

Degree Programme in Industrial Engineering and Management

Jamal Armel

# Augmenting Transactive Memory Systems in Virtual Teams by means of Natural Language Processing and Machine Learning

Master's Thesis

Helsinki, July 29, 2017

Supervisor: Professor Henri Schildt

---

**Author** Jamal Armel

---

**Title of thesis** Augmenting Transactive Memory Systems in Virtual Teams by means of Natural Language Processing and Machine Learning

---

**Degree programme** Master's Programme in Industrial Engineering and Management

---

**Major / Code** Strategy / SCI3051

---

**Thesis supervisor** Professor Henri Schildt

---

**Date** 29.07.2017**Number of pages** 69**Language** English

---

**Abstract**

A Transactive Memory System (TMS) is a mechanism that captures the ability of teams to encode, store and retrieve knowledge collectively. TMS, thus, helps in locating “who knows what”. Such knowledge enables team members in an organisation to solve problems requiring knowledge beyond their own expertise, and has thus been suggested as one of the microfoundations of dynamic capabilities – the ability of organisations to renew themselves. TMS has been shown to be valuable for efforts to integrate and renovate knowledge assets of the organisation. However, prior research on TMS has focused mainly on face-to-face teams with only few studies considering the more difficult case of distributed work arrangements. In this research, I will expand on this proposition and present a computational framework that supports TMS in virtual teams.

The objective of the research was to broadly examine the ways in which machine learning algorithms and natural language processing techniques could be employed to provide support to TMS in virtual teams. Specifically, this research builds and evaluates a computational framework that pushes the boundaries of knowledge on distributed work arrangements through the lens of TMS.

The research methodology followed the design science research. The validation of the computational framework has been done using data mined from archived mailing lists of a real Free Open Source Software development virtual team. In order to identify who knows what in the studied virtual team, I used mined data from experts' conversations and survey data. Based on these foundations, I built a computational framework that involves two main components: The first component handles the mining of raw textual data and the second handles the classification of this data into broad areas of expertise.

My findings highlight the impediments to TMS in virtual teams and prove the usefulness of machine learning techniques and natural language processing in identifying expertise. Also, these findings suggest that it is possible and beneficial to support TMS through algorithmic means. From a theoretical point of view, this research contributes to the TMS research with a novel framework for augmenting TMS in distributed work arrangements. These findings are generalisable to a similar type of virtual teams. Although, only a limited number of skills were considered, the developed computational framework can be improved and extended to include a greater range of skills and other types of communities.

---

**Keywords** Transactive memory systems, Talent analytics, Distributed work arrangements, Natural language processing, Text mining, Machine learning, Python, Stack Overflow, Apache Spark

---

## Preface

There is no magic formula a data scientist can use to uncover the right patterns from any given big data. To leverage Big Data, creativity and insight are needed to bring to light the trends that would help organisations sustain their competitive advantage.

The completion of this research would not have been possible without the support of many persons. First and foremost, I would like to thank my supervisor, Professor Henri Schildt, for not only giving me the opportunity to work in his team but also for his direction and assistance. Henri, you have my gratitude for being patient with me and for giving me guidance and helping me with technical questions during my work. You added enthusiasm towards this research and writing this thesis. Your advice was invaluable and your comments and observations helped me improve the quality of this work.

The project was financed by the *Strategy Work and Big Data* project which is funded by the Academy of Finland. The project investigates at the societal level the changes Big Data could announce to management practices and norms, as they are complemented, challenged, and somewhat substituted by computerised simulations and algorithm-based solutions.

I would like to also express my gratitude to my research colleagues for the nurturing and fun atmosphere making this journey so pleasant.

Special gratitude goes to my girlfriend, parents and friends for their love and generous support in the successful completion of my studies towards a master's degree.

Helsinki 29 July 2017

Jamal Armel

## Table of Contents

<b>Preface .....</b>	<b>ii</b>
<b>1 Introduction .....</b>	<b>6</b>
1.1 Background .....	6
1.2 Research problem .....	7
1.3 Research questions .....	10
1.4 Structure of the research.....	11
<b>2 Literature review.....</b>	<b>12</b>
2.1 Transactive memory system .....	12
2.2 Machine learning .....	17
2.3 Natural language processing.....	19
2.4 ML and NLP combined.....	20
2.5 Python.....	24
2.6 Existing tools to support TMS in virtual teams.....	25
2.7 Summary .....	28
<b>3 Methods and Data .....</b>	<b>29</b>
3.1 Research design .....	29
3.2 Empirical settings .....	29
3.3 Data collection and pre-processing.....	32
3.4 Survey .....	35
<b>4 Data analysis and Results .....</b>	<b>37</b>
4.1 Considerations when choosing the right classification technique .....	37
4.2 Support Vector Machines .....	40
4.3 Text feature extraction .....	40
4.4 Training the Support Vector Machine classifier .....	43
4.5 Survey report .....	51
<b>5 Discussion.....</b>	<b>54</b>
5.1 Effects of virtualness on TMS.....	54
5.2 Effects of the developed framework on TMS in virtual teams.....	55
<b>6 Conclusion.....</b>	<b>57</b>
6.1 Summary of the accomplished work.....	57
6.2 Evaluation of the research and its limitations .....	58
6.3 Improvements and Recommendation for future research .....	60
6.4 Concluding remarks .....	62
<b>References.....</b>	<b>63</b>
<b>Archived code repositories .....</b>	<b>67</b>
<b>Appendix A: Transactive Memory Systems Survey .....</b>	<b>68</b>

## List of Figures

Figure 1 Transactive memory system .....	13
Figure 2 Machine learning Supervised process .....	18
Figure 3 Linear SVM geometric Intuition .....	23
Figure 4 Flow diagram for supervised document classification .....	24
Figure 5 Apache Spark development activity: Commits per month.....	31
Figure 6 Programming languages used to write Apache Spark .....	32
Figure 7 Workflow of building the training corpus .....	34
Figure 8 Terminal output for the data extraction process .....	34
Figure 9 Chart guide for choosing the right machine learning algorithm.....	39
Figure 10 Sample text with unigram, bigram and bag-of-words .....	42
Figure 11 One-vs-Rest scheme .....	44
Figure 12 Terminal output for the classification.....	48
Figure 13 Heatmap of the confusion matrix (percentage) .....	49
Figure 14 Answers to survey question Q4 .....	51
Figure 15 Answers to survey question Q2 .....	52

## List of Tables

Table 1 Ten most popular data science tools in 2016 .....	24
Table 2 Summary of technologies used to support TMS.....	27
Table 3 Summary of scraped data from Stack Overflow.....	33
Table 4 Classification performance of different k-fold sizes of the dataset .....	46
Table 5 Pipeline implemented for classification.....	50
Table 6 Team scores on three dimensions of TMS.....	53

## Definitions

Corpus	A large collection of texts. A corpus is a body of written or natural speech data upon which a linguistic analysis is based.
Feature vector	A feature vector is a vector that contains information describing a phenomenon's important properties. For example, in image processing, a simple feature representation of an image is the raw intensity value of each pixel.
Tokens	Loosely referred to as terms and roughly corresponding to words. Derived from a text through the process of tokenisation.
Classification	Establishing which category an entity falls under; this is a classic machine learning task. For instance, identifying whether an image represents a cat or not classifies it into two groups. Analysing data about music albums can result in classifying them into genres.
Web crawler	Also called a spider, is an Internet robot that automatically and methodically browses and indexes the World Wide Web, usually for the purpose of Web indexing by search engines.

