Hybrid System for Collaborative Knowledge Traceability
An Application to Business Emails

Francois Rauscher, Nada Matta and Hassan Atifi
Institute ICD/Tech-CICO, Université de Technologie de Troyes,
12 rue Marie Curie, BP. 2060, 10010, Troyes Cedex, France

Keywords: Knowledge Management, Traceability, Problem Solving.

Abstract: In this paper we propose a Knowledge Traces Retrieval (KTR) system. It addresses the problem of retrieving elements of problem solving and design rationale inside business emails from a project. Even if knowledge management tools and practises are well spread in industry, they are rarely used for small projects. Our system aims at helping user retrieve traces of problem solving knowledge in large corpus of email from a past project. The framework and methodology is based on enhanced context (project data, user competencies and profiles), and use machine learning technics and ranking algorithm.

1 INTRODUCTION

Knowledge management became very popular in organizations in the 2000s after the works and discoveries of the past decades (Steels, 1993; Van Heijst et al., 1998). With the growth of the Internet in the 1990s, the dissemination and sharing of information has increased greatly. Many firms have the ambition to be able to apply this to knowledge. Companies have actually recognized the strategic value of knowledge as intangible capital (Grundstein, 1995) that could lead to competitive advantages.

However the path from Information management to Knowledge management is complex and involve the developments of systems that allow the firms to recognize, create, transform and distribute knowledge. The goal of these corporate memories is always the same: in one of the company's operations expertise has been created and/or used. This knowledge should not be lost because it can be used in similar areas for the company, and sometimes in different ones (by analogy or conceptualization), or in terms of traceability to determine by whom and why a decision was taken and what procedure has resulted. Information technology is an essential element to mobilize social capital for creation of knowledge, but how to proceed when the people involved in a project have left the company? This is especially true in software design projects. Software engineering is a quickly evolving, knowledge-intensive business involving many people working in different phases and activities (Rus et al, 2002). In this study, we focused on companies with software design and development projects involving geographically distributed teams, or remote workers. These companies have consulted us with identified needs in terms of traceability and knowledge reuse but with limited resources. The projects were completed (sometimes years ago), there were the deliverables/products, project management data (planning, documentation, and specifications), the list of participants and their skills, and their overall electronic exchange with related documents. Note that no recording (audio or video) of meetings or telephone conversations and videoconferencing were available.

A typical situation occurring was a manager asking to an employee: “What is the exact file format in this use case of the software XYZ and why?” And the only solution for the employee is to explore tens of gigabytes of emails requesting by keywords to try to find useful information. We present here a system called KTR that aims at helping users retrieving relevant (according to a query) traces of problem solving knowledge among a large emails corpus. Analysis method described in our previous work (Rauscher et al, 2002) suggested that taking into account the enhanced user context, organization, roles... can be used to find usable traces of problem solving knowledge.

The rest of this paper is organized as follows. In Section 2 related work is discussed briefly. Section 3 introduces our proposal then explains our approach and choices. Section 4 describes our system architecture and algorithm. A real life application is given in Section 5 and Section 6 comprehends our conclusions.

2 RELATED WORKS

2.1 Emails and Tasks

Emails is closer to written language (Baron, 1998), but in a computer mediated communication approach, as stated by Herring, et al. (2004), it can be considered as a good candidate for using discourse analysis techniques.

Collaborative Problem solving often leads to people assigning tasks to others. A study from Kalia, et al. (2013) on Enron corpus presents a method to identify and track tasks and commitments inside business emails. This was done by pure NLP technics like using n-grams, part of speech tagging and machine learning. Our system enhanced the context with user competencies and roles, project organization and phases. One can also notes the work of Scerri, et al. (2010) that propose a system called Semanta to assist users during their daily emails workflow to track actions items like Meeting Request, Task Assignment, and File Request.

Traceability of requirements in software development is usually done through the usage of a simple matrix but does not keep the record of the "how, why and who" during the design and implementation process. As our approach uses all the aspects of the project, phase, roles and not only textual content, it is closely related to traceability in the perspective of project memory.

