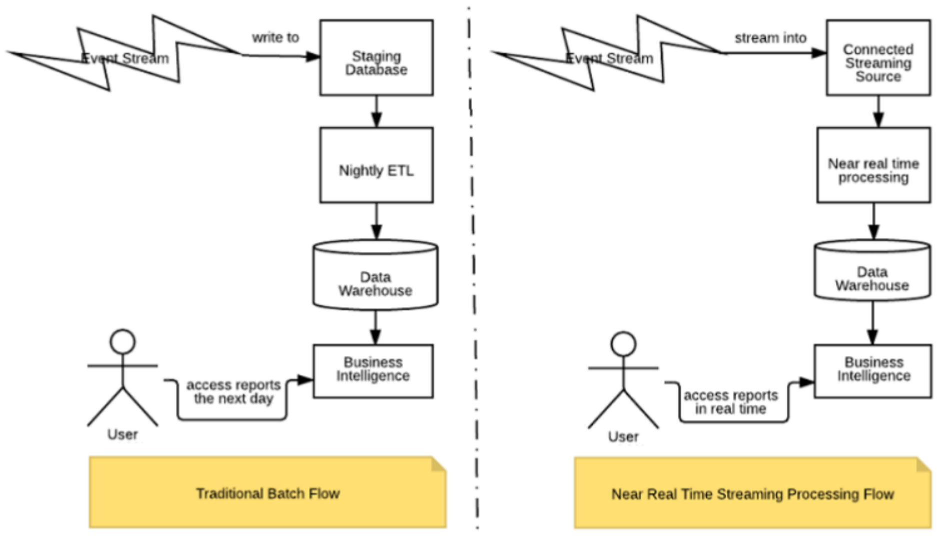


Figure 1.1: Comparison between batch and stream processing dataflows [32]

1.1.3.2 Apache Flink

Apache Flink is an open source platform that uses a common runtime for scalable batch and stream data processing. Flink's stack includes the DataStream and the DataSet APIs for creating stream processing and batch processing applications respectively. It also bundles libraries for domain-specific use cases such as Machine learning and graph analytics. A Flink Program can be easily deployed according to three different modes: locally, on a existing YARN or a Standalone cluster or on existing clusters in the cloud.

- Local Mode: ...

- Standalone Cluster Mode: ...

- YARN:...

- Flink's architecture overview:
After a Flink program is submitted, three processes are put into place to compile it:

- A **Client** is created that performs the pre-processing and transforms the program into an optimized JobGraph, a generic parallel data flow with arbitrary tasks that consume and produce data streams, and sends it to the JobManager. After that, it can disconnect, or stay connected to receive progress reports.The client program runs either as part of the Java/Scala program that triggers the execution, or in the command line process.

- The **JobManagers** (also called masters) they ensure the program parallelization by co-ordinating the distributed execution of the execution Graph. They schedule tasks by assigning tasks to Task Managers, they also perform other tasks such as checkpoints co-ordination and recovery on failures coordination.

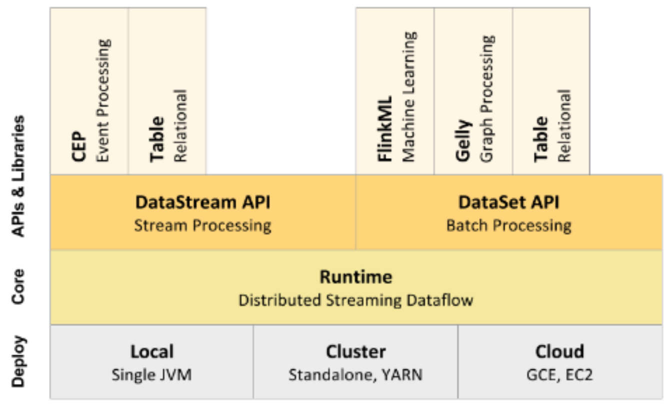


Figure 1.2: Flink Stack [20]

A Flink program uses at least one Job Manager. To ensure high-availability, a setup would use multiple JobManagers, with one leader and the other JobManagers as standby .

- The **TaskManagers** (also called workers) execute the tasks (or more specifically, the subtasks) of a dataflow, and buffer and exchange the data streams. Operations are split up into tasks depending on the specified parallelism Each parallel instance of an operation runs in a separate task slot The scheduler may run several tasks from different operators in one task slot There must always be at least one TaskManager.
  TaskManagers connect to JobManagers, announcing themselves as available, and are assigned work.

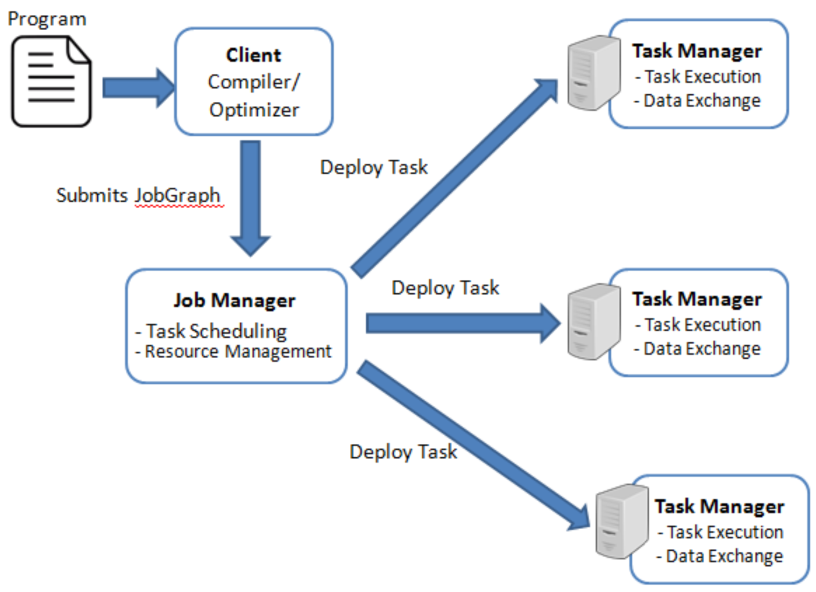


Figure 1.3: Flink Architecture Overview

- Apache Flink Programming Model:

- Program Dataflow:

A Flink program can be viewed as a streaming dataflow that takes the data streams from a source, performs transformation operations on it and stores the resulting streams in a sink. Programs in Flink are inherently parallel and distributed. Data sources create the

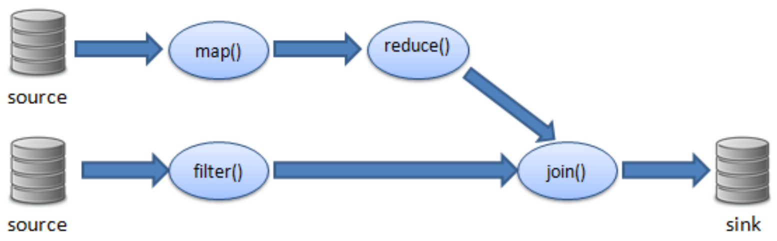


Figure 1.4: Flink Streaming dataflow

initial Datasets, from Text or CSV files or from Java collections.
Data sinks store the returned values after the transformations in a Dataset.
Data transformations transform one or more DataSets into a new DataSet. Programs can combine multiple transformations into sophisticated assemblies. Some common transformations and their definitions according to Flink's documentation [**?**] include:

- Map():
  Takes one element and produces one element.
- Filter():
  Evaluates a boolean function for each element and retains those for which the function returns true.
- Reduce():
  Combines a group of elements into a single element by repeatedly combining two elements into one. Reduce may be applied on a full data set, or on a grouped data set.
- Join():
  Joins two data sets by creating all pairs of elements that are equal on their keys.

- **Time:** Time When processing data which relate to events in time, there are three inherent domains of time to consider.

  - Event time:
    is the time at which the event itself actually occurred, it is usually described by a timestamp attribute in the event data and is assigned by the source of the event. Flink accesses event timestamps via timestamp assigners. Once assigned, the event time for a given event never changes.
  - Ingestion time:
    This is the time the event arrives to the Flink dataflow at the source operator. It is assigned by the Flink system. Like the event time, it is assigned once and never changes.

- ○ Processing Time:
  is the current time according to the system clock at which an event is observed at any given point during processing at each operator that performs a time-based operation. It changes constantly for each event as it arrives for processing at an operator instance.

- Window:
  Windowing slices up a dataset into finite chunks for processing as a group, it contains multiple events on which several operation types such as aggregation can be applied on events in a window. Windowing can be applied to both batch and stream processing.
  In batch processing where bounded data is processed, windowing is an optional concept of semantic usefulness.
  In stream processing where unbounded data is processed, the windowing concept is crucial for some operations to delineate finite boundaries in most forms of grouping such as aggregation and unnecessary for others such as filtering and mapping.
  Flink contains two windows types: built-in time and count based windows that can be further subdivided into tumbling and sliding windows.

  - Time-based windows: time windows group stream elements by time. They can be defined as tumbling or sliding.
    - Tumbling windows: A tumbling time window collects events for a specific time and applies a function on all events in the window after the time is passed and each event is assigned to one only one window.
      Let's consider the example of an online gaming application, where a gaming server receives immediately after the completion of an instance of game data streams that contain the player identifier, an event time stamp and the score. let's suppose we want to track the generated events each 5 seconds. Figure 1.5 illustrates this example.