## Abbreviations

AI	Artificial Intelligence
API	Application programming interface
CS	Computer Science
CSV	Comma Separated Values
FOSS	Free Open Source Software
IS	Information System
Q&A	Question and Answer
ROI	Return On Investment
SVM	Support Vector Machine
TMS	Transactive Memory System
TOS	Terms of Service

# 1 Introduction

## 1.1 Background

Competitive advantage is a fascinating longstanding riddle in management studies. Many explanations have been given for the reasons for which some businesses achieve and sustain competitive advantage while others stumble. One of these explanations is the dynamic capability argument. This argument stresses the importance of the organisation's outstanding capacity to sense new opportunities and seek them by incessant adaptation, integration and reconfiguration of its internal and external abilities in a continuously shifting landscape (Teece et al., 1997). There is a need for more research on identifying the mechanisms, processes and routines that can be identified as offering microfoundations by which organisations develop dynamic capabilities (Teece, 2007).

Argote and Ren (2012) proposed transactive memory as a microfoundation of dynamic capabilities. They described how a system that encodes, stores and retrieves knowledge in an organisation can assist it with its efforts to integrate and renovate its knowledge assets. In this research, I will expand on their proposition and present a computational system that supports this microfoundation.

Coming up with novel methods to collect and analyse data should not be confined anymore to marketing, finance, sales or product development departments. Human resources function, just like the aforementioned operations has a large volume of data about employees and their performance. Still, the problem facing Human resources is that its approach to managing talents differs greatly from the methodologies developed by the other organisational functions. While marketing and finance generate insights that executives build upon to make strategic decisions, Human resources usually concentrates on what it does (Boudreau and Ramstad, 2005).

To address this gap, Human resources should move its attention from its own performance and strive to come up with talent insights that support strategic decisions of the organisation (Boudreau and Ramstad, 2005). Human resources can be assisted in this regard with the latest advancements in talent analytics.

As competition through talent intensifies between organisations, human capital investments become a determining factor in their competitive position. Talent analytics are being used to attract and retain talents in industries experiencing intense competition for talent. Human resources is already using analytics in such ways as to assess flight risk and team performance for example (Davenport et al., 2010). Still, in this regard, Human resources trailed other departments. Since, although it amasses a staggering amount of data on employees that makes it more efficient at measuring turnover and ROI on specific programs, it still struggles to relate this data to improvements in business performance and leading programs to the long-term needs of the organisation.

Recently, Human resources departments are also experimenting with new tools and building new capabilities for insight discovery that would let them contribute to the improvement of the organisation's strategic decision making. Granted that to secure funding for the acquisition of such technologies, Human resources needs the approval and full backing of the leadership in the organisation. HR needs to show that they too can foster a culture based on the scientific method and data-driven decision making and not just gut feelings.

## **1.2 Research problem**

Human resources as part of organisations that are faced with fast-changing business landscape, is more than ever pressed to help create, use and share knowledge to establish and sustain competitive advantage. And at the centre of these knowledge-based organisations are teams which are becoming more and more virtual. These teams are increasingly geographically distant and the bulk of their communication is computer-mediated. This raises issues in managing and coordinating knowledge between teams as this knowledge is scattered among team members. Hence, it is important that the Human resources can identify and make the most of these individual's knowledge. Still, there is a lack in the understanding of how members of virtual teams are aware of their teammates' knowledge, how they build trust in each other's expertise, and how they achieve effective coordination of that knowledge.

Expanding the concept of Transactive Memory System (TMS) beyond teams and couples, speculation has been made on how organisations may work as TMS. Anand et al. (1998) devised a prototype that demonstrates how an organisation may be regarded



as a collection of TMS. They argued that some sort of information system (IS) (e.g. intranet, search engine, forum...etc.) may be utilised to improve the development of organisational TMS.

Moreland (1999) argues that TMS in an organisation could have other characteristics than those of the TMS in couples and groups. For instance, sizeable groups have a diminished cohesion and their members are less inclined to share knowledge. He suggests that organisational TMS may be built following interpersonal and technological approaches. The technological approach would consist of using IS to construct and support an organisational TMS. Moreover, Griffith et al. (2003) indicate that poor TMS development in virtual teams “will be mitigated to the extent that technologies or organisational systems are used to support transactive memory development”.

Free and Open Source Software (FOSS) is a good illustration of distributed work arrangement in which the development process is mostly done in virtually distributed teams. FOSS has been extensively researched because of the possibility to mine data from archives of mailing lists, source code and bug reports. Such repositories are mined in hope of uncovering empirical evidence supporting hypotheses concerning FOSS’s development process. Even though existing FOSS mining techniques yielded sound results, more challenging questions require different and a more suitable approach to analyse it. In this thesis, I am exploring an alternative approach to mining FOSS repositories called text data mining. In addition to development processes and design decisions, these mailing lists contain information about developers’ traits and qualities. Text analysis tools will be used to understand and predict these developers’ characteristics including their skillset.

Text contained in a mailing list is less structured than other types of data encountered in a FOSS development environment such as source code or issues report. This gives developers more flexibility to discuss a wide range of issues, yet it hinders the mining of meaningful information. This wide variety of topics discussed and the resulting ambiguity in the conversations on a mailing list is higher than the one found in an issues tracker database for example. This results in making the classification of text a complex task.

Due to this nature of text as not having a structure, we are referring to it as unstructured data. Nonetheless, text has a linguistic structure suited for human comprehension but not

for machines. To transform the unstructured text data to a structured data suitable for analysis I am going to apply natural language processing (NLP) and the analytical methods associated with it.

In this thesis, I develop a model that assesses behavioural characteristics linked to Transactive Memory Systems (TMS) in virtual teams by means of Natural Language Processing (NLP). The scope is limited to virtual teams which are defined by Jarvenpaa and Leidner (1999) as a temporary, geographically distant working team which communicate electronically. The fact that these teams are temporary means that the members do not necessarily know each other a priori and might not work in the same team in the future. The geographical distance between the team's members means that they are located across geographical but also usually across organisational boundaries. They hardly if at all have a face-to-face meeting. Lastly, this kind of teamwork is possible through computer-mediated communication.

TMS is assessed in this thesis as it was conceptualised by Lewis (2003) as a combination of three dimensions: Group knowledge stock; Specialisation of expertise; and accuracy of knowledge identification.