2.2 Project Memory

Compare to corporate memory, project memory (PM) is a restricted part of a much larger capitalization exercise of a whole range of diverse experiences within the company. A project memory is generally defined as a representation of the experience gained during the implementation of projects (Dieng, et al., 1998). It describes the history of a project and the lessons learned during the lifetime of a project (Pomian, 1996). This memory must contain elements of experience from both the context of problem solving and its resolution. Project memory aims at traceability and reuse in similar project. Project memory used as materials the project data, the products of the project, stakeholders (roles, competences in the organization, exchanges and meetings, electronic communications, telephone) and related documents. However a reverse scale effect might appear compared to corporate memory: it is sometimes difficult to establish a method, software and collection of interviews for a project involving fewer than a dozen people. Tools and procedures exist for such tasks, but the size of the teams makes the systematic collection expensive and difficult. Especially when the project is finished and the team dismissed. In (Matta, et al., 2010) linguistic pragmatics was used on discussion forums to identify criteria that help analyzing messages of coordination in design project. KTR system will use context knowledge in a similar way.

3 TRACES OF KNOWLEDGE

As we stated in the introduction, the goal of our KTR system is to help the user retrieve "traces of knowledge". We define *Knowledge Traces* (KT), as messages containing meaningful information regarding the team's members having a problem-solving mediated exchange over email.

As a typical use, the user will input a query and the KTR system will present a list of messages from the email corpus matching the query and having a high score of being part of collaborative problem solving between the project's members. Retrieval process can be viewed as ranking or classification problem, in this study we consider it as ranking problem. We will compute a score on each message based on the user query and the KT elements. In order to decide if a message contains KT, we will check if:

- The message is dealing with topics from the project
- The message thread contains at least a request (problem statement)
- The messages in the same thread following the initial request contains elements of answer or a decision.

3.1 Topics

We called *topics* a lexicon of words regarding the project. This lexicon can be built from the following sources:

- Project phasing and specifications documents ;
- domain ontology if available
- an expert;

For instance we could decide that the lexicon will contains the topic XML (because it was an important milestone in the project) defined by a list of keywords (e.g: xml, tag, tree, xsd, dtd, schema, markup, structuration ...). When we will match any of the keywords in a message, we will know the message deals with the topic XML. The name of the topics in themselves are not relevant, there are simply a convenient way of handling them.

3.2 Problem Solving: The Request

As stated above, we chose first to focus on problem solving (Newell, et al., 1972) because useful knowledge for the business is most likely to be used or created during this kinds of exchanges. Especially on the undefined types of problem or "wicked problems" (Shum, 1997; Conklin and Weil, 1997), where collaborative knowledge occurs naturally. Hardin (2002) distinguish three main components in problem-solving: *givens* (facts and context), *goal* and *operations* (action to be performed).

In software development, when a team is trying to fulfill the specifications and requirements of the product or to correct a bug, it involves abstract and cognitive tasks. The first one being by fully qualifying the demand (the “goal”), then stating the problem to reach it. The “operations” will occur in the following exchanges when the team is going to face and solve the problem (see section 3.3 on the solutions). As the teams are using computer mediated communication, we have to carefully examine the requests in the email threads.

Finding requests requires interpretation of the intent that lies behind the language used. We chose to approach the problem as one of speech act detection. The theory of speech acts stems from the original works in philosophy of language from Austin (1975) and Searle (1969). Since then many studies have been conducted on speech acts (and particularly request), in different disciplines such as theoretical linguistics and natural language processing (NLP) with for instance works on automated speech acts identification in emails (Carvalho, et al., 2005; Lampert, et al.,2010). De Felice, et al. (2012) noted there is very little concern with data other than spoken language.

In the present study we narrowed our research to the analysis of the act of requesting in problem solving sequences. In pragmatic linguistic, request is part of the directive illocutionary acts. An illocutionary act being an act that is performed by saying it. The purpose of a request is to “get the hearer to do something in circumstances in which it is not obvious that he/she will perform the action in the normal course of events” (Searle, 1969). At first level we can we can distinguish between the request for saying (e.g. “could you me tell me”) and the request for a doing (“Could you do that”). These types of request are quite common among professional email exchanges.

However illocutionary acts have different enunciative modalities: There is not a one to one relationship between a speech act and its linguistic realization. As we show above a request can be performed linguistically in several ways: "do that", "could you do that?", and «I would like you to do that". A direct request may be imperative (order) or performative (like an assessment of obligation or need). An indirect request may question the capacity, the willingness, etc... of the hearer or give a suggestion. Often indirect requests are used in professional context because request in its inner nature is an FTA (Face Threatening Act) (Brown, et al., 1978; Goffman, 1959) that can be soften by politeness.

