CIRA: A COMPETITIVE INTELLIGENCE REFERENCE ARCHITECTURE FOR DYNAMIC SOLUTIONS

Marco Spruit[1] and Alex Cepoi[1]
[1] *Department of Information and Computer Sciences, Utrecht University, Princetonplein 5, Utrecht, the Netherlands*
m.r.spruit@uu.nl

Keywords: Competitive Intelligence, Web Intelligence, Reference Architecture, Text Analytics.

Abstract: Competitive Intelligence (CI) solutions are the key to enabling companies to stay on top of the changes in today's competitive environment. It may come as a surprise, then, that although Competitive Intelligence solutions already exist for a few decades, there is still little knowledge available regarding the implementation of an automated Competitive Intelligence solution. This research focuses on designing a Competitive Intelligence reference Architecture (CIRA) for dynamic systems. We start by collecting Key Intelligence Topics (KITs) and functional requirements based on an extensive literature review and expert interviews with companies and Competitive Intelligence professionals. Next, we design the architecture itself based on an attribute-driven design method. Then, a prototype is implemented for a company active in the maritime & offshore industry. Finally, the architecture is evaluated by industry experts and their suggestions are incorporated back in the artefact.

1 INTRODUCTION

The business environment has changed a lot during the last decades—constantly increasing competition due to globalisation, shorter product lifecycle, increased popularity of outsourcing as a means of cost reduction... These are just a few of the many reasons why companies nowadays need to exploit a lot more information than before about the competitive environment on which to base their strategic decisions (Zanasi, 2001). They need to perform thorough analyses before committing company resources towards a product which may not last long on the market, due to fierce competition or any other reason.

It is no wonder, then, that a lot of academic activity has been carried out over the years in the domain of Competitive Intelligence (CI), a field which focuses on monitoring the competitive environment with the aim of providing actionable intelligence that will provide a competitive edge to an organisation (Safarnia et al., 2011).

The large growth in the size of unstructured content created by consumers as well as the increasing availability of traditional media on the Internet have brought the necessity of quantitative analysis of this information via Competitive Intelligence Capturing Systems (Ziegler, 2012), tasked with collecting and performing analyses in an automated manner. A study performed back in 1998 shows that 90% of all information needed by a company to make informed critical decisions was already available on the Internet (Teo and Choo, 2001), but Oder (2001) noted that the software industry was "a long way from delivering a satisfying business or competitive intelligence solution".

Some implementation suggestions have been proposed over the years (*e.g.* Zhao and Jin, 2010), but they have not been validated in practice by a prototype or cross-analysed for similarity. To the authors' knowledge there is no in-depth analysis of what technological solutions should be used to meet the different requirements of the CI solution components, or how one would architect a CI solution based on a set of requirements. With such a big gap in literature, it is of little wonder that the development of competitive intelligence systems is still an ongoing endeavour in most enterprises (Zhao and Jin, 2011).

Therefore, our main research objective is *to develop a technical reference architecture for a dynamic competitive intelligence system, which can be easily customisable depending on a specific set of requirements.*

2 RESEARCH METHOD

Our research method starts with a structured literature review adopting a snowball method. We start from two searches for competitive intelligence, one including all papers and one selecting only on papers newer than 2009. We reviewed the first 10 results of each query, and for each of them looking up relevant papers who cite or are cited by them (for a total of 58 papers).

By using the coding method in grounded theory, we extract the concepts describing KITs, solution requirements or techniques to be used in implementation and label them. This process is iterative, as labels are continually refined and duplicates are merged.

In order to make sure these are still aligned with recent developments and in order to better explore the state of the art, we performed expert interviews with two companies involved in implementing such a system and CI professionals. A total of 6 interviews were conducted with 2 companies and 3 CI professionals.

We construct a Competitive Intelligence Reference Architecture (CIRA) by following the attribute-driven design method proposed by Bass et al. (2012). This involves an iterative process which decomposes the architecture into parts. Each part is designed and tested individually based on functional requirements and quality attribute requirements.

Evaluation is performed using a mixed method: a single use-case analysis and expert evaluations which commented on the design and gave feedback for further improvements. The use case involves implementing a prototype for a CI solution for a company active in the maritime & offshore industry and assessing how well it performs. The architecture model is then assessed by an expert panel of 3 professionals with experience in system design and architecture. Their suggestions for improvement were incorporated into the final artefacts.