With the flourishing applications of machine learning in various areas and the surging need in software engineering to deal with the huge amount of text online, machine learning gained traction in speech recognition and natural language classifications. This study takes advantage of the advancements in machine learning tools in linguistics to build classifiers based on labelled text mined from a large programming Question and Answer (Q&A) site. Q&A sites such as Stack Overflow (2017) make it possible for developers to explain their problem to the other members and get solutions, these may include code snippets together with attached explanations, named code-description mappings (Wong et al., 2013).

This research addresses issues in NLP concerned with large corpora built from FOSS's mailing lists to classify conversations by domain of knowledge. NLP is a branch of computer science that uses Machine Learning (a subfield of Artificial Intelligence) and linguistics to analyse and understand human languages. The input can be speech, text or keyboard input but NLP is particularly aimed at processing sizable corpora of text.

Document classification aim is to choose the right class label for a given textual input. In the basic classification task, independence of each input is assumed, and the list of labels is pre-defined. Some illustrations of document classification are:

- Categorising emails into spam or not spam.
- Assigning a topic to news articles, from a fixed set of topics like "Business" "Science" and "Politics"

The prohibitive costs of manual annotations have made automatic classification a much more attractive alternative. This automated process is based on choosing words from the document that would provide a satisfactory impression about its content. Several classification methods have been developed by using properties of words in the document and corpora of texts or by means of external sources like dictionaries.

### 1.3 Research questions

This research studies how one of the fundamental microfoundations of dynamic capability can be augmented with artificial intelligence techniques, and what benefits could this have on the TMS of virtual teams. I emphasis three key processes: TMS, Machine learning and NLP. Through the lens of these processes, I will present a novel method to supporting TMS in virtual teams algorithmically, as opposed to the prevailing methods that use only survey data.

The hypothesis that drives this research is that: By offering algorithmically augmented support to TMS we can be benefit the development of TMS and as a result improve the functioning of the virtual team.

Consequently, this research will focus on the following three research questions:

**RQ1:** How TMS manifests itself in virtually distributed teams and how can we locate who knows what within these teams?

Answering this first question would lead to a proper grasping of both notions of TMS and virtually distributed work, this in turn would help answering the following questions.

The suggested potential of text classification to pinpoint the skillset of a team member sets the basis for the second research question of this thesis:

**RQ2:** How can we map the skillset of the individual members of a virtually distributed team? And how can we extract the information that represent these skillsets?

Finally, the third question is founded on the outcome of the study on the empirical settings featured in the third chapter of this thesis:

**RQ3:** How would the developed software tool support the development of TMS within virtually distributed teams?

The answer to this question may pave the way for future IS built with requirements satisfying the optimal functioning of TMS when establishing actual virtually distributed teams. Indeed, through an algorithm-based artefact founded on NLP and ML, I propose a conceptual model of an IS to augment organisational TMS. This may inform future implementations of computationally efficient forms of TMS.

## 1.4 Structure of the research

This research is organised as follows: In chapter 1, I identified the problem, formalised the research questions and presented the structure of this research. In chapter 2, I performed a systematic literature review on Transactive memory systems and an outline of the state of the art in text mining, machine learning and natural language processing. I also briefly reviewed the existing tools that support in one way or another TMS in virtual teams. Chapter 3 identifies the methods and design propositions, it also presents a description of the research data and outlines its collection and pre-processing methods. It also presents the plan of the survey. In chapter 4, I formalise the proposal of the artefact to solve the research questions and present the steps followed to develop it, I also evaluate the performance of the artefact, moreover, a brief survey report is provided at the end of this chapter. In chapter 5, I discuss the findings of the research and put them into their theoretical perspective, I also evaluate the relevance of the developed artefact in helping improve the development of TMS. Finally, chapter 6 concludes the research by giving a summary of the work conducted and identifying the limitations of the research and also suggesting directions for future research.

## 2 Literature review

In this section, I review the literature about the theoretical foundations of transactive memory and the advancements in machine learning and natural language processing techniques. Additionally, I will give an overview of the already existing tools that support TMS in organisations.

### 2.1 Transactive memory system

The Transactive Memory System (TMS) was first described by Wegner et al. (1985) in a study of the memory structure of romantic couples where they defined it as “a set of individual memory systems in combination with the communication that takes place between individuals”.

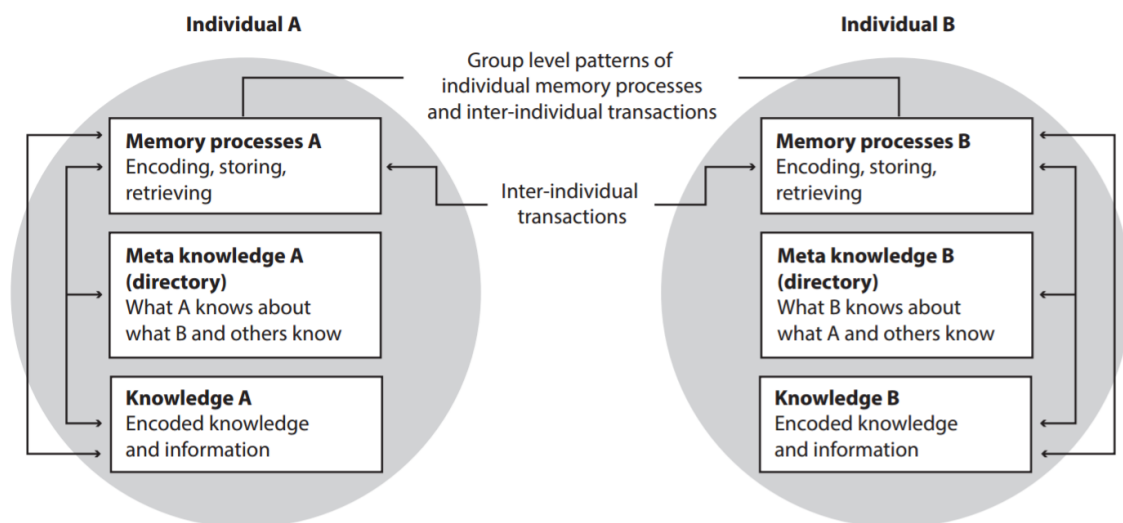
In his study of TMS in close romantic relationships Wegner (1987) asserted that individuals tend to expand their limited memory by utilising external memory aids such notes, dairies or by relying on other individuals for recalling information. For this reason, Wegner concluded that TMS is an automatic process that arises in natural groups. It is this last aspect that prompted research on whether work groups develop a similar mechanism of memory retrieval and organisation as the one studied in romantic couples.

Wegner's original theory was a significant departure from the general understanding of the group mind (Wegner et al., 1985; Wegner, 1987). Until that time, the group mind was thought to be a linear progression that replicated the processes of the individual mind. Wegner disagreed with this concept and he argued that the group mind behaves rather differently from the cumulative processes of the individual mind. Moreover, he argued that TMS is not traceable to any individual component of the group but rather exists separately from the individual memories of the group. Thus, Wegner's contribution in the conceptualisation of TMS was to better clarify the workings of the group mind as a distinct concept by describing its various elements.