Our approach following resolutely a discourse analysis, we are working on statements that can be properly interpreted by taking into account both the discursive context (i.e. what constitutes the content of the email message) and context of “utterance” (that is to say, the situation of dialogue). The linguistic markers alone are not sufficient to indicate what is actually done in a computer mediated communication situation. As a side remark, it could be interesting in future studies to look at other types of speech acts (e.g. promissive, assertive...)

3.3 Solutions

When the system have detected a potential request regarding the projects topics, we would like to track the exchange between the team members to keep a trace of arguments, related matters, decisions and possible solutions to the initial problem. Our simple hypothesis is that collaborative knowledge is more likely to be created when some of people exchanging messages have the necessary competencies to solve the current problems.

Competency definition depends highly of the discipline (sociology, psychology, management) as stated in (Harzallah, et al., 2002; Vergnaud, et al., 2004). In the perspective of human resources, competencies are the measurable or observable knowledge, skills, abilities, and behaviors (KSAB) necessary to achieve job performance. One can distinguish between soft competencies (managerial and social interaction), hard competencies (functional and technical specific to a field (Tripathi and Agrawal, 2014). In our model, we will focus on technical competencies and their relations to the tasks that must be accomplished for the project.

4 KT-RETRIEVAL SYSTEM

In this section, we will go into further details explaining which kind of information is used and how the ranking for retrieval is calculated.

4.1 Overview

KTR follow a two-step approach, first indexing the messages, then ranking the relevant ones according to a user query. In order to do the indexing step, we first compute for each message a feature vector composed of a topic part, a request part, and a solution part. This gave us a KT score for each message. For the ranking part we use a linear combination of the KT score with the basic matching score between the user query and the message.

The overview of KTR system is illustrated in Figure 1.

4.2 Message Feature Vector

An email corpus is a set of messages ordered by time of arrival and grouped by threads (i.e. initial message with replies).

In a thread T, consisting of messages $(Mi)_{i\in T}$, each message M as an emitter E_M and receivers $R_M=(TO_M,CC_M)$ (respectively direct receivers (TO) and carbon copy (CC)).

As in Carvalho and Cohen (2006) we preprocess the messages to make them suitable for parsing and extracting features. For message in the same thread we remove the duplicate part in case of reply or forward (e.g. quoted reply content) and the signature or disclaimer part. The remaining parts are: the subject of the email, the sanitized body, the name of attachments.

4.2.1 Topics Part

Using existing context knowledge (project phases and specifications), we build the topics lexicon. This lexicon is voluntarily kept simple and have the form $L = (t_i)_{0\leq i<t}$, where t is the number of topics:
Topic t_i : $keyword_{i1}$, $keyword_{i2}$... $keyword_{ip}$.
Keywords chosen in topics shall not overlap too much to keep the results significant.

The content of all messages are represented by Vector Space Model (Salton and Buckley, 1988), i.e. $M= (w_i)_{0\leq i<k}$, in which each term is weighted by its *tfidf* (term frequency, inverse document frequency) score, k being the size of the vocabulary. The same is done for topics. We then compute a ranking between our messages and each topics using a cosine similarity based algorithm. This give us a topics matrix T where (T_{ij}) represents weight of topic j in message i. The score of topic part for a message M_i will be:

$$\text{Topic\_part}(M_i)=\frac{1}{t}\sum_{1\leq j\leq t} T_{ij} \qquad (1)$$

4.2.2 Request Part

The request detection is a well-known non trivial problem. We took a simple approach similar to Lampert, et al. (2010). However we worked at sentence level and if a request was detected in any of the sentences, the message was classified as request.