3 LITERATURE REVIEW

There is a large consensus among publications regarding the three major phases that should make up any competitive intelligence solution: (i) collection and storage, (ii) analysis and (iii) dissemination. Data collection will be the phase responsible for extracting data from internal or external sources into a collection of entities we refer to as posts. A post is an abstract article or unitary text snippet (*e.g.* Spruit and Vlug, 2015) together with all its metadata (*e.g.* date, author, source, user comments). The analysis phase will be running various algorithms to create CI insights, which will then be disseminated to the end-users. Note that these are also compatible with CRISP-DM based processes (*e.g.* Otten and Spruit, 2011).

Table 1: Competitive Intelligence solution requirements identified in literature.

CI Phase	***#***	***Functional Requirement***	***#***	***Sources (e.g.)***
Extraction	1	Track changes in posts	4	(Chen et al., 2002)
	2	Collect both structured and unstructed data	2	(Rao, 2003)
Analysis	3	Credibility analysis	7	(Zhao and Jin, 2011)
	4	Topic exploration	5	(Ziegler, 2012)
	5	Document summarisation	2	(Bose, 2008)
	6	Scenario simulation	2	(Bose, 2008)
	7	Event deduplication	2	(Chen et al., 2002)
	8	Integrate structured and unstructured data	2	(Fan et al., 2006b)
	9	Product comparison	1	(Xu et al., 2011)
	10	Predict competitor data	1	(Cobb, 2003)
	11	Company, people and markets profiles	1	(Bose, 2008)
	12	Multilingual support	1	(Chen et al., 2002)
Dissemination	12	Ad-hoc querying	8	(Mikroyannidis et al., 2006)
	14	Monitoring & alerts	5	(Fan et al., 2006b)
	15	Personalised information routing	5	(Fan et al., 2006a)
	16	Information visualisation	5	(Rao, 2003)
	17	Newsletter summaries	1	(Prescott, 1995)
	18	Security policies	1	(Bose, 2008)