TMS theory supplants previous theories which predominantly considered group memory as an individual memory system (Wegner, 1987). Even though TMS is more complex, notions associated with the memory of an individual are still relevant to those regulating TMS, these include encoding, storage, and retrieval. Still, other notions became

prominent in this model, these include accuracy, validation and sharedness of the group's shared mental model (Brandon and Hollingshead, 2004).

TMS captures who knows what in a group and check for the uniqueness of that knowledge. Individuals in interpersonal relationships tend to form a specialised division of cognitive labour regarding encoding, storage and retrieval of knowledge from various substantive domains. Therefore, expertise in narrow areas of knowledge is developed by each individual in the group. Other individuals in the group are expecting a member to possess such expertise as explained in Figure 1. This results in a reduced overlap of members' knowledge and a more efficient processing of the information within the group.



*Figure 1 Transactive memory system (Wegner, 1987)*

In its original form, TMS was conceptualising how the group mind might operate. When it was first conceptualised, there was not enough empirical evidence to back up the existence of TMS as a phenomenon. It could not be effectively measured empirically and not enough metrics were in place to clarify the various TMS processes.

Building on their research on learning in successful teams, Moreland, R. L. et al. (1996) moved empirical research on TMS from personal relationships to groups. The goal of their research was to test whether TMS had a mediating effect on the training methods and group performance. What resulted however is the emergence of three constituents that indicated TMS: Specialisation, Coordination and Credibility. Specialisation

represents the degree of knowledge differentiation in a group. The specialisation of members of a group may be explained by coincidental knowledge, pre-existing individual expertise, or by the consent to take responsibility for a specific knowledge as part of the group. Subsequently, those members are supposed to re-share information with the group. Coordination denotes the ability of individuals in a group to collaborate efficiently. Credibility indicates the level of confidence in the accuracy of the information each individual of the group has about other teammates.

These pioneering empirical studies therefore produced some preliminary measures by which TMS is tested in groups and thus led to an improved understanding of this phenomenon.

Moreland, R. L. et al. (1996) observed student teams during an experimental radio kit assembly exercise. Among these teams, those instructed together performed better than those instructed individually. The researchers examined the videotaped experiment and coded behaviours indicative of TMS. Measuring the TMS was derived from observations made about participant's knowledge (inferring Specialisation), participants' confidence about the reliability of teammates' knowledge (inferring Credibility) and about the level of efficiency of knowledge processing (inferring Coordination). When all these TMS behaviours were accounted for however, all teams performed equally, those trained together and individually. Therefore, the researchers concluded that TMS accounts for the enhanced group performance. There could be, however, other causes for the improved performance apart from a stronger TMS, these can be a better communication or training.

R. Moreland, L. Myaskovsky (2000) conceived two new experimental settings to determine whether the performance improvement attributed to group training is due to better communication among teammates rather than to TMS. This set of experiments showed that TMS is different from training and communication and is improving group performance by knowledge of teammates' skills. Students were randomly assigned to sixty-three three-person same-sex teams which were then subjected to different training conditions: Individual, Performance Feedback, or Group training. These teams were then instructed to perform a radio kit assembly. Teams whose members underwent Performance Feedback condition with no prior communication with other teammates performed just as well as teams in the Group training condition. The aforementioned

training methods did not affect team members' ratings on the ease of communicating with teammates about radio assembly.

TMS coordination represents a group members' aptitude to collaborate efficiently, it is deemed critical for improved group performance. Team members engaging in coordination action were observed during a laboratory experimental set by Hollingshead (2000). For this experiment, clerical workers were tasked to work together. The workers managed to recall and acquire new knowledge when their collaborators were in different area of expertise. Opposite outcomes to these were obtained when the same workers collaborated with partners with the same expertise.

TMS credibility indicates the level of confidence team members have in the knowledge accuracy of teammates, it is an indicator of members' mutual trust in each other's expertise. R. Moreland, L. Myaskovsky (2000) showed through the radio assembly experiment that the teams who trained together had a higher level of TMS accuracy than those who trained individually. These findings support Wegner's conceptualisation of TMS that indicates that a person relies more on others for information when he/she considers them as a credible source of knowledge.

TMS has been studied in field settings and it was demonstrated that it accounts for improved group performance. Research by Austin (2003) further examined multiple dimensions of TMS and worked on operationalising them. These generalisable measures benefitted research on TMS in various contexts.

Austin (2003) put forward hypotheses that investigated the connexion between a group's performance and the behavioural dimensions of its TMS. The results indicate that the accuracy of the TMS is a strong indicator of a group's performance. Thus, accurately assessing an individual's area of expertise within a group would improve a group's performance. Austin (2003) studied the relation between a developed TMS and improved performance in 27 teams working for a sporting goods organisation. Austin identified through semi-structured interview protocol 11 skills and knowledge areas for the study of these teams. Team members were tasked to identify individuals in their team whom they thought had the most knowledge in that specific expertise or knowledge domain.

Austin (2003) conceptualised TMS as a combination of four dimensions: Group knowledge stock; Consensus about knowledge sources; Specialisation of expertise and



accuracy of knowledge identification. Knowledge stock of a group is, as defined by Austin, the aggregation of individual knowledge. Group knowledge stock is considered the central corpus of knowledge that can be accessed by any member of the group. The consensus about knowledge sources can be thought of as the degree to which members of a group agree about who has what knowledge. This aspect draws a parallel with Wegner's transactive encoding where a group examines received knowledge and process it in various ways to achieve a common understanding. Austin states that this component of TMS offers a link to research on team mental models for performance evaluation of a group. Specialisation of expertise, a dimension proposed by Wegner as well, is defined by Austin as a deeper knowledge base in a narrowly delimited field of expertise. The last dimension, accuracy of knowledge identification, is described by Austin as the extent to which members recognised by others as holding a specific knowledge actually do possess it. These results showed that TMS can be assessed in field settings although this requires first-hand observation involving an identification of the team's skill sets which can be inconvenient. Measuring TMS using this method would prove laborious and cumbersome in organisations with teams that have vastly diverse areas of expertise.

Although laboratory settings supported conceptualising TMS, adapting these measures to a field setting was difficult. For example, measures utilised during an experimental setting like radio assembly set were developed for a setting with well-defined tasks which did not vary among comparison groups, this would be a remarkable feat in a real-life organisation. To a certain extent, this caused fragmentation in TMS research and a reduced interest from organisations. This difficulty was mainly overcome by (Lewis, 2003) when she described how she developed and test validated a field measure of TMS. The measure consisted of a 15-item scale tested in a laboratory setting of 124 groups, a field setting of 64 Master of Business Administration consulting teams, and a field setting of 27 teams from technology companies. Lewis' scale consists of three sub-scales assessing the three components of TMS: Specialisation, Credibility and Coordination. The results of her study prove that indirectly measuring TMS is as effective at monitoring TMS as the direct measuring. This is significant to field research since measuring TMSs by direct observation is impractical in several applied settings.