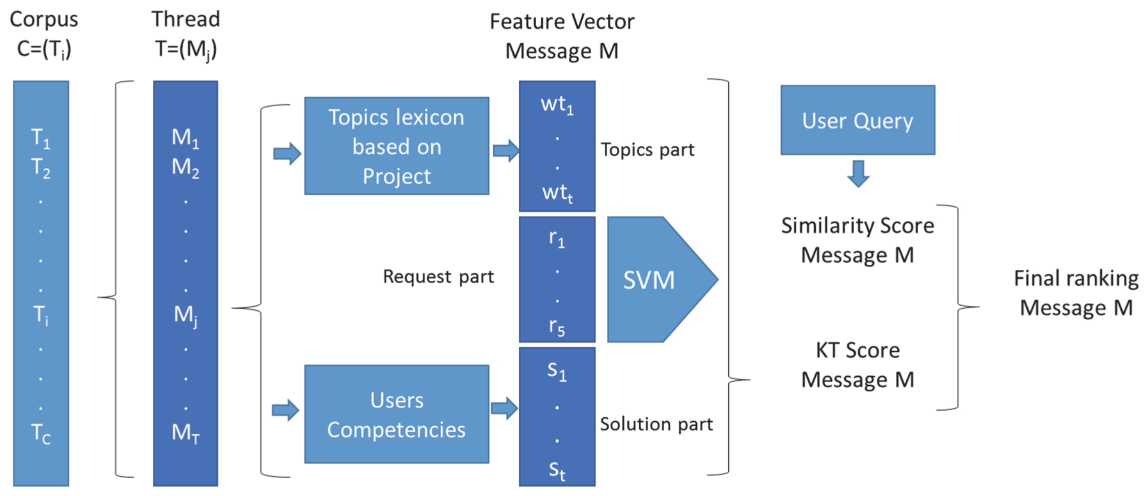

Figure 1: Overview of the KTR system and structure.

Sentence splitting was done according to punctuation and paragraph signs.

We chose customs parameters and an SVM classifier, implemented using Azure ML (Microsoft cloud machine learning platform). Given sentences of an email message as input, our binary request classifier predicts the presence or absence of request utterances within the sentence.

In order to do so, we establish with a linguist a set of custom features for each sentence. Presence or absence of: interrogation sign, specific bigrams and trigrams patterns based on pragmatic ("you should", "we must", "can you", etc.) and keywords ("question", "problem", "error", "who", "what", "how", etc.). We also add a temporal part by taking into account if a request sign was already present in previous messages from the same thread. When users are facing a problem, they are usually asking a lot of questions before reaching an agreement on a solution.

Another axis of analysis are the relationships of the project members. Roles in the organization are important to our study because they could help detect indirect requests. For instance, if a manager is writing to a developer "I would like (...)", it is for us the sign of an implicit request. Our model takes the roles into account using official function (hierarchical) or business relations (client/contractors). We built a "relationship" matrix R by using a weighted directed graph representing both hierarchical and client/contractors links. Users are vertices of the graph and the weights on edges bring a measure of user/user "influence" (real values between 0 and 1). R_{ij} stands for the "ordering capacity" from user i to user j. We use the max over to TO receivers.

The features we use in our SVM request classifier for a sentence are:

- Presence of verbal signs patterns (e.g., might, may, should, would, do you, etc..
- Presence of specific keywords
- Presence of interrogation mark
- Presence of previous request in same thread
- Influence score of emitter/receivers

It is important to note that this classifier is not project specific and once trained can be use in other projects.

Finally the score of the topic part is computed as in Equation (2), 1 if a request sentence was found in message and topic_part >0 (we discard requests not related to the project's topics), 0.5 if the message was part of a thread where a previous request was found, 0 elsewhere.

$$\text{Request\_part}(M_i) = \begin{cases} 1 & \text{if request} \wedge \text{topic} \\ 0.5 & \text{post request} \\ 0 & \text{not request} \end{cases} \quad (2)$$

4.2.3 Solution Part

We look for pieces of knowledge related to the requests founds in the previous step. For each thread T, we have identify messages M_r where a request is likely to occur. We will then examine all the following messages in the same thread i.e. $M_{s,\,T} = (M_i)_{(i>r),\,T}$. taking into account user competencies.

First we built a matrix CU representing user competencies, (using curriculum vitae and function description of their role in the project). CU = (CU_{ij}) representing the skill level of user i in competence j. We took similar approach as Vergnaud et al., (2004) to measures skills (0=not knowing, 0.25 =novice, 0.5=medium, 0.75=experienced, 1= expert).

Then we built a matrix CT representing the competencies needed to fulfill a topic (its associated tasks) for the project, CT = (CT_{ij}) representing the importance of competency i regarding the topics j. This matrix is built with experts in each topics, again with discrete weighting (ranging from 0=competency useless for the tasks, 0.25, 0.5, 0.75, 1= vital competency). We then construct the matrix UT = ${}^{t}CU.CT$ where (UT_{ij}) stands for a very rough estimation of the skills of user i regarding the topic j.