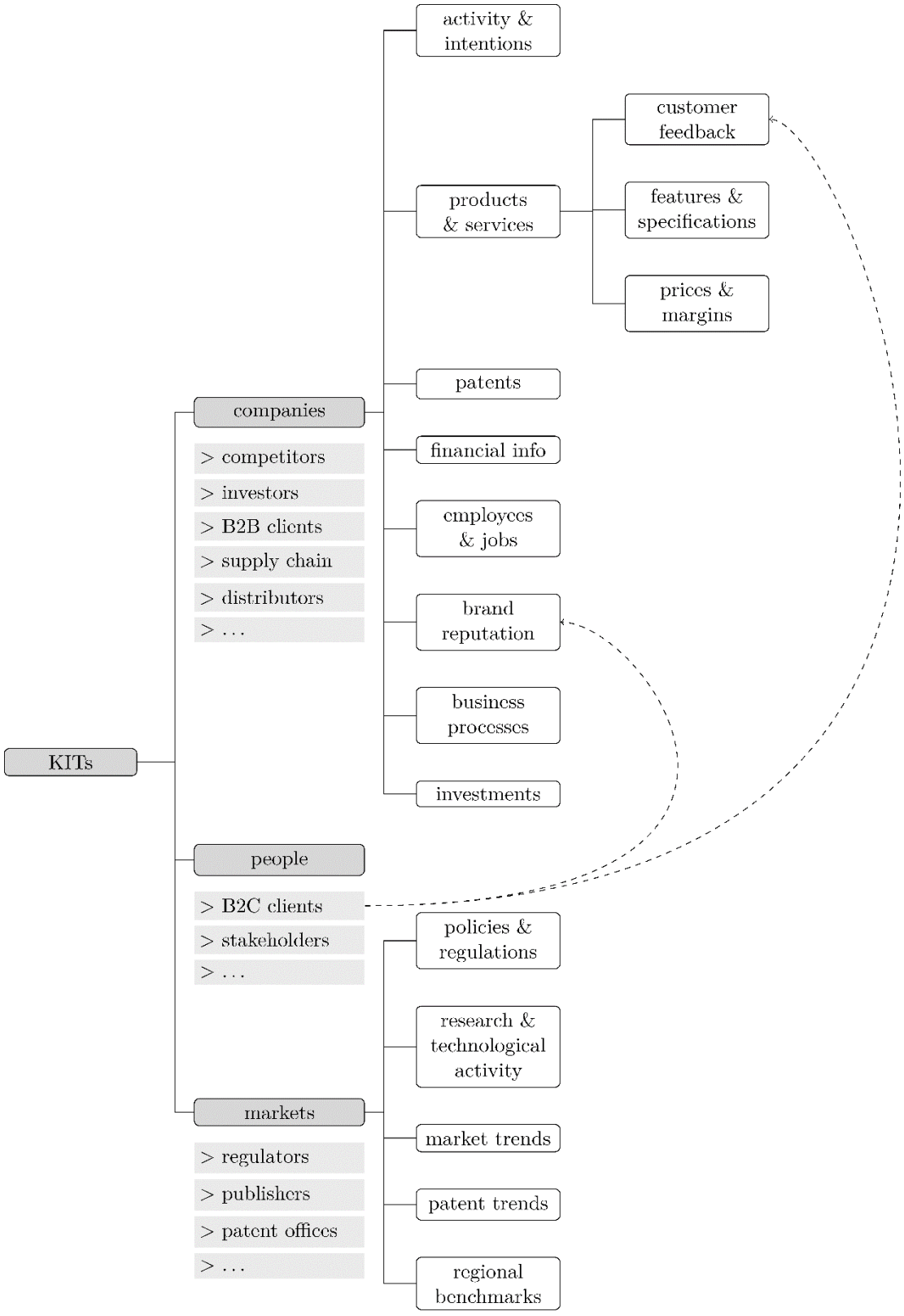

Figure 1: The Key Intelligence Topics (KITs) model.

3.1 Key intelligence topics and solution requirements

Our literature review resulted in identifying the following Key Intelligence Topics (KITs) as depicted in Figure 1. We identify three main types of entities we would like to extract intelligence about: companies (including competitors, companies in the supply chain, and B2B clients), people (including B2C customers, key employees, and stakeholders) and markets (inclusing research activity & trends, regulators, and policy makers).

The next step involves identifying functional requirements for a CI solution. While the list is not meant to be a complete reference, we found that we did not uncover any new requirements during the interviews process. We classified the functional requirements into the three main phases for a better overview in Table 1.

3.2 Architectural Fragments

In the three phases proposed by Bernhardt (1994), we consider two intermediary steps between data collection and analysis: pre-processing (the elimination of clutter from data sources, such as ads and navigational elements) and post-processing (including annotating content with part of speech, and named entities).

We have identified the architectural fragments for CI implementations presented throughout the literature. Zhao and Jin (2011) proposes an architecture reliant on the extraction of entities and relationships using Information Extraction techniques and modelling them onto an ontology. Using a rule-based technique, intelligence insights would be generated and be subject to a credibility analysis process. While not going into detail on how this process would work, he proposes a social-network-based evaluation model, where the credibility of each insight would depend on the credibility of the facts and the sources from where these facts have been extracted.

Ziegler (2012) proposes a similar architecture, but proposes using a plain storage for posts. Indexes would be build by preprocessing the content and weighing the resulting terms while unsynchronized annotators would generate annotation data stored in a separate storage system.

Dai (2013) identifies two main types of analysis which can be performed: opinion mining (which extracts general perception from users comments) and event detection (which clusters posts reporting the same events). Using the systems mentioned above as building blocks, Dai (2013) proposes a high-level CI solution which she calls Data Analysis and Visualisation AId for Decision Making (DAVID).

Liu et al. (2009) proposes a solution to be used for trend analysis, called event change detection. The system analyses differences in the conditional and consequent parts of association rules in order to detect emerging patterns, unexpected changes patterns and added/perished rules.

To analyse the similarities and differences between these fragments, we compare the 5 fragments. First we extract the activities for presented in each of the fragments and then we develop a comparison table by specifying all activities and comparing them across all the other fragments. We mark the presence of each activity with either ✓ (if they are present), × (if they are absent), n/a if they are not applicable to the current fragment (Wei and Lee (2004) and Hu and Liu (2004) are but subfragments of a complete CI solution) or a custom string (which means they are present under a different name or are of a specific type). The comparison table is present in Table 2.

Table 2: Comparative analysis of Competitive Intelligence fragments identified in literature.