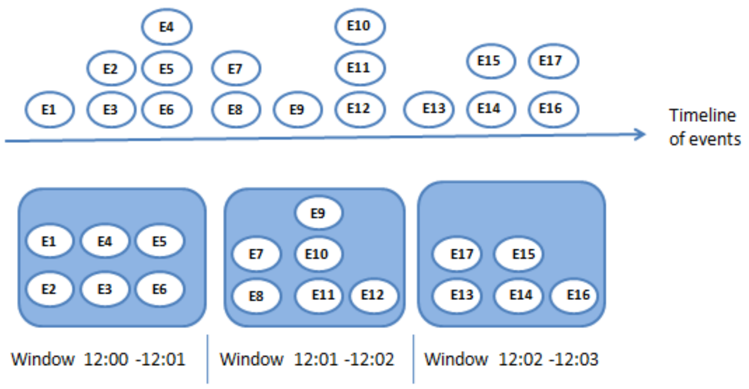
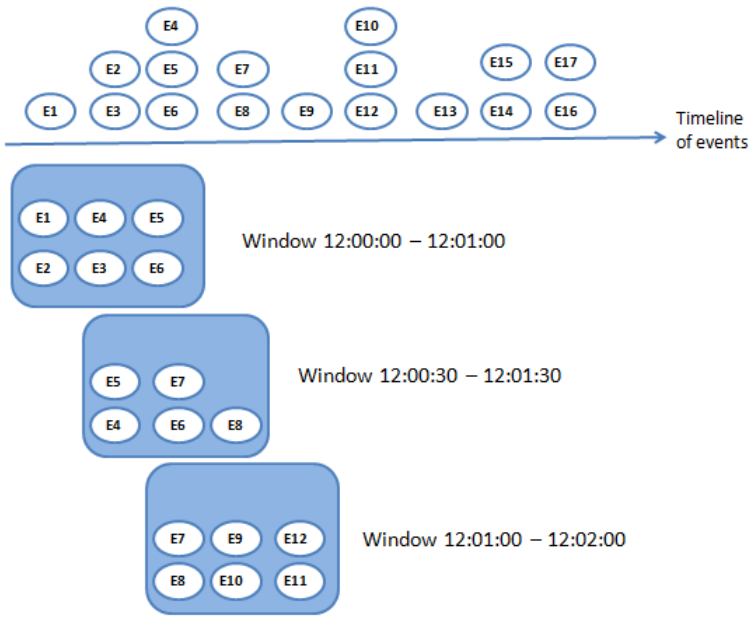


Figure 1.5: Tumbling windows

Time windows can also be sliding.Figure 1.6 illustrates a sliding window which is one minute long which slides every 30 seconds.

  - Count-based windows:
    in this case, each window is evaluated based on the number of elements it contains.



E1
E2
E3
E4
E5
E6
E7
E8
E9
E10
E11
E12
E13
E14
E15
E16
E17
Timeline of events
Window 12:00:00 – 12:01:00
Window 12:00:30 – 12:01:30
Window 12:01:00 – 12:02:00

Figure 1.6: Sliding windows

We also distinguish here between tumbling and sliding count based windows.
In tumbling count window, the window pane is evaluated when the element count in the pane reaches a pre-defined level, whereas in sliding count windows, the window pane is evaluated based on the number of elements it contains and how much it slides by.

1.1.3.3 Apache Spark

Apache Spark is an open source framework that combines an engine for distributing programs across clusters of machines with an elegant model for writing programs atop it which was originated at the UC Berkeley AMPLab[18].
Figure 1.7 depicts the Apache Spark stack components.

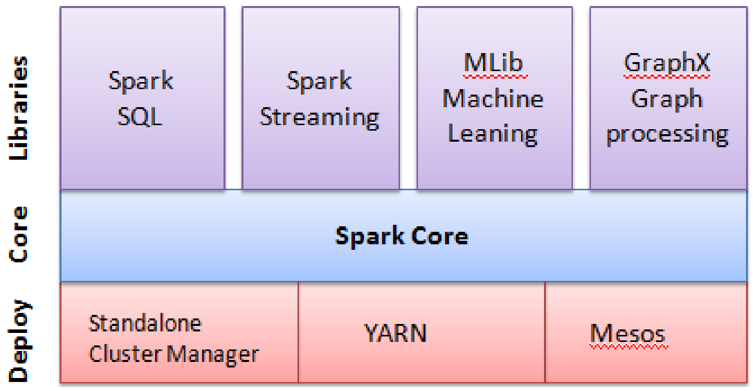


Figure 1.7: The components of the Spark Stack

A Spark program follows a master/slave architecture with one central coordinator called driver that launches various parallel operations on cluster nodes called executors via a cluster manager such as Hadoop YARN or the standalone Hadoop manager. Figure 1.8 depicts the components of a distributed Spark application. The Spark engine implicitly creates a logical

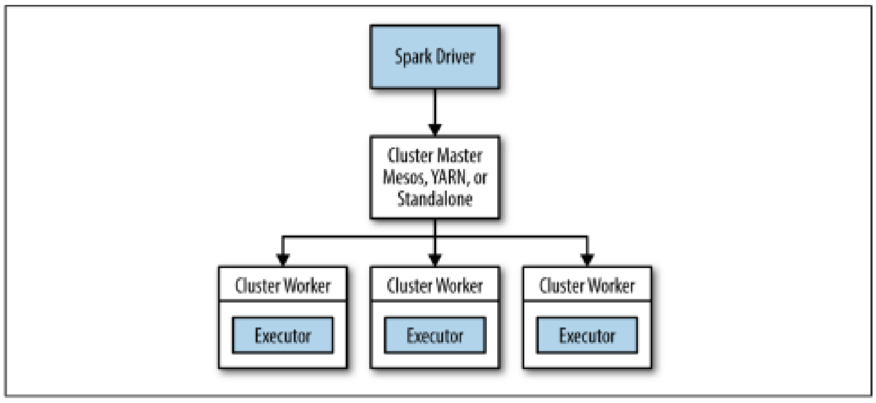


Figure 1.8: The components of a distributed Spark application[33]

directed acyclic graph (DAG) of operations and optimizes this graph by rearranging and combining operators where possible. For instance if we suppose that a Spark job contains a map operation followed by a filter operation, the Spark DAG optimizer would rearrange the order of these operators, so that the number of records would be reduced before a map operation takes place [33].
When the driver runs, it converts this logical graph into a physical execution plan, where each stage of this plan consists of multiple tasks. The Spark Driver tries to schedule each task in an appropriate Executor.
The resilient distributed dataset (RDD)is the Sparks core abstraction for working with data, a fault-tolerant collection of elements that can be operated on in parallel [17]. Every Spark program consists of creating some input RDDs from external data and performing operations on those RDDs. The two major types of operations are:

- ◇ **Transformations:** They take an RDD as an input and produce one or many RDDs. Existing transformations include map(), filter()
- ◇ **Actions:** They are computations performed on a RDD that return a value. Some supported actions include reduce()

The RDD contents are stored into memory across the cluster nodes, meaning that the future recomputations on an RDD content need not to recompute it or reload it from disk [18].

1.1.3.4 Performance comparison between Apache Spark and Apache Flink

Ovidiu-Cristian Marcu et al. provided in [**?**] a performance comparison between Apache Spark and Apache Flink. The authors performed a series of extensive experiments involving six representative workloads for batch and iterative processing.
As for batch workloads, three benchmarks implementing the Word Count[**?**], Grep[**?**] and Tera Sort[**?**] were selected.
As for iterative workloads, three benchmarks evaluating the loop-caching: K-Means, Page

Rank[**?**] and Connected Components[**?**] algorithms were selected.
For both frameworks, some parameters have influence on some performance metrics such as the overall execution time, scalability and resource consumption.
These parameters manage the task parallelism, the network behavior during the shuffle phase, the memory and the data serialization. For the batch workloads, the goal was to validate strong and weak scalability. For the iterative workloads, the focuse was on scalability, caching and pipelining performance.
Two experiment secanrios were conducted, the first scenario consists of having the same dataset size and increasing the number of nodes to characterize the weak scalability, the second scenario consists of having the same the number of nodes and increasing the dataset size to characterize the strong scalability.

- Batch Workload:

	Word Count	**Grep**	**Tera Sort**
Weak Scalability	both frameworks show a similar performance for a small number of nodes (2 to 8). For a larger number (16 and 32), Flink performs slightly better.	Spark outperforms Flink, with up to 20% smaller execution times for large datasets (16 and 32 nodes).	Flink is performing on average better than Spark, showing smaller execution time.
Strong Scalability	Flink constantly outperforms Spark by 10%.	Spark outperforms Flink	Flinks advantage is increasing with larger clusters.

- Iterative Workload: - KMeans: A dataset of 51 GB has been used in both platforms to evaluate the performance. Flink outperforms Spark by more than 10% .

- Page Rank and Connected components : Small, medium and large graphs have been used to evaluate the performance of these two algorithms.
For small graphs, Flink performs slightly better for both algorithms. For medium-size graphs, for the Connected Components algorithm, Flink outperforms Spark by 30%. Whereas Spark is more efficient for large graphs.