In face-to-face settings, research showed that groups develop TMS by using any appropriate knowledge on hand, this may include past experiences, formal or informal

conversations between teammates, expertise based on assignments and surface characteristic such gender, ethnicity and age (Brandon and Hollingshead, 2004; Lewis, 2004; Wegner, 1987). Notably, Hollingshead (1998) observed that important information was encoded in the nonverbal and paralinguistic communication cues present in the communication used to retrieve information from others in a team. Since there is a scarcity or sometimes absence of such cues in virtual teams, developing TMS can be difficult. Overcoming these difficulties is possible when communication between teammates is frequent and effective.

According to the first conceptualisation by Wegner (1987), TMS is a team-level concept, which makes measuring it a demanding undertaking. Measuring the group knowledge can be approached essentially in two ways: holistic and collective (Cooke et al., 2000). The collective method requires, first, that individual measures be gathered and then combined into a team-level measure. The individual measures may be gathered from surveys, discussions, or by observing groups (Cooke et al., 2000). Although this aggregation method is preferred by most recent researches for its ease of use, still it overlooks the significance of the interactions between individuals in a team (Cooke et al., 2000). The collective approach thus assumes that individuals equally influence group's cognitive phenomena (Mohammed et al., 2000).

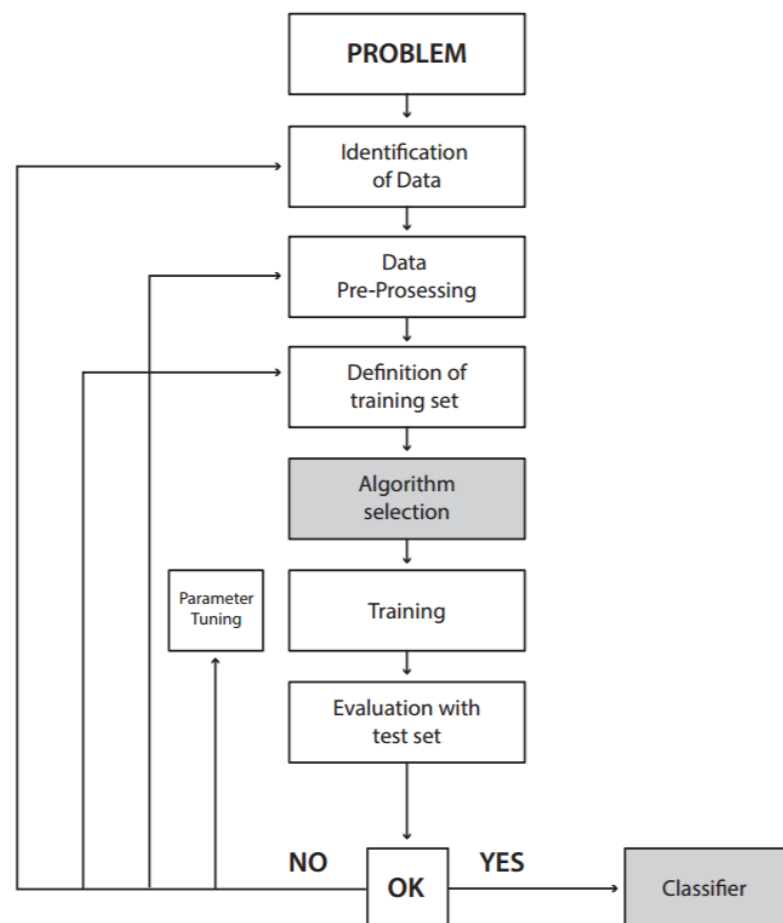
## **2.2 Machine learning**

Machine learning is about extracting knowledge from data. It is an important branch of artificial intelligence and it is also an interdisciplinary field of statistics and computer science (Sebastiani, 2002). It began as pursuit to develop an algorithm that is able to learn, adapt, optimise and get better from experience and environment (i.e. data). Machine learning has many applications in various disciplines, notably in marketing personalisation, fraud detection, NLP and financial trading (Marr, 2017).

Machine learning uses algorithms to make it possible for computers to learn, without being programmed, to identify patterns indiscernible to human operators. Essentially, a machine learning algorithm is presented with a collection of teaching data, then is expected to utilise that data to solve a problem. For instance, an algorithm can be given a teaching set of images with captions saying either “this is a fish” or “this is not a fish”. Then the algorithm is provided with a new set of images and it would recognise which

images represent fish. Machine learning keeps on adding images to its teaching set. Each image that it classifies (successfully or unsuccessfully) is included in the teaching set, and the algorithm actually becomes smarter and more reliable at solving the problem with each iteration. It is, basically, learning.

Machine learning algorithms can be categorised according to the anticipated outcome of the algorithm. Typically, machine learning algorithms are classified into three categories: Supervised, unsupervised and reinforcement learning. A supervised learning process, as seen in Figure 2 requires that when given input and output variables, an algorithm has to estimate a general rule for mapping inputs to the desired outputs. The objective is to get an accurate approximation of the mapping function that can be used to predict the output variables for that data from a new input (Manning, Christopher D., and Hinrich Schütze., 1999).



*Figure 2 Machine learning Supervised process*

This method is named supervised learning since the algorithm is trained on a dataset with known correct answers. This set acts as a teacher that supervises the learning process. The algorithm repeatedly attempts to make predictions on the training data and the “teacher” is correcting it. This learning process ends once the algorithm reaches a satisfactory level of accuracy (Manning, Christopher D., and Hinrich Schütze., 1999).

Unsupervised learning is when given only some input variables with no mapped output variables. The objective of such method is to uncover hidden patterns in the data. These algorithms are named unsupervised learning since contrary to the supervised learning, there are no correct answers and there is no “teacher”. The goal is to have the algorithm learn how to find the interesting structure in the dataset (Manning, Christopher D., and Hinrich Schütze., 1999, p.232).

## 2.3 Natural language processing

Organisations are just beginning to understand the enormous potential value stored in all the text generated on a daily basis. Organisations are leveraging Natural Language Processing (NLP) to derive understanding from the massive unstructured data available online and create new value and improve efficiency.

NLP has long been one of the issues that are hotly researched in computer science because of its great significance. Although machines surpass humans at making sense of highly structured data, yet there are certain vital areas where humans are indisputably better than computers. Comprehending natural language is one of these areas. In this section, I am explaining what NLP is, how it works, and how pairing it with machine learning could solve text classification problems.

NLP roots date to the period just after World War II, it started as the intersection of artificial intelligence and linguistics. NLP was founded on several diverse fields such as computer science, linguistics, mathematics, electrical engineering and psychology (Jurafsky, D. Martin, J.H., 2014), NLP researchers are thus expected to widen their knowledge base considerably.

With the advent of machine learning, the 1980s was a crucial decade for NLP as it resulted in deep changes, these include:

- Deep analysis was supplanted by approximations that are simple and robust.

- The use of probabilities in machine learning processes grew considerably.
- Sizable, structured set of texts called corpora were utilised to train machine learning algorithms.