Action	***Fragment***	***Zhao & Jin (2011)***	***Ziegler (2012)***	***Wei & Lee (2004)***	***Hu & Liu (2004)***	***Dai (2013)***
Preprocessing	Detect Content	×	DOM-based learning	n/a	n/a	n/a
	Post Storage	×	✓	n/a	✓	✓
Postprocessing	POS Tagger	×	✓	×	✓	✓
	NER + ERD	✓	×	✓	×	✓
	Entity Storage	ontology	annotations	n/a	×	✓
Analysis	Event Deduplication	(implied)	×	✓	×	✓
	Credibility Evaluation	SN Model	×	n/a	×	×
	Opinion Mining	×	polarity	n/a	✓	(postprocessing)
	Trend Detection	×	co-occurrences	n/a	×	✓
	Results Storage	×	annotations	n/a	n/a	Knowledge Base
Dissemination	Adhoc Querying	✓	×	n/a	n/a	×

4 REFERENCE ARCHITECTURE

Experts deemed the list of solution requirements to be comprehensive during our interviews, as we did not uncover any new ones. They advised designing a modular customizable systems using an annotation based storage. They also suggest storing metadata and summary instead of fulltext and using a gazetteer approach to Named Entity Recognition (NER) since it generally yields more accurate results.

By combining the identified solution requirements, and design decisions identified in literature and interviews according to the Attribute-Driven Design method by Bass et al. (2012), we here propose the high-level Competitive Intelligence Reference Architecture (CIRA) presented in Figure 2.

4.1 Collection and Storage

We distinguish two types of components which handle the collection part of a CI solution: either a crawler (in case the input is an unstructured data source) or an Extract, Transform, Load (ETL) tool (for ingesting structured data). The crawlers and ETL tools would run independently of one another.

We define a crawler as a component which can execute actions and traverse any unstructured data source in a predefined way. A web crawler is probably the most well-know example, although a generic crawler should be able to extract data from other types of data sources (all sorts of data APIs, rdf, etc.).

A web crawler would be responsible with authenticating itself on the web source (if needed), following a predefined traversal logic and extracting from each post the metadata and content. There are, to the authors' knowledge, three ways of implementing the extraction part of the process: manually-provided selectors (*e.g.* XPath), template-based extraction (Zhai and Liu, 2005) or fully automated extraction (Ziegler, 2012).

The post items then enter through a pipeline of preprocessors which have the task of cleansing the data. Examples include stripping html tags, normalize dates, etc. Optionally, we can at this point filter out duplicate (or rather unchanged) post items by looking up a hash value in the database. The post items now enter a pipeline of postprocessors and filters before being finally passed to the storage middleware.

The role of postprocessors is to implement post-level analysis processes (NER could be a candidate here) during the collection phase. The role of the filters is to discard the posts which are not relevant for the user. One usage example is a NER