Spark is about 1.7x faster than Flink for large graph processing, while the latter outperforms Spark up to 1.5x for batch and small graph workloads using sensitively less resources and being less tedious to configure.

1.2 Machine Learning

Machine Learning is the science of providing computers with the ability to learn from data [14].
According to Tom Mitchell, a computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E [24].
For instance, the spam filter is a machine learning program that takes a training set composed

of spam emails that are labeled by the user and non-spam emails. The system learns form this training data, which constitutes the experience E to perform the task T of labeling new emails as spam or not. The performance P can be measured by calculating the ratio of correctly classified emails [14].
Machine Learning systems can be classified based on the amount of human supervision they get while being trained. In the remainder we will give an overview of the three types of learning: supervised, unsupervised and semi-supervised learning.

1.2.1 Unsupervised Learning

In unsupervised learning, as you might guess, the training data is unlabeled. The system tries to learn without a teacher. Examples of unsupervised learning algorithms include: Clustering, Visualization and dimensionality reduction.

1.2.1.1 Label Propagation Algorithm

This algorithm was proposed by Raghavan et al. in [36]. It is graph-based semi-supervised learning algorithm, where the dataset consists of both labeled and unlabeled datapoints.
Let G(V, E) be an undirected network where V is the set of vertices and E is the set of edges. Each node v(v$\in$ V) has a label at time t $L_v(t)$. We denote by $N_(v)$ the set of neighbors of vertex v. the algorithm can be described as follows:

1. Initialize the labels at all nodes in the network. For a given vertex v , $L_v(0)$ = v.

2. In the asynchronous version, where at every update one node updates its current label, for each v $\in$ V chosen in a random order, let:
   $L_v(t) = f(L_(v_(i1))(t),...,L_(v_(im))(t),L_(v_(i(m+1)))(t1),...,L_(v_(ik))(t1))$.
   where $v_(i1)$ ... $v_(im)$ are neighbors of v that have already been updated in the current iteration while $v_(im+1)$ ... $v_(ik)$ are neighbors that are not yet updated in the current iteration.
   In the synchronous version, where at every update each node updates its current label, for each v $\in$ V chosen in a random order:
   $L_v(t) = f(L_(v_(i1))(t-1),...,L_(v_(ik))(t1))$. f returns the label occurring with the highest frequency among neighbors and ties are broken uniformly randomly.

3. The algorithm converges if every node has a label that the maximum number of their neighbors have, otherwise t is set to t+1, another label propagation iteration is done and the labels are recalculated.

The Figure 1.11 shows the two clusters obtained after running the Label Propagation algorithm on that graph. Each cluster is identified by a color. We notice that this algorithm identifies the highly connected components in a single cluster.

According to [22], this algorithm presents some advantages such as it doesn't require parameter, it is easy to implement, fast to execute for large networks, and able to detect valid clusters even in random graphs. However, it may lead to assigning the same label to disconnected communities.

1.2.1.2 Kmeans

Kmeans is an unsupervised learning algorithm that aims to organise data from a given training set into few clusters. The Kmeans algorithm proceeds in two steps: First K data items from the set of data points are randomly chosen as initial centroids, then, in the second step, each data point is assigned to the cluster which has the closest centroid and each cluster centroid is moved to the mean of the points assigned to it. This second step is repeated until all data points are assigned to one of the K clusters.

The Kmeans algorithm proceeds in two steps:
- First K data items from the set of data points are randomly chosen as initial centroids.
- In the second step, each data point is assigned to the cluster which has the closest centroid according to a chosen distance measure and each cluster centroid is moved to the mean of the points assigned to it. This second step is repeated until all data points are assigned to one of the K clusters.

Figure 1.12 illustrates those steps. The circles refer to the datapoints and the triangles to the centroids. The datapoints with the same color belong to the same cluster.

- Silhouette Value:

A silhouette is a graphical display for partition techniques which shows which objects lie well within their cluster, and which not. The average silhouette width provides an evaluation of clustering validity and might be used to select an 'appropriate' number of clusters [**?**].
For each data point i, the silhouette $s(i)$ is given by the equation 1.1:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{1.1}$$

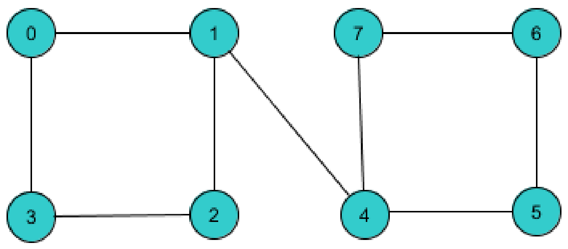


Figure 1.9: Initial graph

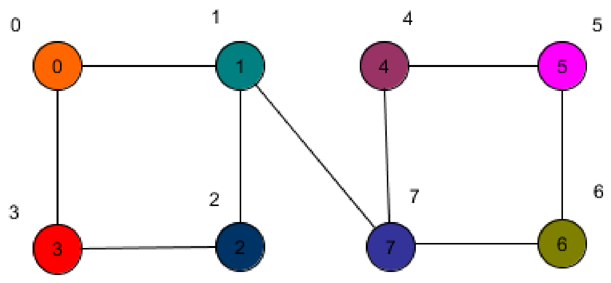


Figure 1.10: First Iteration

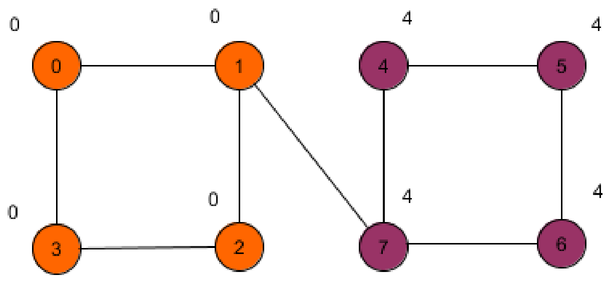


Figure 1.11: Final Iteration

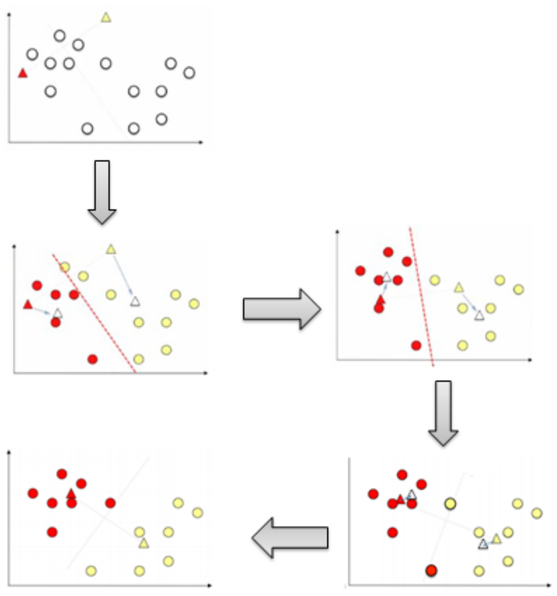


Figure 1.12: KMeans clustering example[23]

where $a(i)$ is the average dissimilarity of i with all other data within the same cluster and $b(i)$ is the lowest average dissimilarity of i to any other cluster, of which i is not a member.

The instances are colored according to which cluster they belong to. the centroid values are recomputed according to the euclidean distance

1.2.2 Supervised Learning

In supervised learning, the training data you feed to the algorithm includes the desired solutions, called labels.
A typical supervised learning task is classification. Figure **??** illustrates an example of a classification task: the system is trained with many labeled images, the label can be ”airplane” or ”car”, the machine-learning process analyzes those labeled images and builds a model. This model is then used by the prediction process to classify unlabeled images. All the images were correctly labeled.