Certain NLP tasks are done more routinely than others, these include:

- Tokenisation: dividing text into a sequence of tokens, which more or less corresponds to words;
- Part-of-speech tagging: reading text and assigning parts of speech to each token, such as noun, verb, adjective;
- Generating parse trees: performed to identify grammatical relationships in sentence;
- Named entity classification: e.g. grouping names of animals, months of the year, or countries;

From these elementary tasks, it is possible to build more complex applications, like the one I will be reviewing in the next section.

## 2.4 ML and NLP combined

The automatic classification of text is a good instance of how ML and NLP can be paired together to allow machines to develop better models to understand human language. The aim of text classification is to categorise documents into one, two (binary classification) or more (multiclass classification) classes to help sort through big datasets of documents. The costs of doing this operation manually are prohibitive and labour intensive. Moreover, this automation would allow machines to develop better models than they would have done by relying only on a static set of commands from human programmer.

Unsupervised ML models (e.g. Clustering and Topic Modelling) are utilised in the automated uncovering of clusters of similar documents within a larger collection of documents. However, in this thesis I focused only on supervised techniques of classification.

### 2.4.1 What is the Classifier's purpose?

Classifiers make predictions. In other words, when a classifier is presented with a new text to classify, it predicts the class to which the text belongs and assigns it a class label. If the classification algorithm or the strategy used allows it, the classifier can also return

a confidence measure to show the accuracy of the classification in assigning labels to documents.

## 2.4.2 Flow of Supervised Documents Classification

Document classification process is done in the following three main steps:

### 2.4.2.1 The dataset

A set of documents is needed to perform a statistical method of classification, these documents must be already manually labelled with the correct class name. The quality of this set determines the accuracy of the statistical NLP classifier.

Another important characteristic of this set is its size, it has to be large enough and include a suitable number of items in every class. Additionally, the set must contain documents which have a relatively low degree of attributional similarity in various classes to allow a well-defined demarcation between these classes.

### 2.4.2.2 Pre-processing

Supervised approaches to text classification include statistical features which are computed based on statistical information gathered from the training documents. The raw form of these documents features words of equal importance. Pre-processing is needed to rid the text of the superfluous words and give more weight to words based on their importance in the text. A popular methodology used to do this is term frequency and inverse document frequency (TF-IDF). This methodology that helps to classify documents has been studied at length, in particular by (Salton et al., 1975; Salton and Buckley, 1988) for their vector space model. As a result of their reflections on the term discrimination model, they demonstrated that terms appearing in stored entities (documents) should be weighted proportionally to the term frequency and inverse proportionally to the entity's frequency (Salton et al., 1975). Additionally, Sparck Jones (1972) introduced the concept of inverse document frequency (IDF) by proposing a weighting scheme for specificity of a term.

Combining TF-IDF weights and entity length normalisation resulted in superior retrieval performance on collections of documents and it has been widely used in supervised approaches (Provost, 2013).

TF-IDF is usually calculated as follows:

Given a set of  $Q$  documents, let  $f_{wd}$  be the frequency of the term  $w$  in the entity  $d$ . Then the term frequency  $TF_{wd}$  is defined as:

$$TF_{wd} = \frac{f_{wd}}{\max_k f_{kd}}$$

Meaning that the term frequency of the word  $w$  in the document  $d$  is  $f_{wd}$  normalised by means of dividing  $f_{wd}$  by the maximum count of occurrences of any other word in the document  $d$ . Therefore, the word with the highest frequency in the document  $d$  would get a  $TF_{wd}$  of 1, while the other words would get frequencies which are less than 1.

On the other hand, the IDF for a word is calculated as follows:

Assuming that the word  $w$  occurs in  $q_w$  of the  $Q$  documents in the collection, then:

$$IDF_w = \log_2(Q/q_w)$$

Together, the TF-IDF weight for the word  $w$  in the document  $d$  is the product of the two statistics calculated above:  $TF_{wd} \times IDF_w$ . The words with the highest TF-IDF are usually the words most suitable to describe the topic of the document (Provost, 2013).

#### 2.4.2.3 Classification Algorithms

There are numerous algorithms that have been used on quantitative data. And it is conceivable to utilise the majority of these existing techniques for text classification. Because text can be modelled as quantitative data where word's attributes has frequencies. Still, textual data is a special type of data because the word's attributes are scarce, and high-dimensional, with the majority of words characterised by low frequencies.

Broad classes of techniques, that are typically utilised for text classification include Support Vector Machines (SVMs), Bayesian, Decision Trees and Neural Network.

In machine learning, an SVMs are a subclass of supervised learning paradigm coupled with learning algorithms that analyse data utilised in classifications. Presented with a set of training samples, each one labelled, an SVM training algorithm can build a model that labels new samples. It is thus partitioning a feature space into two or more subsets. Which in our case means separating a corpus of text files into two or more classes.

An SVM creates  $n$ -hyperplanes in an  $n$ -dimensional space, which may be utilised in classification and regression tasks. Thus, an effective separation is constructed by the

hyperplane with the greatest distance to the adjacent sample points from each class, given that typically the bigger the margin the smaller the generalisation error of the classifier.

Since an SVM is a large margin classifier, specifically, it maximises the geometric distance between the decision boundary and the classes of samples. In simple instances, the separation boundary is linear, leading to clusters that are separated by lines in  $n$ -dimensional spaces. Thus, the geometric intuition behind it is often plotted in the following way:

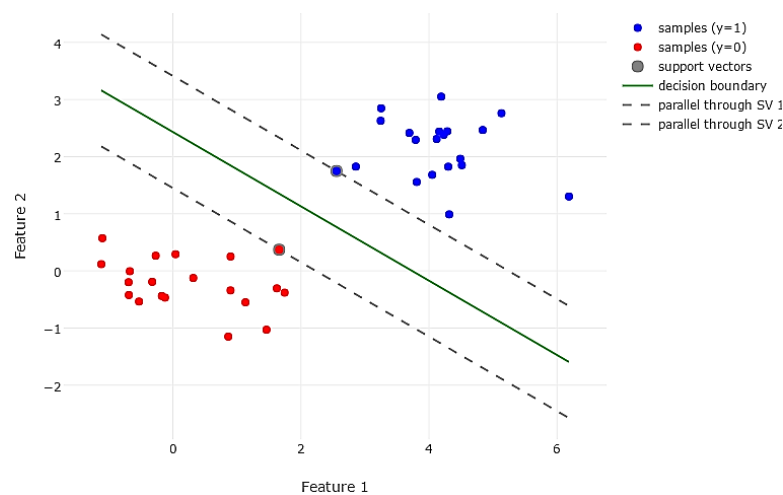


Figure 3 Linear SVM geometric Intuition (Hamel, 2009a)

Experimentation showed how SVMs reliably achieve excellent performance on text classification tasks, they usually outperform other techniques considerably. SVMs are able to generalise appropriately in high-dimensional feature spaces and they do not need feature selection, thus making the text classification significantly simpler. Furthermore, SVMs are more robust than the other conventional techniques. Together, these advantages make SVMs a suitable and easy-to-use choice for training text classifiers from example documents. (Joachims, 1998)

Since their inception in 1995 (Cortes and Vapnik, 1995), SVMs have become one of the leading machine learning models. SVMs are now used commonly in fields ranging from classification of images and hand-written character recognition to mining big databases (Hamel, 2009b).