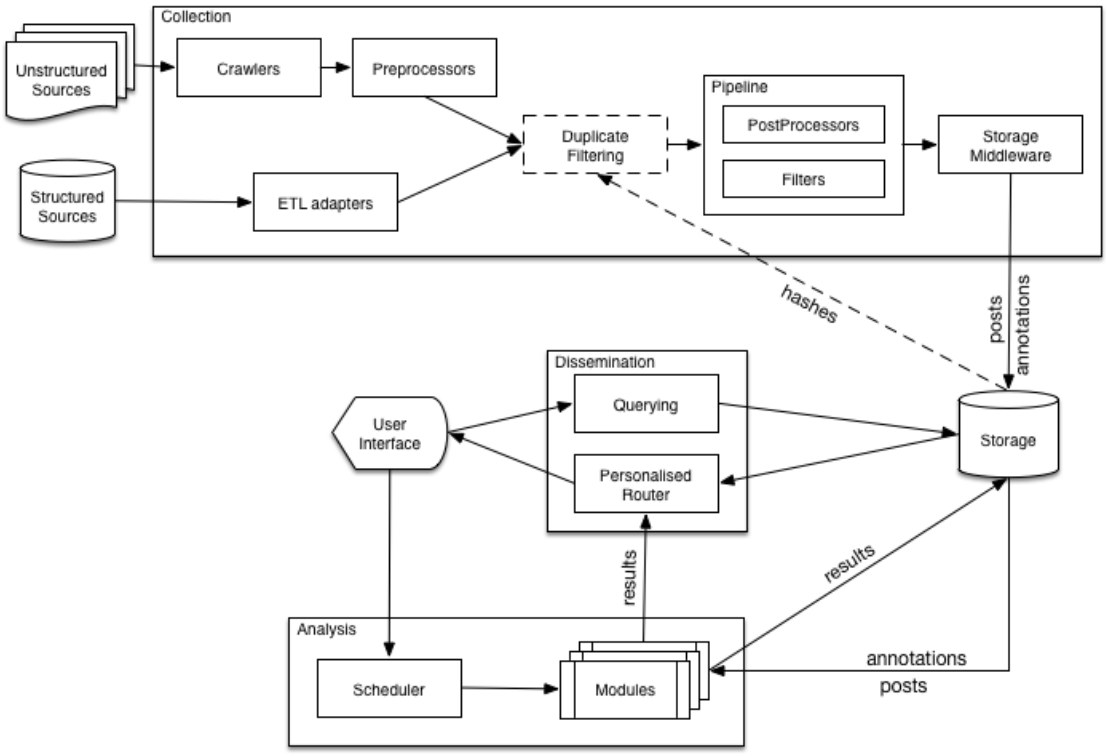

Figure 2: The Competitive Intelligence Reference Architecture (CIRA).

postprocessor and a filter which discards all posts in which no named entities have been detected.

While many types of storage engines may do the job, especially for smaller volumes, a column- or document-oriented database would work best, due to the flexibility in storage schema (personal communication, Company C1, 8 jan 2014).

In order to properly update existing data, an unique identifier needs to exist for each post. This unique key would be a combination of the source and an unique site_id (which can be chosen at discretion). In order to avoid data consistency issues when data importers and analysis modules run concurrently, we recommend storing newly crawled posts in separate snapshots and import them at the end of the collection process.

4.2 Analysis

The functional requirements of the analysis phase have been extracted in Table 1. Except for multilingual support, which involves adapting the analysis algorithms for other languages, each of the requirements refer to a distinct analysis module.

Any analysis module which operates at the post level (without information about the rest of the posts in the storage) can be implemented as a postprocessor module. Generally that is recommended as postprocessors greatly reduce data consistency issues.

It is common for analysis modules to have dependencies between themselves, so we need to figure out a way to schedule them in the correct order. For our usecase running the modules sequentially in a topological order (the most basic algorithm for task scheduling) should be a more than adequate solution, but other more sophisticated algorithms can be used as well.

One issue that was raised during the expert validation was preserving data consistency between analysis modules, since new posts can be imported while a dependency module is running and the next module would then find new posts for which the dependent module has not run. In order to cater for this the scheduler needs to index the items present before the chain of analysis modules is scheduled and use that subset for all modules it schedules. This can be achieved in an number of different ways, among which we mention creating a snapshot of post item identifiers and storing a timestamp and filtering based on it in each module.

4.3 Information Dissemination

We spent little time discussing information dissemination, partly because the topic is generic, not specific to a CI solution. During our literature review we identified the main methods of delivery to be ad-hoc queries, monitoring, alerts and digests.

The main methods of enhancing the results are personalised information routing and visualisation tech-niques. Fan et al. (2006a) performs a thorough comparison of the possible algorithms which can be used for information routing and evaluates their performance. Whatever the choice, the architecture remains the same: a personalised information router component which gets results either from the database or from the analysis modules and decides to which of the users to send them.

5 USE CASE ANALYSIS

In order to test the architecture created, we developed a prototype for a company active in the maritime & offshore industry. Their main requirements for a CI solution were to be able to quickly explore articles based on competitors, markets and areas as well as identify patterns and trends over the years in the large amount of available data.

When we started the use-case design, a commercial product for information collection was already being developed, so, in order to bootstrap the process and due to time constraints, we decided to use it as a part of our CI solution prototype process. While it does not perfectly match the architecture presented in the previous section, it serves as an example of how existing systems can be used as a “collateral” in developing a CI solution starting from a reference architecture (Bass et al., 2012).

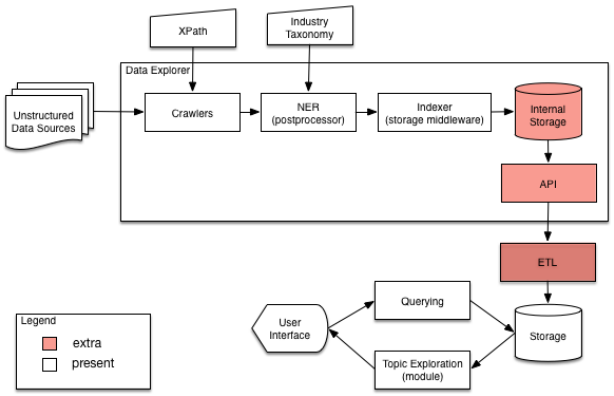

Figure 3: High-level Architecture of the prototype.