The matrix UT is normalized using Frobenius norm not to depend on number of competencies or users. We compute on each message $M_i \in M_{s,\,T}$, its solution score:

$$\text{Solution\_part}(M_i) = \frac{1}{t} \sum_{1 \le j \le t} (UT_{ij}.T_{ij}) \quad (3)$$

We are dealing messages with potential solution of the problem solving raised by request in M_r, we are trying to assert that the emitter have the necessary competencies to bring new knowledge regarding the current topics.

4.3 KT Score and Ranking

The KT Score is calculated using the previous sub scores on each message.

$$\text{KT\_Score}(Mi) = \begin{cases} \text{Topic\_part}(M_i) + \\ \text{Request\_part}(M_i) + \\ \text{Solution\_part}(M_i) \end{cases} \quad (4)$$

This score evaluates the relevance of message of being part of a problem solving trace in the project.

In order to do the final ranking for the retrieval step based on user query Q, we have to use also the basic cosine similarity being sim(M_i,Q) (in Vector Space Model with *tfidf* weighting) between user query and message M_i. This is to take into account the specific terms of the user query

Finally the global ranking r for a message M_i is a linear combination which is calculated as follow:

$$r(M_i) = \mu KT\_Score(Mi) + (1-\mu)\mathrm{sim}(M_i, Q) \quad (5)$$

Where μ ($o \leq \mu \leq 1$) is a combing parameter. If $\mu=0$, we reverse back to simple *tfdidf* similarity ranking like Lucene text-search engine (Gospodnetic, 2004)

4.4 Algorithm

The overall algorithm for indexing and ranking is described in Figure 2. The global ranking is giving a relevance score for each message. We output the messages in decreasing ranking order but for better understanding for the user we kept them grouped by threads of conversation.

```
Inputs: Project Data, Corpus, User competency
Output: KT_Score for message
1 Indexing:
2 Prepare Topic lexicon L from Project Data
2 Prepare Topic matrix T (section 4.2.1)
3-Prepare CU,CT and UT matrix (section 4.2.3)
4-Prepare Influence Role R matrix (section 4.2.2)
5-Train SVM for request sentences
6 For each thread T ∈ Corpus
7   For Each message M∈T
8     Compute Topic_part(M) (Equation (1))
9     Compute Request_part(M) (Equation (2)
10    Compute Solution_part(M) (Equation (3)
11    Compute KT_score(M) (Equation (4)
12  end
13 end
14 Retrieving:
Inputs: User Query Q, KT_Score, Corpus
Output: Global ranking and messages
15 For Each message M∈Corpus
16  Compute sim(Q,M) (section 4.3)
17  Compute rank(M) (Equation (5))
18 end
19 Output messages by decreasing rank grouped by thread.
```

Figure 2: KTR Algorithm.

5 EXPERIMENTS

A publishing group editing magazines and books had in 2009 a software design project that lasted 2 years. A remote software development company was hired to create a workflow tool for journalists and lawyers. Nearly all the communications during specification, implementation, tests and delivery were done through email. The corpus was collected 2 years after the end of the project.

5.1 Corpus and Project

The corpus represent 3080 messages/ 14987 sentences in 801 threads between 30 projects actors The team was split between the contractors and the development team. Among these, various roles and skills were present, but the main actors were:

- U1: Chief editing manager (skill: law and management, Role: Contractor);
- U2: Law Journalist (skill: law and management, Role: Contractor end-user);
- U3: Information System Manager (Skill: Information system, Role: Contractor);
- U4: Information System Project Manager (Skill: Information system, Role: Contractor Employee);
- U5: Information System Developer (Skill: Software Engineering, Role: Development manager).

5.2 Application Settings

A topic lexicon was built according to section 4.2.1 and the message topic T matrix was computed accordingly. We had 10 topics. Stemming was done to compute the term frequency and cosine similarity.

For request, we built a feature vector as described in section 4.2.2 for each sentence containing (pragmatic verb markers, specific keywords, interrogation mark, temporal part, max emitter influence over TO (direct) receivers). For the ranking algorithm, to compute the global ranking combination parameter μ was set to 0.4. In future works, we will evaluate the impact of this parameter. Finally the two matrices CU and CT defined in section 4.2.3 were computed. This operation is done once and valid for the whole project.

On Table 2 and 3, we can see for instance that the user U1 possess good competencies in law and some in InDesign, and that these competencies are important for the tasks in topics Code and Paper.

Table 1: Excerpt from topic lexicon.