Figure 4 summarises all the previously discussed text classification steps:



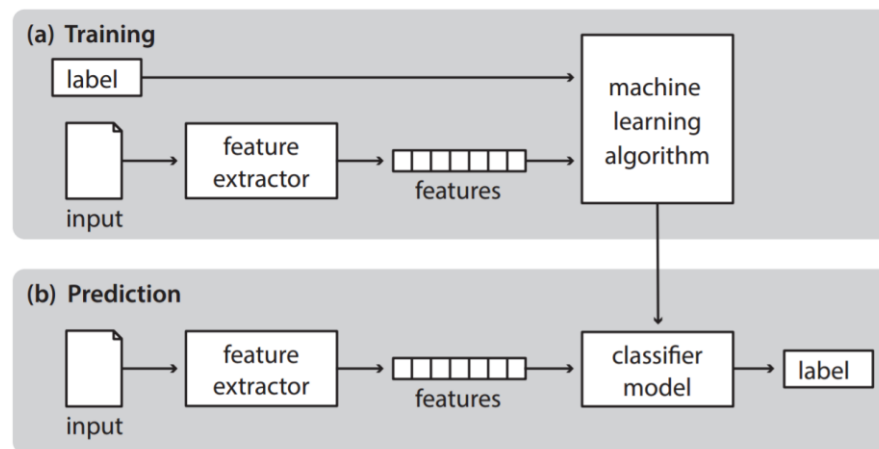


Figure 4 Flow diagram for supervised document classification

## 2.5 Python

The most prevailing data science tools used by the analytics community are the R and Python programming languages. The two languages are commonly used by statisticians. Although R's functionalities were conceived with statisticians in mind, Python is commended for its intuitive and easy-to-use syntax enabling developers to write concise and efficient algorithms. Python is better suited for data scientists who have a CS/developer background, which is why I chose to work with it in this research.

Table 1 Ten most popular data science tools in 2016 (KDnuggets, 2017)

Tool	2016 % Share	% Change
R	49	+4,5
Python	45,8	+51
SQL	35,5	+15
Excel	33,6	+47
RapidMiner	32,6	+3,5
Hadoop	22,1	+20
Spark	21,6	+91
Tableau	18,5	+49
KNIME	18	-10
Scikit-learn	17,2	+107

While not as feature-rich for analytics as R, Python's resources for scientific computing are expanding fast. As of June 2016, R and Python are duelling as top analytics/data science tool but Python is growing faster and soon will overtake R. Table 1 Shows the most commonly used data science tools in 2016 according to a poll by (KDnuggets, 2017).

The growing pace of Python's functionality is staggering. As of May 2014, there were 44000 packages in the Python Package Index. At the end of May 2017, there are over 108,000 packages listed. Of these, 6929 are labelled as Scientific/Engineering (Python.org, 2017).

Python, a general-purpose programming, natively supports basic functionalities necessary for analytics, such as data import and program control. For advanced analytics, additional libraries (i.e. NumPy and SciPy) offer added capabilities, such as, multidimensional arrays and matrices; high-level mathematical functions; optimisation; and linear algebra (Dinsmore, 2016).

The most popular and feature-rich advanced analytics library for Python is by far scikit-learn. This open source and free library builds on NumPy and SciPy to offer several supervised and unsupervised machine learning algorithms for classification, regression and clustering. Notable algorithms include logistic regression, support vector machines, random forests, naïve Bayes classifier and K-means. (Scikit-learn, 2017)

Continuum Analytics is the publisher of Anaconda, a freemium and open source distribution of Python that delivers a multitude of improvements for large-scale data processing and predictive analytics. These enhancements include carefully selected Python packages and their dependencies for data science and analytics. Continuum Analytics also aims to simplify package deployment through an online repository with an updated version of every package. (Continuum Analytics, 2017)

## **2.6 Existing tools to support TMS in virtual teams**

In this section, I will briefly review few of the computational systems that have been used to support TMS, by using the following illustrative examples:

- *Answer Garden* (Ackerman and Malone, 1990) is a web-based tool that help the development of organisational memory by assisting individuals in finding and sharing questions and answers;
- *GIMMe: Group Interactive Memory Manager* (Lindstaedt, 1996) is also web-based tool which is a centralised repository to store and access emails and conversations in an organisation environment;
- *KSE/Jasper II: Knowledge sharing environment* (Merali, Y. and J. Davies, 2001) is a system that facilitate the sharing of information agents linked to each user. These information agents are able to organise, summarise and share knowledge from potentially many information sources in order to answer queries;
- *MILK* (Agostini et al., 2003) is at its core a metadata management system used by organisations to help communities of interest. It integrates knowledge related to all sorts of involved entities, which includes users, communities, and informal knowledge;
- *OntoShare* (Davies, J. and A. Duke, 2004) is an environment used by virtual teams to share knowledge based on ontologies. OntoShare uses Semantic Web technologies to assist communities of practice;

All the mentioned systems and tools promote the development of transactive memory in one way or another. A search tool that assists users in locating relevant knowledge and associated people is offered by all the systems and is used to ease the building of TMS. For instance, MILK's search tool indexes people's expertise and the information available within the community based on information collected from the user's profile and the metadata associated with the documents. MILK's Metadata Management System (MMS) oversees the acquisition and organisation of knowledge from profiles that contain metadata descriptions of different aspects of the involved entities. Inaccuracies, however, occur with this method since the metadata are labelled by individuals who uploaded the material, while the profiles were built merely from how the users' interacted with the system. These issues were mostly tackled in KSE/Jasper II and OntoShare which offer improved search tools that use keywords extracted from the whole document. Additionally, KSE/Jasper II and OntoShare allow individuals to do queries to look for users based on their interests. These search results are created from dynamically maintained users' profile that can be edited by any user. Similarly, Answer Garden and

GIMMe emphasise the importance of NLP to benefit the development of TMS. For instance, Answer Garden operates a full text search engine to assign trusted answers by experts to a user's query. Even though pinpointing expertise is associated with stronger TMS, Answer Garden chooses to keep the users' contributions anonymous, which prevents locating the communities' expert members. GIMMe uses latent semantic indexing to recognise the underlying structure in the relations between the words and topics included in an unstructured corpus of text. This eases the search through vast repositories of email archives. GIMMe also mines mails sent to a specific group alias and automatically adds it to an information space and categorises it according to its subject, this could be beneficial for TMS. Table 2 summarises the above mentioned illustrative example.