Table 3: Example post item in our data set.

```
{  "lang":          "english",
   "title":         "Brent falls under $106",
   "url":           "http://www.upstreamonline.com/live/article1355570.ece",
   "text":          "Oil fell under $106 a barrel on Wednesday to a six-week low as
                    concerns eased about [...]",
   "publish_date":  "2014/03/19",
   "annotations":
       { "areas":   ["Asia > Russia", "Commonwealth of Independent States > Ukraine"],
         "markets": ["Offshore Oil & Gas"],
         "orgs":    ["US Federal Reserve", "American Petroleum Institute",
                     "Energy Information Administration"]
         "people":  ["Christopher Bellew", "President Vladimir Putin",  "Brent"],
}      }
```

The technical architecture of this prototype is depicted in Figure 3. The main analysis components used were NER as a postprocessor and a Topic exploration module. Using NER, we were able to identify for each of the extracted posts:

- Organisations
  (using both a gazetteer and a pre-trained NER)
- People (using a pre-trained NER)
- Areas (using a gazetteer NER)
- Markets (using a gazetteer NER)

For the gazetteer NER, we made use of two taxonomies which were provided by the client, one for areas (`area > country`, *e.g.* `"Asia > Korea, South"`) and another for markets (`domain > market`, *e.g.* `"Seagoing Transport > LNG Tankers"`).

5.1 Topic Exploration

We adopt and modify the event change detection algorithm proposed by Liu et al. (2009) in order to exploit how association rules between tags evolve over time (identifying here trends as well as unexpected new/perished/changed rules).

The example post in Table 3 contains 9 tags (`["areas", "Asia > Russia"], ["areas", "Commonwealth of Independent States > Ukraine"], ["markets", "Offshore Oil & Gas"]`, …) grouped into 4 features ("areas", "markets", "orgs", "people").

The next step is what we call *tag promotions*, a process which consolidates the tags with the same meaning into one. Example promotions include acronyms (*OSE* becoming *Oslo Stock Exchange*) or named entities identified by the classifier being "promoted" to the named entity specified in the taxonomy (*Hyundai Heavy Industries* becoming *Hyundai*).

As part of topic exploration we first identify the top tags for each feature. Table 4 shows an example. For example, Table 4 lists the top organisations which are co-mentioned with Norway:

Table 4: The top organisations which are co-mentioned with Norway in the data set.

Arti-cles	*Quarter occurrences*	*Feature*	*Co-occurring organisation*
89	|++++++++|	["orgs",	"Hyundai"]
57	|++++++++|	["orgs",	"Daewoo"]
56	|++++++++|	["orgs",	"Samsung"]
34	|++++++++|	["orgs",	"Rolls-Royce"]
25	|++++++++|	["orgs",	"Bergen Group"]

At the beginning of each line in Table 4 is the total number of articles tagged with both Norway and that specific organisation, as well as a list of occurrences, *e.g.* `|+++ ++++|`. In this example, an occurrence was found in each of the quarters except Q4 2012; there are 8 quarters for 2012-2013 period, and each + signifies that there was at least one occurrence for that quarter.

Table 5: Two examples of association rules in the data set.

Confi-dence	*Occurrences*	*Rule*
0.84	|++++++++|	["markets", "Seagoing Transport > LNG Tanker"] → ["areas", "Asia"]
0.73	|++++++++|	["markets", "Seagoing Transport > LNG Tanker"] ["areas", "Asia"] → ["areas", "Asia > Korea, South"]

The second step is to divide all articles into quarters (3 calendar months periods) and generate rules using the association rule learning technique for

each of the quarters. To this extent we use a standard Apriori algorithm for sparse datasets (Agrawal and Srikant, 1994), which despite its simplicity is one of the best algorithms for association rule learning (Hipp et al., 2000). The output of the algorithm are rules in the form as shown in Table 5.

The first rule means that 84% (the confidence) of all articles tagged with LNG Tanker (LNG stands for liquified natural gas) will also be tagged with Asia and that 73% of all articles tagged with LNG Tanker and Asia are also tagged with South Korea. These two rules seems to suggest that South Korea is a leading authority in the LNG market, a fact which was later confirmed to us by the client.