Topic	Keywords
Code	Law, legifrance, insurance, chapter, article, annexes, labor code
Paper	Indesign, Xpress, print, template, styles, margin

Table 2: Excerpt Competency User matrix.

	U1	U2	U3	U4	U5	U6	U7
XML/XSL	0	0	0	0	1	1	0
C#	0	0	0	0	1	0	0
SQL	0	0	0,25	0	1	0	0
Architecture	0	0	0	0	1	0	0
Law Code	1	0,75	0	0	0	0	0,5
Law Writing	1	1	0	0	0	0	0
Indesign	0,25	0	0	0,25	0,5	0	1
HTML	0	0	0	0,25	1	0,25	0

Table 3: Excerpt Competency Topics matrix CT.

	XML	BDD	Workflow	Code	Paper
XML/XSL	1	0	0	0	0,5
C#	0,25	0	1	0	1
SQL	0	1	0,5	0	0
Architecture	0,5	0,5	0,5	0	0
Law code	0	0,25	0,5	1	0
Law writing	0	0	0	1	0
Indesign	0	0	0	0	1
HTML	0	0	0,5	0	0

5.3 Corpus Labelling

In order to evaluate KTR system, we had to label the corpus. At first this was done by a linguist looking for the presence of direct/indirect requests.

Then we had to label the messages where answers to these requests were explored during the problem solving. In that case the ground truth or gold standard is estimated from the subjective opinion of a small number of experts. At first a technical expert was asked to label messages that correspond to collaborative knowledge traces, then a manager that worked on this precise project 4 years ago did the same. The manual labelling is an expensive and time-consuming process. All the messages selected by both experts were considered as good candidate for containing solutions to problems.

This part was especially tedious and due to time constraint only 70% of the corpus (starting from beginning in emails date order) was annotated by these last two experts.

5.4 Evaluations and Results

To measure the performances, we decided to compare KTR with baseline cosine similarity (Lucene-like text search). This was done by setting μ parameter to zero. We forged 40 queries based on real life scenario (for instance "xml file format insurance law comment") and compared the results of the two methods by counting the numbers of results labelled by experts in the 20 first answers. The dimension of the feature vector was 25. We found 7% improvement over the keywords only search on theses queries. We also noticed that request part is bringing noise (giving high relevance to messages containing request but not always related with the keywords of the query) and fine tuning would be necessary.

6 CONCLUSION

In this article we proposed KTR system for analysing professional emails from a project in order to help users locate problem-solving traces. Our study aims at traceability in the context of project memory. In contrast to previous models of extracting meaningful elements (like tasks, or concepts, etc...) from emails by using pure NLP techniques, we emphasize on enhancing the context and incorporate pragmatics and organizational components (speech acts, competencies modelling, roles).

Numerical results shows that the task of detecting problem solving knowledge traces is very delicate and subjective. However this task is necessary and is accomplished by employee in everyday life. This results further indicate that interesting pieces of knowledge does not always came from an initial request, and that requests and answers are often interweaved creating noisy context. Still there are some issues regarding the granularity (sentence or message level) of the approach requiring future study to achieve a better understanding of patterns of exchanges during collaborative knowledge creation.

While this study has shown relatively interesting results, we are planning further research studies to investigate the possibility of improving the algorithmic part. First by enlarging the context to surrounding messages and relevant threads and taking into account attachment's content. Second by making a more precise match between topics of request and possible answers. It has become clear that taking only into account the textual content of email (and not the overall human context surrounding it) would lead to very limited results as detecting traces of collaborative knowledge. The proposed system will also be evaluated on scenario where the project is yet not completed and will involve the users. Ultimately, our work contributes to project memory: traceability and structuration of knowledge in daily work realization of project.

REFERENCES

Austin, J. L., 1975. *How to do things with words* (Vol. 367). Oxford university press.

Baron, N. S., 1998. Letters by phone or speech by other means: The linguistics of email. *In Language & Communication,* 18(2), pp. 133-170.

Brown, P., Levinson, S.C., 1978. *Universals in language usage: politeness phenomena. Question and politeness*, ed. by Esther N. Goody. Cambridge: Cambridge University Press.

Carvalho, V. R., and Cohen, W. W., 2005. On the collective classification of email speech acts. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* pp. 345-352.

Carvalho, V. R.., and Cohen, W. W., 2006. Improving email speech acts analysis via n-gram selection. In *Proceedings of the HLT-NAACL 2006 Workshop on Analyzing Conversations in Text and Speech* (pp. 35-41). Association for Computational Linguistics.