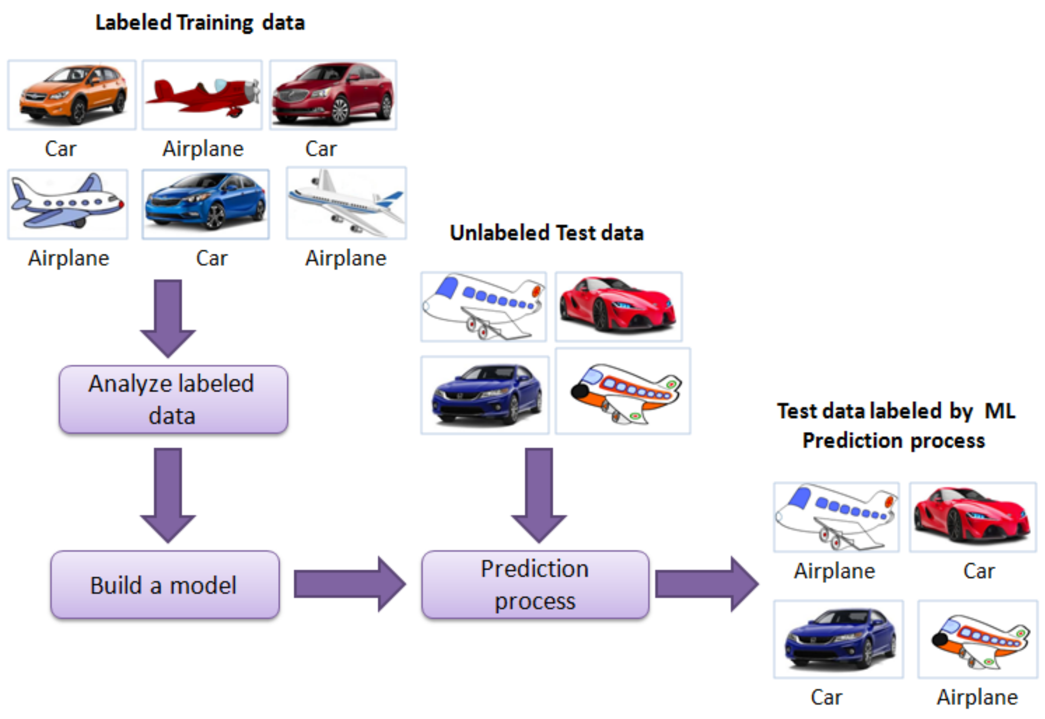


Figure 1.13: Workflow of a classification example

1.2.2.1 Logistic Regression

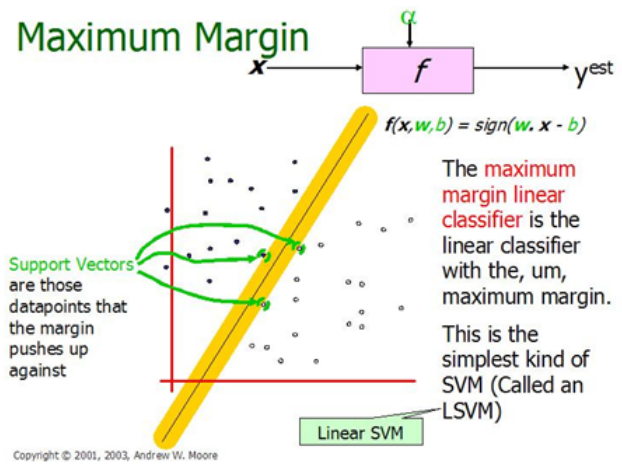
1.2.2.2 Support Vector Machines

Support Vector Machines have been introduced in 1992 at the COLT conference by Boser, Guyon, Vapnik as a supervised learning approach used to solve classification and regression problems [29].

The simplest kind of support vector machines called Linear SVM aims to find a hyperplane that optimally separates the training vectors into two classes by achieving the maximum separation, such that it will be equidistant to both datasets. Support vectors are defined to be a subset of the training data having the minimum distance to the hyperplane. An example of a simple linear SVM is illustrated in Figure 1.14.

The goals of SVM are separating the data with a hyper plane and extend this to non-linear boundaries when the data is far from linear and the datasets are inseparable by using the kernel methods that allow SVMs to form nonlinear boundaries, by non-linearly mapping the input data to a high-dimensional space. The new mapping is then linearly separable [8] [24].

The kernel functions enable operations to be performed in the input space rather than the potentially high dimensional feature space. Various kernel functions can be used [8], such as: *Polynomial*, *Gaussian Radial Basis Function*, as well as *Exponential Radial Basis Function*.



Maximum Margin

$f(x, w, b) = sign(w. x - b)$

The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Support Vectors are those datapoints that the margin pushes up against

Copyright © 2001, 2003, Andrew W. Moore

Figure 1.14: Linear Support Vector Machines [25]

2

State of the Art

This chapter gives an overview of existing conducted research that deal with the problem of concept detection from a given dataset.
The developed approaches of solving this problem depend on the dataset type. This chapter is divided into two sections. The first section deals with existing approaches on text datasets whereas the second section deals with existing approaches on image and video datatsets.

2.1 Concept Detection in Text Datasets

A lot of approaches were developed to detect concepts from texts. For instance, we can distinguish between Term Frequency-Inverse Document Frequency (TF-IDF) based approaches, ontology based approaches and graph based approaches.

2.1.1 TFIDF based approaches

TF-IDF

TF-IDF stands for term frequency-inverse document frequency and it was first presented in 1973 in [30]. TF-IDF is the most common term weighting scheme that evaluates the word importance to a document in a collection.
We suppose we have a query term t and a document d. The term frequency $tf_t(d)$ depends on the number of occurrences the term t appears in the document d and is calculated as follows:

$$tf_t(d) = \frac{Number\ of\ occurences\ of\ the\ term\ t\ in\ the\ document\ d}{Total\ number\ of\ terms\ in\ the\ document} \tag{2.1}$$

With the raw term frequency, all terms are considered equally important when it comes to assessing relevancy on a query. To attenuate the effect of too often occurring terms in the collection, the term frequency weight of a term t is reduced by the document frequency of the term t df_t, defined to be the number of documents in the collection that contain a term t. Therefore, the inverse document frequency is defined as follows:

$$idf_t = \log \frac{N}{df_t} \tag{2.2}$$

where N is the total number of documents in a collection. Thus the idf of a rare term is high, whereas the idf of a frequent term is likely to be low. By combining those twos definitions we get the following composite weight:

$$tf{-}idf_t(d) = tf_{t,d} \cdot idf_t \tag{2.3}$$

Concept Keyword Term Frequency/Inverse Document Frequency

A TF-IDF based metric called Concept Keyword Term Frequency/Inverse Document Frequency (ckTF/IDF) was presented in [16] to efficiency mine concept keywords from identifiers present in the software source code. They used this metric to extract the concept keywords from udos [15], an educational operating system consisting of 5,000 lines in C code.
- Concept Keyword:
concept keywords are a small subset of the words in identifiers and that represent a key concept that can aid in program understanding.
The authors distinguished between three kinds of concept keywords:

- Ideal concept keywords: they prove to enhance program understanding by performing an objective measurement.
- Human-selected concept keywords: are words that the code user believes they improve program understanding. For instance, *dirent*, which implies directory entry is a human selected concept keyword.
- Machine-extracted concept keywords: is an algorithm that aims to extract keywords that are approximate to the human selected ones.

- Definition of the ckTF/IDF method:
The ckTF/IDF method treats one source file as a document. It quantizes tf(t,d) and idf(t) into 0 or 1. The idf(t) is defined as follows:

$$IDF(t) = \begin{cases} 1 & \text{if} \quad 1 \leq df(t) \leq n \neg isprefix(t) \\ 0 & \quad otherwise \end{cases} \tag{2.4}$$

where df(t) is the document frequency of term t, n($\geq$ 1) the threshold (default is 1) to quantize tf(t, d) and idf(t) and isprefix(t) predicate is true if and only if t is a prefix for all its occurrences. The metric assumes that prefixes are unlikely to be concept words, for example in the identifier ”read_dirent” from udos [15], read is a prefix and is a generic verb not a concept keyword, whereas dirent is concept keyword.
The term frequency TF(t) in all documents.

$$tf(t) = \begin{cases} 1 & \text{if} \quad \in d, tf(t,d) > n \\ 0 & \quad otherwise \end{cases} \tag{2.5}$$

The word weight for ckTF/IDF is:

$$w(t) = tf(t) * idf(t) \tag{2.6}$$

A term t is selected as a machine-extracted concept keyword if w(t) = 1.
If we suppose that n=1 (the default value) then *w(t)*=1 if t exists in only one document, it is not

a prefix and there is one document where t occurs more than once.
The authors proposed a method to compute *w(t)* in a fast way. They defined two flags for each term: a local frequency denoted *local(t)* and a global frequency flag denoted *global(t)*. After dividing all identifiers in source code into terms by using some delimiters like underscores, the two flags are computed for each term as follows:

- if a term t is found twice in a document then local(t) is set.
- if this term t is found in two documents then global(t) is set.
- *w(t)*= 1 if local(t) is set, global(t) is clear, and the term is not a prefix otherwise *w(t)*= 0.

- Performance evaluation:
The authors developed a framework called Identifer Exploratory Framework (IEF) for the ckTF/IDF method. To evaluate the performance of the ckTF/IDF method, the authors applied the ckTF/IDF and TF/IDF methods to their educational operating system udoss source code, around 5,000 lines in C [15] and compared the concept keywords extracted by a human programmer with that found by the algorithm. Therefore, they computed the performance metrics Accuracy and Coverage, which they defined as follows:

$$Accuracy = \frac{C_r}{C_m}$$

where C_r is the total number of concept keywords that both human and algorithm have chosen and C_m is the total number of all machine extracted concept keywords.

$$Coverage = \frac{C_r}{C_h}$$

where C_h is the total number of human extracted concept keywords.
For the ckTF/IDF method, they achieved an accuracy of 57% and a coverage of 26%, whereas for the TF/IDF method, they obtained as accuracy of 28% and a coverage of 13%. Thus, ckTF/IDF delivers twice better accuracy and coverage than TF/IDF.
The authors have also applied the ckTF/IDF and TF/IDF methods on the Ruby interpreter to compare the execution speeds and found that ckTF/IDF is about 6 times faster than TF/IDF.

2.1.2 Ontology based approaches

In this section, we will present two ontology based approaches to detect documents' concepts. First, let us define what is an ontology.

2.1.2.1 What is an ontology?

According to R. Studer et al., an ontology as a formal, explicit specification of a shared conceptualization [**?**]. In other words, an ontology is composed of a set of vocabularies used to explicitly and consensually describe a topic area and a set of explicit hypotheses on the meaning of terms. It is expressed as a set of objects and the relationships between them and is described in a formal language.