However, the association rule learning is going to generate a lot of such rules (in our experiments a couple hundred per quarter), making it complicated to analyse them all and spot trends or changes in the marketplace. The final step is the event change detection algorithm which compares the rules mined from two successive quarters in order to identify new, perished or growing trends as well as unexpected changes. For each of these changes, Liu et al. (2009) propose a way to calculate the degree of change (also marked as `deg`) Below we present each type of changes with examples.

1. Emerging: Emerging rules, as shown in Table 6, are the ones that have been found also in the previous quarter and are seeing an increase in support in current quarter (we use a threshold of 20%, i.e. a ratio between the two values of 1.2 or larger).

Table 6: Rules for query "emerging 2013 Q3 => 2013 Q4".

Change	*Rule*
+1.05	["areas", "Asia > Russia"] → ["areas", "Arctic Region"]
+1.05	["areas", "Arctic Region"] → ["areas", "Asia > Russia"]
+0.78	["areas", "Asia > Russia"] → ["markets", "Offshore Oil & Gas"]

2. Added: Added rules, as shown in Table 7, are rules that have been found in this quarter but were not found in the previous one.

Table 7: Rule for query "added 2012 Q4 => 2013 Q1 (query: LNG Tanker)".

Change	*Rule*
+0.20	["markets", "Seagoing Transport > LNG Tanker"] ["areas", "Asia > Japan"] → ["areas", "Asia > Korea, South"]

3. Perished: Perished rules, as shown in Table 8, are the ones that have been found in previous quarters, but not in the current one.

Table 8: Perished rule for query "perished 2012 Q4 => 2013 Q1 (query: Shell)".

Change	*Rule*
+0.21	["markets", "Offshore Oil & Gas"] ["orgs", "Shell"] → ["areas", "Asia > Korea , South"]

4. Consequent change: Consequent change rules, as shown in Table 9, are unexpected changes in the consequent part of a rule. The rule found in the previous quarter is not found in the current one, but instead there is a rule with a very similar conditional part and a different consequent part.

Table 9: Consequent change rule for query "consequent change 2013 Q1 => 2013 Q2".

Change	*Rule*
+1.05	["areas", "Europe > Norway"] → ["areas", "Asia > China"] ➔ ["areas", "Europe > Norway"] → ["areas", "Asia > Korea , South"]

5. Condition change: Condition change rules, as shown in Table 10, are unexpected changes in the condition part of a rule. They operate similarly to the consequent changes.

Table 10: Condition change rule for query "condition change 2012 Q4 => 2013 Q1".

Change	*Rule*
+1.01	["markets", "Ship Repair"] → ["areas", "Asia > China"] ➔ ["markets", "Seagoing Transport > Container Vessel"] → ["areas", "Asia > China"]

Finally, we classify the most popular rules by averaging their confidence across quarters in Table 11. This shows the most important trends which have been happening for the entire period of two years. For example the most prominent trends identified with the entire dataset suggest Hong Kong's interest in search and rescue vessel markets as well as Daewoo and Hyundai doing business together in South Korea.

5.2 Client Feedback

Due to KDIR page constraints we refer to Cepoi (2014) for the in-depth evaluation of our prototype.

6 DISCUSSION

During the research process we discovered many of its modules to be research topics in their own right. To cater for this variability we limited ourselves to a higher level architecture which focuses more on the interaction and requirements of the analysis modules rather than their implementation details.

Table 11: The Top-3 association rules indicate the most important trends during the two year-period.

Confidence	*Occurrences*	*Rule*
1.0	|++++++++|	["areas", "Southeast Asia > Hong Kong (SAR)"] → ["markets", "Environmental Safety & Control > Search and Rescue Vessel"]
0.994	|++++++++|	["areas", "Southeast Asia > Hong Kong (SAR)"] → ["markets", "Defence & Security > Search and Rescue Vessel"]
0.946	|++++++++|	["orgs", "Daewoo"] ["orgs", "Hyundai"] → ["areas", "Asia > Korea , South"]

We argue that our reference architecture is more flexible than the framework proposed by Zhao and Jin (2011), which assumes a CI solution only focusing on basic rule-based intelligence generation, while catering equally well for that scenario. It is also more descriptive than the architectures from Ziegler (2012) or Dai (2013), which specify little more than the possible analysis modules.