Conklin, E. J., & Weil, W., 1997. Wicked Problems: Naming the pain in organizations (White Paper): *Group Decision Support Systems*.

De Felice, R., Deane, P., 2012. Identifying speech acts in emails: Toward automated scoring of the TOEIC® email task. *ETS Research Report* No. RR-12-16, Princeton.

Dieng, R., Giboin, A., Amergé, C., Corby, O., Després, S., Alpay, L., and Lapalut, S, 1998. Building of a corporate memory for traffic-accident analysis. In *AI magazine*, *19*(4), p.81.

Goffman, E. 1959. *The presentation of self in everyday life*.

Gospodnetic, O., and Hatcher, E., 2004. *Lucene in Action*. Manning Publications.

Grundstein, M., 1995. La capitalisation des connaissances de l'entreprise, systèmes de production des connaissances. In *Acte du colloque de l 'Entreprise apprenante et les Sciences de la Complexité*, Aix-en-Provence.

Hardin, L. E., 2002. Problem Solving Concepts and Theories. In *Journal of Veterinary Medical Education*, 30(3), pp. 227-230.

Harzallah, M., Leclère, M., & Trichet, F., 2002. CommOnCV: modelling the competencies underlying a curriculum vitae. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering* pp. 65-71.

Herring, S. C., Barab, S., Kling, R., & Gray, J., 2004. An Approach to Researching Online Behavior. *Designing for virtual communities in the service of learning*, p. 338.

Kalia, A., Motahari Nezhad, H. R., Bartolini, C., & Singh, M., 2013. Identifying business tasks and commitments from email and chat conversations. In *tech. report*, HP Labs.

Lampert, A., Dale, R., and Paris, C. 2010. Detecting emails containing requests for action. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 984-992). Association for Computational Linguistics.

Matta N., Ermine, J-L., Aubertin, G., Trivin, J., 2002. *Knowledge Capitalization with a knowledge engineering approach: the MASK method. Knowledge Management and Organizational Memories*, Dieng-Kuntz R., Matta N. (Eds.), Kluwer Academic Publishers.

Matta, N.; Atifi, H.; Sediri, M.; Sagdal, M., 2010. "Analysis of Interactions on Coordination for Design Projects," In *Sixth International Conference on Signal-Image Technology and Internet-Based Systems (SITIS*), pp.344-347.

Newell, A., Simon, H. A., 1972. *Human Problem Solving*. New Jersey: Prentice-Hall, Inc.

Pomian, F., 1996. *Mémoire d'entreprise, techniques et outils de la gestion du savoir*. Sapientia,

Rauscher F., Matta N., Atifi H., 2014. Discovering Problem-Solving Knowledge in Business Emails - Traceability in Software Design Using Computer Mediated Communication in *Proceedings of KMIS 2014/IC3K*.

Rus, I., & Lindvall, M., 2002. Guest editors' introduction: Knowledge management in *software engineering. IEEE software*, (3), pp. 26-38.

Salton, G., and Buckley, C., 1988. Term-weighting approaches in Automatic text retrieval. *Information processing & management*, 24(5), pp. 513-523.

Searle, J. R., 1969. *Speech acts: An essay in the philosophy of language* (Vol. 626). Cambridge university press.

Scerri, S., Gossen, G., Davis B., and Handschuh S.; 2010. Classifying action items for semantic email. In *Proceedings of the 7th Conference on International Language Resources and Evaluation* (LREC).

Shum, S. B., 1997. Representing hard-to-formalise, contextualised, multidisciplinary, organisational knowledge. In *Proceedings of the AAAI Spring Symposium on Artificial Intelligence in Knowledge Management*.

Steels, L., 1993. Corporate knowledge management. In *Proceedings of ISMICK 1993*, Compiegne.

Tripathi K. and Agrawal M., 2014. Competency Based Management. In *Organizational Context: A Literature Review. In Global Journal of Finance and Management*. ISSN 0975-6477 Volume 6, Number 4 pp. 349-356.

Van Heijst, G., van der Spek, R., and Kruizinga, E., 1998. The lessons learned cycle. In *Information technology for knowledge management*. Springer Berlin Heidelberg. pp. 17-34.

Vergnaud, N., Harzallah, M., and Briand, H, 2004. Modèle de gestion intégrée des compétences et connaissances. EGC pp. 159-170.