2.1.2.2 Ontology based method for topic identification of learning materials

The authors in [27] presented an ontological approach to automatically identify major topics covered in learning materials, also the subject and discipline to which those topics belong and relevance of the topic in the learning material as compared to other topics present in the same document are also discovered.

- Proposed approach: The system stores the domain ontologies of various subjects, depicting the topics and terms relating to the subject and the relationships between them. For instance, Figure 2.1 shows an example of an ontology of the computer science subject. The authors designed the domain ontology with multiple layers defined as follows:

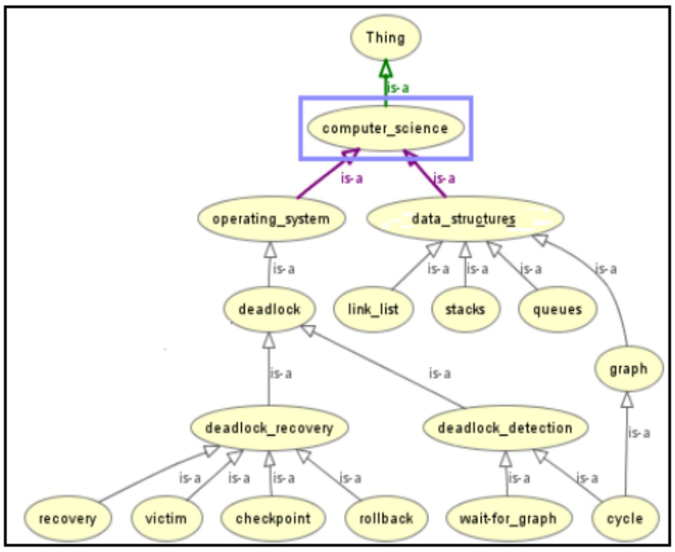


Figure 2.1: Domain Ontology of the subject Computer Science [27]

- The top layer contains disciplines like Computer Science.
- The second layer contains Subjects under that discipline.
- The third layer contains the broad topics covered under that subject.

Topics again can contain sub-topics. These subtopics may be again represented by collection of various terms. Figure 2.2 depicts the workflow for the topic extraction process. After the user submits doc or html files, the Standford Parser [**?**] skimd these document for noun phrases, the stop words are deleted from those phrases which generates keyphrases that are ranked according to word frequency and displayed for user approval. These approved keyphrases are compared to the terms present in the domain ontology. If the domain ontology contains these keyphrases, then the OWL Parser retrieves the corresponding topics and subtopics are listed out and the relevance of document with respect to topic(s) /subtopic(s) enclosed in the learning material is calculated as follows:

$$Relevance_{topic}(document) = \frac{number\ of\ its\ key\ terms\ identified\ under\ the\ topic/subtopic}{Total\ number\ of\ key\ terms\ identified\ in\ the\ entire\ document}$$

Document can be characterized using the topic with higher relevance.

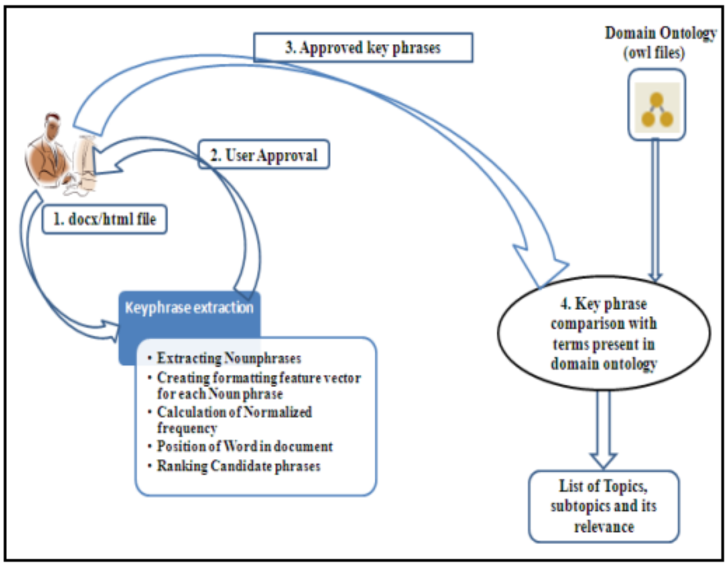

3. Approved key phrases
Domain Ontology (owl files)
2. User Approval
1. docx/html file
Keyphrase extraction
• Extracting Nounphrases
• Creating formatting feature vector for each Noun phrase
• Calculation of Normalized frequency
• Position of Word in document
• Ranking Candidate phrases
4. Key phrase comparison with terms present in domain ontology
List of Topics, subtopics and its relevance

Figure 2.2: Workflow of the topic extraction process [27]

- **Performance evaluation:** Documents consisting of 200 learning materials of different subjects were processed by the developed tool. The performance evaluation was done in two phases:

  - First phase:
    The tool generated topics/subtopics were compared to those listed by subject experts. The authors used the F-score metric to evaluate the results. It is defined as follows:

    $$F_\beta = \frac{(\beta^2+1)\cdot Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$

    β = 1 given precision and recall equal weights.
    Precision and Recall are defined as follows:

    $$Precision = \frac{Number\ of\ topic(s)/subtopic(s)\ identified\ correctly\ by\ the\ system}{Total\ topic(s)/subtopic(s)\ generated\ by\ the\ system}$$

    $$Recall = \frac{Number\ of\ topic(s)/subtopic(s)\ identified\ correctly\ by\ the\ system}{Number\ of\ topic(s)/subtopic(s)\ identified\ by\ the\ authors}$$

  - Second phase:
    In this phase, the topics/subtopics that were generated by the system but not listed out by the authors are shown to the authors for approval.

Strict and lenient evaluations were performed. In strict evaluation, while comparing the system generated output with the expert list of topics/subtopics, the partially matched topics/subtopics are considered not found, whereas in lenient evaluation, they considered as matched. The obtained results are:

- **Strict evaluation:** · Precision: 0.5057541
  · Recall: 0.8015656

· Fscore: 0.63

- Lenient evaluation: · Precision: 0.6273393

· Recall: 0.9609625

· Fscore: 0.76

The authors have agreed on 82.3% of the generated system topics/subtopics.

2.1.2.3 A Wikipedia ontology based approach for automatic topic identification

The authors in [21] extracted the background knowledge from Wikipedia, and convert it into a structured Wikipedia Hierarchical Ontology (WHO) and presented an approach to automatically identify the documents' topics using this ontology.
Wikipedia categories were used to represent the WHO concepts, then the representative articles were gathered and organized for each concept so that they can be used to identify the relevant terms for that concept. After that the articles were cleaned by removing all kinds of formatting, the stop words were removed and the words were stemmed. After the preprocessing of the articles, a weight is assigned for each term based on its importance to the concept according to the following equation:

$$TF - ICF = TF * \log(ICF) \tag{2.7}$$

where TF stands for the Term Frequency, it is the number of occurrences of a term in a concept divided by the total number of terms in that concept, and ICF is the total number of concepts divided by the number of concepts that contains this term. After the document-term matrix D is generated, the concept-term matrix C of the specific subset of concepts that we are interested in is extracted from WHO and matrix C^* is formed.
The document-concept similarity matrix S is calculated as follows:

$$S = DC^{*T} \tag{2.8}$$

2.1.3 Machine learning based approaches

In this section, we will give an overview of the Label Propagation Algorithm, that was used as a clustering algorithm in our approach and then present an example of a machine learning based approach.

2.1.3.1 Kmeans based Algorithm

The authors in [?] applied the Kmeans clustering algorithm to deal with the problem of topic detection. In their approach, they used the Vector Space Model for topic representation, K-means algorithm for text clustering and the Topic Detection and Tracking (TDT) method for performance evaluation. Figure 2.3 depicts the architecture of the topic detection prototype. In the remainder of this section, we will explain this diagram by going through each component.

1. **Topics Representation Model:**
The topic model is a statistical language model that reveals a hidden thematic structure of the text collection and finds a highly compressed representation of each document by a set of its topics. It is based on the idea that documents are mixtures of topics, where a

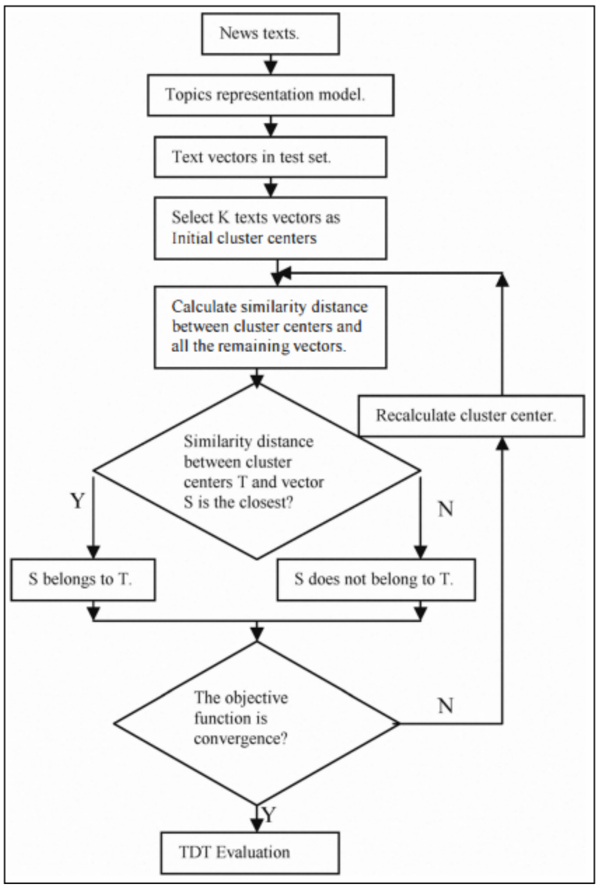

News texts.
Topics representation model.
Text vectors in test set.
Select K texts vectors as Initial cluster centers
Calculate similarity distance between cluster centers and all the remaining vectors.
Recalculate cluster center.
Similarity distance between cluster centers T and vector S is the closest?
Y
N
S belongs to T.
S does not belong to T.
The objective function is convergence?
N
Y
TDT Evaluation