The architecture has been validated both by industry experts and by being instantiated in a prototype, which delivered results evaluated by the customer to be accurate. We believe this research will prove useful for future implementation projects.

REFERENCES

Agrawal, R and R Srikant (1994). *Fast algorithms for mining association rules*. VLDB, 487-499, Santiago.

Bass, L. et al. (2012). *Software architecture in practice*. Addison-Wesley.

Bernhardt, D. (1994). 'I want it fast, factual, actionable'—tailoring competitive intelligence to executives' needs. *Long Range Planning, 27*(1), 12–24.

Bose, R. (2008). Competitive intelligence process and tools for intelligence analysis. *Industrial Management & Data Systems 108*(4), 510–528.

Cepoi, A. (2014). *A Reference Architecture for a Dynamic Competitive Intelligence Solution*. MSc thesis, Utrecht Univ. http://dspace.library.uu.nl/handle/1874/298561.

Chen, H. et al. (2002). CI Spider: a tool for competitive intelligence on the Web. *Decision Support Systems 34*(1), 1–17.

Cobb, P (2003). Competitive Intelligence through Data Mining. *Journal of competitive intelligence and management 1*(3), 80–89.

Dai, Y. (2013). *Designing Text Mining-Based Competitive Intelligence Systems.* Doctoral dissertation, UEF.

Fan, W. et al. (2006a). An integrated two-stage model for intelligent information routing. *Decision Support Systems 42*(1), 362–374.

Fan, W. et al. (2006b). Tapping the power of text mining. *Communications of the ACM 49*(9), 76–82.

Hipp, J. et al. (2000). Algorithms for association rule mining — a general survey and comparison. *ACM SIGKDD Explorations Newsletter 2*(1), 58–64.

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. 10th ACM SIGKDD international conference on Knowledge discovery and data mining.

Liu, D. et al. (2009). Mining the change of event trends for decision support in environmental scanning. *Expert Systems with Applications, 36*(2), 972–984.

Mikroyannidis, A. et al. (2006). *PARMENIDES: towards business intelligence discovery from web data*, Int. Conference on Web Intelligence, 1057–1060.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes 30*(1), 3–26.

Oder, N. (2001). The Competitive Intelligence Opportunity, 1–3.

Otten, S. and Spruit, M. (2011). *Linguistic engineering and its applicability to business intelligence: towards an integrated framework*. Int. Conf. on Knowledge Discovery and Information Retrieval, Paris, 460–464.

Prescott, J. (1995). The evolution of competitive intelligence. *International Review of Strategic Management 6*, 71–90.

Rao, R (2003). From unstructured data to actionable intelligence. *IT Professional 5*(6), 29–35.

Safarnia, H. et al. (2011). Review of Competitive Intelligence & Competitive Advantage in the Industrial Estates Companies in the Kerman City. *International Business and Management 2*(2), 47–61.

Spruit, M., and Vlug, B. (2015). Effective and Efficient Classification of Topically-Enriched Domain-Specific Text Snippets. *International Journal of Strategic Decision Sciences 6*(3), 1–17.

Teo, T. S. and W. Y. Choo (2001). Assessing the impact of using the Internet for competitive intelligence. *Information & management 39*(1), 67–83.

Wei, C.-P. and Y.-H. Lee (2004). Event detection from online news documents for supporting environmental scanning. *Decision Support Systems 36*(4), 385–401.

Xu, K et al. (2011). Mining comparative opinions from customer reviews for Competitive Intelligence. *Decision Support Systems 50*(4), 743–754.

Zanasi, A. (2001). Competitive intelligence through data mining public sources. *Competitive Intelligence Review 9*(1), 44–54.

Zhai, Y. and B. Liu (2005). *Extracting Web Data Using Instance-Based Learning*. Web Information Systems Engineering–WISE 2005, 318–331.

Zhao, J. and P. Jin (2010). Conceptual Modeling for Competitive Intelligence Hiding in the Internet. *Journal of Software 5*(4).

Zhao, J. and P. Jin (2011). Extraction and Credibility Evaluation of Web-based Competitive Intelligence. *Journal of Software 6*(8), 1513–1520.

Ziegler, C.-N. (2012). *Competitive Intelligence Capturing Systems.* Mining for Strategic Competitive Intelligence, 51–62.