Figure 2.3: Architecture for Topic Detection Prototype System

topic is a probability distribution over words [19] [11] [10].
Several topic modeling techniques were introduced in the literature such as Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA) and the Vector Space Model (VSM), which was applied in this prototype.
The Vector Space Model was introduced in 1971 by Gerard Salton [31]. In this model each document in a collection is represented as a vector in a vector space, the distance between the vectors is calculated and the vectors that are close to each other are semantically similar. To express a document text as a vector, the words are first extracted and the stop words are eliminated then the TF-IDF metric is calculated for each word in the document.
If we suppose the training set contains n terms after removing the words, then each document d in the training set is expressed as an n-dimensional feature vector V(d).

$$V(d) = (t_1, w_1(d); t_2, w_2(d); \cdots t_n, w_n(d)) \tag{2.9}$$

Where t_i(i=1,2,,n) is term i, and w_i(d)(i=1,2,,n) is the weight of the text d, calculated using the cosine normalized version of the TF-IDF algorithm.

$$\mathscr{W}_i(d) = \frac{-tf_i(d) \times \log(q/q_i)}{\sqrt{\sum_i (tf_i(d) \times \log(q/q_i))^2}} \tag{2.10}$$

A feature selection algorithm called Information gain was then applied to select an informative subset of text terms.
Information gain (IG) measures the amount of information in bits about the class prediction by knowing the presence or the absence of a term in a document [37].
For the term t and topic c, information gain [37] is defined as follows:

$$IG(t) = -\sum_{i=1}^{m} P(c_i)\log P(c_i) + P(t)\sum_{i=1}^{m} P(c_i|t)\log P(t) + P(\bar{t})\sum_{i=1}^{m} P(c_i|\bar{t})\log P(c_i|\bar{t}) \tag{2.11}$$

Where $P(c_i)$ is the probability that the topic i appears in the corpus, P(t) is the probability that a topic includes the term t, $P(c_i|t)$ is the conditional probability that the term t belongs to the topic i when the topic includes term t, $P(\bar{t})$ is the probability that a topic does not include the term t, $P(c_i|\bar{t})$ is the conditional probability that the term t belongs to the topic i when the topic does not include term t, and parameter m is the number of topics.

2. **KMeans Algorithm for Text Clustering:**
An overview of the Kmeans algorithm was given in 1.2.1.2. In the case of text clustering the Kmeans algorithm proceeds as follows:

- First K data items from the set of data points are randomly chosen as initial centroids or mean of a topic.
- In the second step, each remaining data point is assigned to the cluster which has the closest centroid according to the cosine distance between the text vectors, which is defined as follows:

$$Dis(d_i, d_j) = 1 - Sim(d_i, d_j) \tag{2.12}$$

Where d_i is text feature vector i, d_j is text feature vector j and Sim(d_i,d_j) is the cosine similar function between the document i and document j and is defined according to VSM as follows:

$$Sim(d_i, d_j) = \frac{\sum_{k=1}^{n} \mathscr{W}_{ik} \times \mathscr{W}_{jk}}{\sqrt{\sum_{k=1}^{n} \mathscr{W}_{ik}^2} \times \sqrt{\sum_{k=1}^{n} \mathscr{W}_{jk}^2}} \tag{2.13}$$

where n is the dimension of feature vectors and $\mathscr{W}_{ik}$ is the weight of the feature k in document i calculated according to the TF-IDF algorithm. The smaller the cosine distance between the two texts is, the more similar they are.

- Then, each cluster centroid is moved to the mean of the points assigned to it. This step is repeated until all data points are assigned to one of the K clusters.

3. **TDT Evaluation:**
If a topic could not be detected by the system, the resulting error is called ”missed detection error”, but if a system detects a wrong topic, the error is called ”false alarm”[26]. The official evaluation measure of TDT is based on a cost function, which is a weighted combination of miss and false alarm rates [26]:

$$C_{Det} = C_{Miss} \cdot P_{Miss} \cdot P_{target} + C_{FA} \cdot P_{FA} \cdot P_{non-target} \tag{2.14}$$

P_{target} is the prior probability that the topic is correctly detected ($P_{non-target}$ = 1 - P_{target}). C_{Miss} and C_{FA} are user defined values that reflect the cost associated with miss and false alarm errors respectively and P_{Miss} and P_{FA} are the probabilities of a Miss and a False Alarm. The cost parameters used in the evaluation are: P_{target}= 0.02, C_{Miss} = 1.0, C_{FA}=0.1. The C_{Det} value is normalized as follows:

$$(C_{Det})_{Norm} = C_{Det} / \min(C_{Miss} \cdot P_{target}, C_{FA} \cdot P_{non-target}) \tag{2.15}$$

The authors defined the $S\_Tracking\_P$ metric to test topic tracking overall performance. This metric is calculated as follows:

$$S\_Tracking\_P = \frac{\sum_{i=1}^{m} ((C_{Det})_{Norm})_i}{m} \tag{2.16}$$

where m is the number of topics, and $((C_{Det})_{Norm})_i$ is TDT evaluation scores in topic i.

2.1.4 Graph based approaches

A graph approach for topic identification called LIGA and two extensions that enhance its performance were introduced in [34]. The tests were done on arabic datasets. The authors of [34] applied the LIGA approach, which was first introduced in [?] for language identification, to identify the topics from the documents.

Figure 2.4 shows the different steps of the LIGA approach.

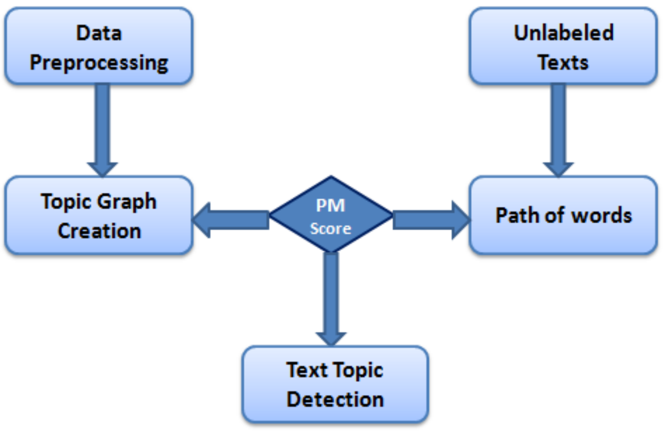


Figure 2.4: LIGA Approach workflow

First the training documents are preprocessed by eliminating the foreign words and the stop words and stemming the rest of words. Then the LIGA approach is applied. This approach can be divided into two steps: model construction and document classification.

- Model Construction:
  For each training document a graph, which corresponds to one topic called ”topic graph” will be constructed. In this graph, each node contains a word and its number of occurrences in the training document, and each edge represents the link between two consecutive words in the training documents, and the weight of the edge represents the number

of occurrences of these consecutive words.
Let’s take the example of the following sentence: ” *A sad movie is a movie that makes the viewer feel sad and cry* ”.
The corresponding topic graph is shown in Figure 2.5. After prepocessing the sentence,

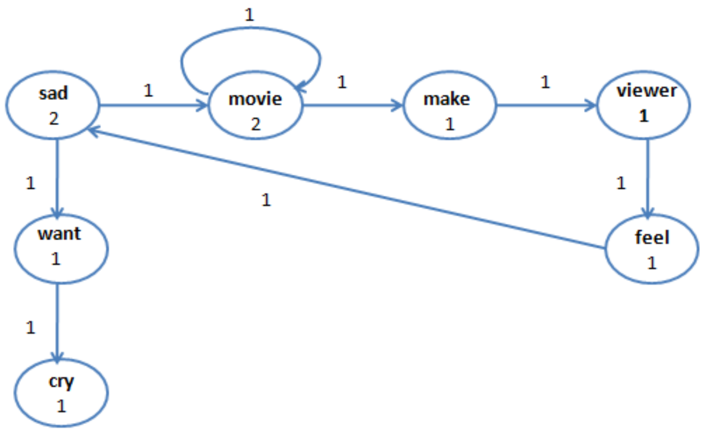


Figure 2.5: Topic graph example

the topic graph would be:

- Document Classification:
The unlabeled texts are presented as a path of words, and the entire paths are matched onto the graphs and the similarity between them is computed using the Path Matching (PM) score. The PM score assigns an integer to each topic. All PM scores of all topics are initially initialized to 0.
To compute the PM score, the path is traversed node by node. If a path node exists in a topic graph G_i , then its weight is added to the PM score and if a path edge exists in the topic graph G_i then its weight is also added to the PM score. The text topic corresponds to the topic graph having the highest PM score.
Let’s take the example of the following sentence: ” *Titanic is one of the best sad movies ever made*.” This sentence corresponds to the path of words presented in 2.6:
The words ”sad”, ”movie” and ”make” existed in the topic graph shown in 2.5 also the

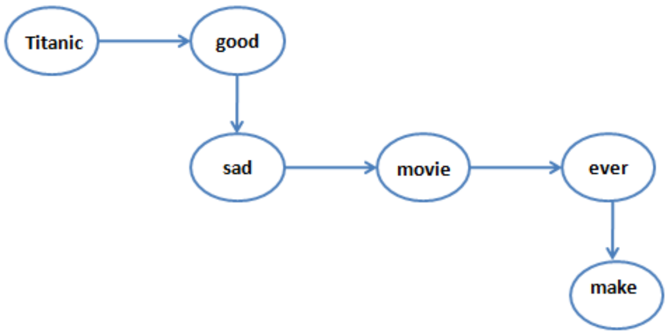


Figure 2.6: Word Path example

edge (sad,movie) existed in this topic graph. Thus $PM = weight(sad) + weight(movie) + weight(make) + weight((sad,movie)) = 5$

Another graph-based approache was introduced in [7] which applies the Page-Rank algorithm on Wikipedia derived graph to decide the importance of a vertex within a graph.

2.2 Concept Detection in Image Datasets

A search engine that allows users to pose natural language queries, interpretes them and retrieves the corresponding images called GOOSE was presented in [12]. The user query is semantically interpreted, concepts are detected from the images and learned by matching these detected concept against the user query which he can refine. The user can also re-rank the results. Figure **??** gives an overview of the system components which will be further explained.

◇ Semantic Query Interpretation:
The user query is first lexically analysed using the Stanford Typed Dependency parser [**?**] which transforms it into a direct graph where nodes are the words and the labeled edges depict the grammatical relations between the words. Let's for example take the sentence: ”*Find a brown animal in front of a red Mercedes*”, the resulting directed graph from depicted in Figure 2.7, where the grammatical relations: dobj, amod, det stand respectively for direct object, adjectival modifier and determiner.

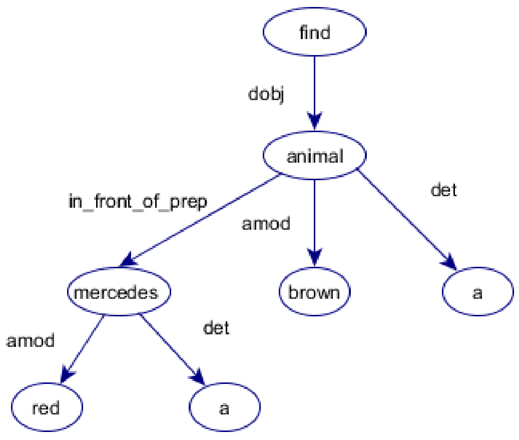


Figure 2.7: Lexical graph

This lexical graph is then semantically interpreted by applying a set of rules to transform the lexical graph elements into objects, attributes, actions, scenes and relations, according to the semantic meta-model depicted in 2.8 which the authors defined. This semantic Meta-model distinguishes objects that might bear attributes, take part in actions, occur in a scene and have relations with other objects. After applying the semantic interpretation on the sentence: ”*Find a brown animal in front of a red Mercedes*”, we obtain the graph shown in Figure 2.9, where the cardinality 1 is derived from the determiner ”the”, which is in a singular form. The attribute ”color” is derived from the adjectival modifier. This semantic graph is matched against the detected image concepts. If there is no exact match, the semantic graph is expanded using ConceptNet [**?**], a large semantic graph representing common human knowledge and the way it is expressed in natural language. The words in ConceptNet are related through predicates such as: ”IsA” and ”Causes”.

These relations were selected to expand the semantic graph where the unknown concepts are matched against concepts from ConceptNet, if a match is found, the matching concepts and their corresponding 'IsA' and 'Causes' relations are imported to the semantic graph. Otherwise, the expansion cycle goes through a second iteration. The expanded semantic graph of the user query is shown in Figure 2.10 where the green nodes are the concepts that were recognized before the expansion, the orange nodes are the concept nodes corresponding to expansions with ConceptNet and red nodes are the still unknown concepts after the expansion. An example of the colored semantic graph for the query Find a brown animal in front of a red mercedes is shown in Figure 2.10. This graph corresponds to the structured query that the user can further refine.

◇ Concept Learning and Concept Detection:
The system automatically recognizes and localizes concepts in video using the R-CNN method, which stands for Region proposals with Convolutional Neural Networks, presented in [9].
This method consists of generating region proposals, extracting a feature vector from each region and classify each region using SVM. In the following, we will go through each of those steps:
- Region Proposals generation:
around 2000 bottom-upt category-independent region proposals were extracted using the selective search method. [**?**, **?**, **?**, **?**, **?**]

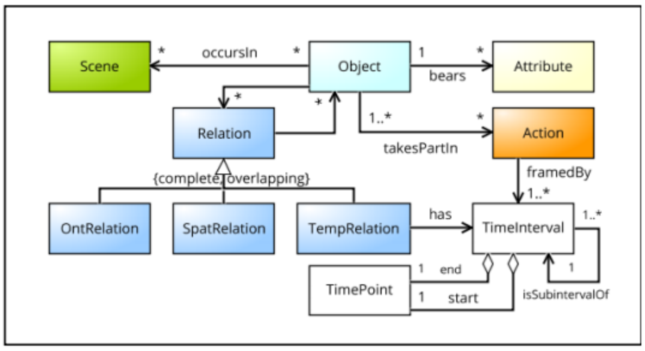


Figure 2.8: Semantic Meta-model

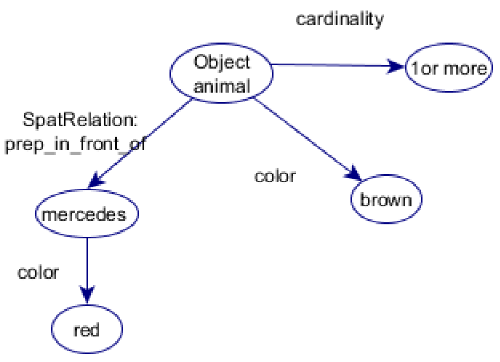


Figure 2.9: Interpreted Graph

- Features extraction: a 4096-dimensional feature vector from each region proposal using a Convolutional Neural Network. CNN are ..
- Regions classification: SVM ..

Finally the semantic graph is matched against the concepts that can be detected using the R-CNN algorithm. If a match exists, then the corresponding nodes in the graph are colored in green. Otherwise, the semantic graph is expanded using ConceptNet, a large knowledge base constructed by combining multiple web sources such as DBpedia, Wiktionary and WordNet.
The authors of [28] developed an algorithm for learning multiple consecutive tasks or Multi-Task Learning algorithm for video concept detection.
This algorithm exentded the Efficient Lifelong Learning (ELLA) Algorithm described in [13]. ELLA learns and maintains a shared basis for all task models. For each new task, the algorithm transfers knowledge through the shared basis to learn the new model, and refines the basis with knowledge from the new task to maximize the performance. The proposed algorithm was referred to as Efficient Lifelong Learning Algorithm with Label Constraint, it extends the ELLA algorithm by solving the objective function of ELLA using quadratic programming, adding a new label-based constraint that incorporates statistical information of pairwise correlations between concepts and instantiating ELLA with two base classifiers linear Support Vector Machines(SVM) and Logistic regression (LR).

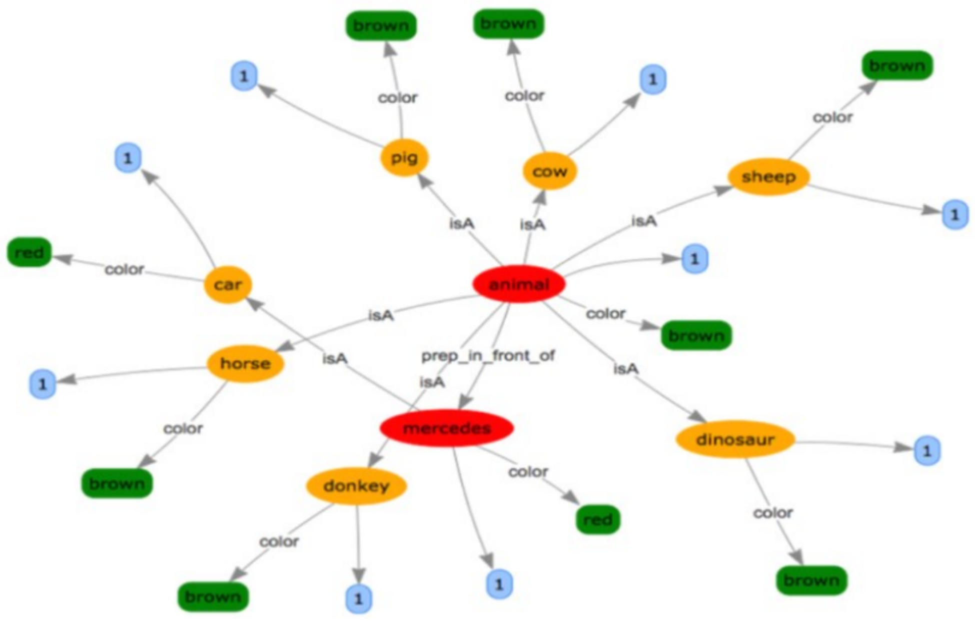


Figure 2.10: Expanded Semantic Graph of the query

3
Contribution

After going through the background and the state of the art, in this section we will present our novel graph-based approach to detect the concept out of highly-linked large datasets. In our model, the text datasets are first prepocessed then the training datasets are fed to our learning algorithm and finally the model is tested on new unlabled texts. In the remainder of this section, we will go through each of those steps.

3.1 Preprocessing

This step consists of preparing the datasets for further analysis. The words in the datasets are first extracted , stop words are removed, the rest of words are stemmed and lemmatised. Stemming is the process of replacing a word by its root independetly of the context of the word, for example, the words ”connection”, ”connected” and ”connectivity” will all be replaced by ”connect” [**?**]. Lemmatisation consists also of determining the base form of a word called lemma depending on its context. For instance the lemma of ”better” is ”good” which won't be determined using the stemming technique [**?**]. Both stemming and lemmatisation are word normalization techniques that will help us determine the occurence of the word and all its derivatives in the dataset. Let's take the following sentence:
It is no secret that fans of Chinese food often find it addictive. S1
After eliminating the stop words, stemming and lemmatizing the sentence, we obtain the following list of words:
secret, fan, chines, food, find, addict

3.2 Approach

3.2.1 Model construction and learning

A training graph having as vertices the set of preprocessed words in the dataset and three kinds of edges: ”follow”, ”child” and ”implication” is construced. For each sentence, the follow edges are built between consecutive words in each sentence. The child edges depict the grammatical

relationships between the words. Each sentence is then presented to a person that enters the list of concepts he understands from it. An implication edge between each word of the sentence and each understood concept is added to the graph. Once the training graph is constructed, the clustering algorithm ”Label Propagation” is applied on it. Each obtained cluster is colored with a different color.
Let’s assume a user understands the concepts: ”food, fans” from sentence S1:
After lemmatizing and stemming those words, we obtain the following list of implications: fan, food.
As a result, we obtain the following training graph:

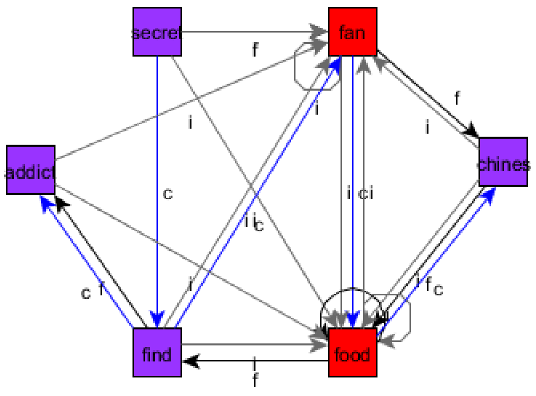


Figure 3.1: Training graph

The implication vertices are colored in red.

3.2.2 Model testing

After preprocessing the testing texts, a graph containing the ”follow” and ”child” edges is constructed in the same way. Then this graph is matched to the training graph, if there are some common nodes in both graphs, then the ”implication” edge between the node and its corresponding concept is copied from the training graph to the tesing graph. The induced implication vertices are colored in red.

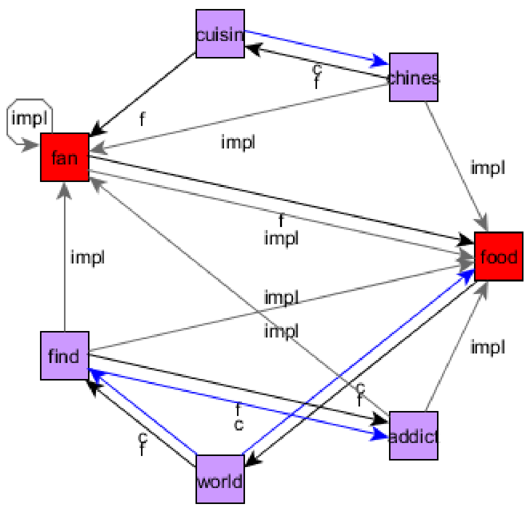


Figure 3.2: Testing graph

Bibliography

[1] Gartner it glossary.

[2] *An introduction to the big data landscape*. O'Reilly, 2012.

[3] *Understanding Big Data Analytics for Enterprose Class Hadoop and Streaming Data*. McGraw Hill, 2012.

[4] *Performance and Capacity Implications for Big Data*. IBM Red Paper, 2014.

[5] Tyler Akidau. The world beyond batch: Streaming 101, Aug 2015.

[6] G. Chowdhury. Natural language processing. *Annual Review of Information Science and Technology*, 2003.

[7] K. Coursey and R. Mihalcea. Topic identification using wikipedia graph centrality. In *Proceedings of NAACL HLT'2009, Boulder, Colorado*.

[8] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. 0070428077. The Press Syndicate at the University of Cambridge, March 2000.

[9] Ross Girshick; Jeff Donahue; Trevor Darrell and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[10] Ali Daud, Juanzi Li, Lizhu Zhou, and Faqir Muhammad. Knowledge discovery through directed probabilistic topic models: a survey. *Frontiers of Computer Science in China*, 4(2):280–301, 2010.

[11] Mark Steyvers2 David Newman1, Padhraic Smyth1. Scalable parallel topic models. *Journal of Intelligence Community Research and Development*, 2006.

[12] Klamer Schutte; Henri Bouma; John Schavemaker; Laura Daniele; Maya Sappelli; Gijs Koot; Pieter Eendebak; George Azzopardi; Martijn Spitters; Maaike de Boer; Maarten Kruithof; Paul Brandt. Interactive detection of incrementally learned concepts in images with ranking and semantic query interpretation. In *13th International Workshop on Content-Based Multimedia Indexing (CBMI)*.

[13] E. Eaton and P. L. Ruvolo. Ella: An efficient lifelong learning algorithm. In *inProc. of the 30th Int. Conf. on Machine Learning (ICML-13)*.

[14] Aurélien Gé.

[15] K. Gondow. Homepage for an educational operating system udos.

[16] Masaru Ohba; Katsuhiko Gondow. Toward mining concept keywords from identifiers in large software projects. In *Proceeding MSR '05 Proceedings of the 2005 international workshop on Mining software repositories*, 2005.

[17] Spark Programming Guide. viewed on:, 01 2017.

[18] Patrick Wendell Matei Zaharia Holden Karau, Andy Konwinski. *Learning Spark Lightning-Fast Big Data Analysis*. 9781449358624. O'Reilly Media, January 2015.

[19] Fang Huang. *Using Language Models and Topic Models for XML Retrieval*, pages 94–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[20] Fern Halper Marcia Kaufman Judith Hurwitz, Alan Nugent. *Big Data For Dummies*. 9781118504222. John Wiley and Sons, Inc., April 2013.

[21] Mostafa M. Hassan; Fakhri Karray; Mohamed S. Kamel. Automatic document topic identification using wikipedia hierarchical ontology. In *The 11th International Conference on Information Sciences, Signal Processing and their Applications*, 2012.

[22] A. Lakhdari, A. Chorana, H. Cherroun, and A. Rezgui. A link strength based label propagation algorithm for community detection. In *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, pages 362–369, Oct 2016.

[23] Victor Lavrenko and Charles Sutton. Kmeans clustering, 2011.

[24] Tom M. Mitchell. *Machine Learning*. 0070428077. McGraw-Hill Science / Engineering / Math, March 1997.

[25] Andrew W. Moore. Support vector machines, Nov 2001.

[26] NIST. *The 2004 Topic Detection and Tracking (TDT2004) Task Definition and Evaluation Plan*, 2004.

[27] Sonal Jain; Jyoti Pareek. Automatic topic(s) identification from learning material: An ontological approach. In *Second International Conference on Computer Engineering and Applications*, 2010.

[28] Foteini Markatopoulou; Vasileios Mezaris; Ioannis Patras. Online multi-task learning for semantic concept detection in video. In *IEEE International Conference on Image Processing (ICIP).*

[29] Richard Benjamins Rudi Studer and Dieter Fensel. Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 1989.

[30] Clementty Salton. On the construction of effective vocabularies for information retrieval[c]. In *Proceedings of the 1973 Meeting on Programming Languages and Information Retrieval.*

[31] G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[32] Hari Rajaram Sameer Wadkar. *Flink in Action*. 978-1617293924. Manning Publications, September 2016.

[33] Sean Owen Josh Wills Sandy Ryza, Uri Laserson. *Advanced Analytics with Spark Patterns for learning from Data at scale*. 978-1-491-91276-8. O'Reilly Media, April 2015.

[34] Kheireddine Abainia; Siham Ouamour; Halim Sayoud. Topic identification of noisy arabic texts using graph approaches. In *26th International Workshop on Database and Expert Systems Applications (DEXA).*

[35] Amazon Team. What is streaming data?, Jan 2017.

[36] Réka Albert Usha Nandini Raghavan and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E 76*, 2007.

[37] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.