*Table 2 Summary of technologies used to support TMS (adapted from (Kleanthous and Dimitrova, 2006))*

Type	Capability	Technology
Search	Search through metadata utilising user's profile	MILK
	Keywords extraction from documents and association with user's profile	OntoShare, KSE/JASPER II
	Text retrieval built on keyword extraction	Answer Garden
	Latent semantic indexing	GIMMe
Recommendations	Recommends resources and experts on user's profile	OntoShare
Semantic-aware techniques	Metadata to link information	MILK
	Category hierarchy	GIMMe
	Ontology	OntoShare

Semantic technology, which encodes meanings separately from data and content files, have been used to support the building of TMS. For example, GIMMe keeps hierarchically structured categories that enables knowledge location. These categories, nevertheless, are freely built by the users and become disorganised, thus hindering expertise and knowledge location, this in turn does not contribute to the development of TMS. On the other hand, OntoShare draws on ontology of domain categories to locate knowledge and identify members with similar interests.

## 2.7 Summary

In the previous sections, I have reviewed transactive memory, machine learning and natural language processing theory. In my review of transactive memory literature, I showed that studies on organisational psychology had found that effective teams working within an organisation build transactive memory and recognise who are their cognitively central and peripheral peers. Moreover, these developments may be extended to a wider setting to include virtually distributed teams.

I showed that typically, machine learning algorithms are grouped into three classes: Supervised, unsupervised and reinforcement learning. And that a supervised machine learning algorithm is trained on a dataset with known correct labels. This set acts as a teacher that supervises the learning process. The algorithm repeatedly attempts to make predictions on the training data until it reaches a satisfactory level of accuracy.

I noted that organisations are leveraging NLP to derive understanding from the massive unstructured data available online and create new value and improve efficiency. And that NLP has a great significance in the field of computer science. Although machines surpass humans at making sense of highly structured data, yet there are certain vital areas where humans are indisputably better than computers. Comprehending natural language is one of these areas. NLP is trying to fill this gap.

I explained how automatic classification of text is a good instance of how machine learning and NLP can be paired together to allow machines to develop better models to understand human language. And that in machine learning, SVMs are a subclass of supervised learning paradigm coupled with learning algorithms that analyse data utilised in classifications. Presented with a set of training samples, each one labelled, an SVM algorithm can build a model that labels new samples.

I pointed to tools already used to support TMS in virtual teams. However, each of these tools has some shortcomings which I will try to address through the present computational framework.

### 3 Methods and Data

In this section, the general design of the research and methods used in data collection are described. Also, the research context and text classification procedures are outlined.

#### 3.1 Research design

The purpose of this study was to examine the ways in which NLP and machine learning techniques could be utilised to foster the development of TMS in virtual teams. This research addresses this problem by building and evaluating an IT artefact that pushes the boundaries of known applications of IT in business organisations. Therefore, it addresses vital problems previously considered not yielding to the algorithm-based approaches. This IT artefact is in the form of a prototype artefact that demonstrates the feasibility of addressing such a problem (March and Storey, 2008). In order to ensure the rigor required in conducting this research, I chose design science research (DSR) as the method for research.

After the initial stages of the formalisation of the aspects of the problem and the literature review, the follow up stage of artefact design and creation proposition process was mainly creative, thus abductive logic was suitable for this stage (Dresch et al., 2015). The design and development of the IT artefact as well as its evaluation were implemented by means of deductive reasoning. It follows that in these stages, I offered solutions based on the existing knowledge for building the artefact (Dresch et al., 2015). I used collected data from a real world virtual team mailing list archives coupled with data from a digital survey to eventually evaluate the artefact. The usefulness of the artefact was, to some extent, demonstrated by this evaluation. Building and evaluating artefacts are essential components of the DSR process (March and Storey, 2008). Lastly, inductive reasoning was followed to demonstrate that the building heuristics of the artefact are generalisable for similar types of problems. This DSR is accessible to both technology-oriented and management-oriented audiences. Together, these stages prove that the type of research method chosen was appropriate to address the research questions.

#### 3.2 Empirical settings

Data were collected from the archives of the electronic communications of two sources: (1) a programming Q&A site; and (2) a FOSS.

Sizeable corpora of labelled training data of each class are required for building an accurate text classification model. Often, such labelled text data is not readily available. The alternative of labelling manually is often difficult to organise, labour-intensive, tedious and the labelling is oftentimes of low quality, since the labels are not set in a realistic natural task context. Programming Q&A sites have question and answer sections containing code snippets and explanations written by programmers that can be used for labelling data as being related to a specific programming topic/skill or not, this reduces the manual labelling effort. In particular, Stack Overflow is commonly used to ask questions related to software development and debugging. These questions usually get useful answers thanks to the sizeable user base.

The Stack Overflow community guidelines define a tag as being a keyword or label that classifies a question with other related questions. When a member with the right privilege chooses to add a new tag, the topic must be unique. Privileges control what a member can do on Stack Overflow. A programmer gains more privileges by increasing their reputation (i.e. points received from other users for posting useful questions and answers). In this instance, the privilege type required for creating a tag is *Creation Privilege* and is awarded at 1,500 Reputation points. (Stack Overflow, 2017)

Stack Overflow has an extensive archive of already labelled Q&A, thus making it a compelling source for mining labelled data for our training data set. It has 14 million questions, 22 million answers and 49 thousand tags as of May 2017 (Stack Exchange, 2017). Such a large number of labelled data is ideal for accurate learning. Therefore, the Stack Overflow community is relevant for this study.

Developers engage in conversations in a multitude of discussion platforms. These platforms typically have diverse ranges of time constraints and facilitate either synchronous or asynchronous behaviour or both. Apache Spark community and its Issue Tracker which tracks bugs affecting the Spark software (Apache Spark, 2017b) is a good illustration of an asynchronous long-established discussion platform. The Issue Tracker is a collection Q&A that were created at different times and the conversations were archived.

Apache Spark is a free and open source framework for fast and general-purpose cluster computing. Apache Spark goal is to make data analytics fast to run and fast to write. It offers high level APIs (for Java, Scala, Python and R) and an optimised engine that backs

general execution graphs. It also provides a stack of higher-level tools such as ad hoc SQL and structured data processing, machine learning, graph analytics, and streaming analytics. (Apache Spark, 2017a)

(Zaharia, M. and Chowdhury, M., Franklin, M.J., Shenker, S. and Stoica, I, 2010) started Spark in 2009 at AMPLab, University of California, Berkeley. Spark was initially a research project for machine learning with big data. Spark become open source in early 2010 under a BSD license. UC Berkley gave away the project to the Apache Software Foundation in 2013. Then, Apache Spark became a top-level Apache project in February 2014 and remains one of the most active projects in Apache Software Foundation.

As of May 2017, Apache Spark has had more than 39000 commits made by 1445 contributors representing more than one million lines of code. Over the last year however, Apache Spark has experienced a considerable decline in its development activity as can be seen in Figure 5. This can indicate many things: it could be a telling sign that the developer's interest in this project is diminishing, or it might be a sign that Apache Spark's code base is mature enough and that it requires fewer bug fixes and modifications. (Open HUB, 2017)

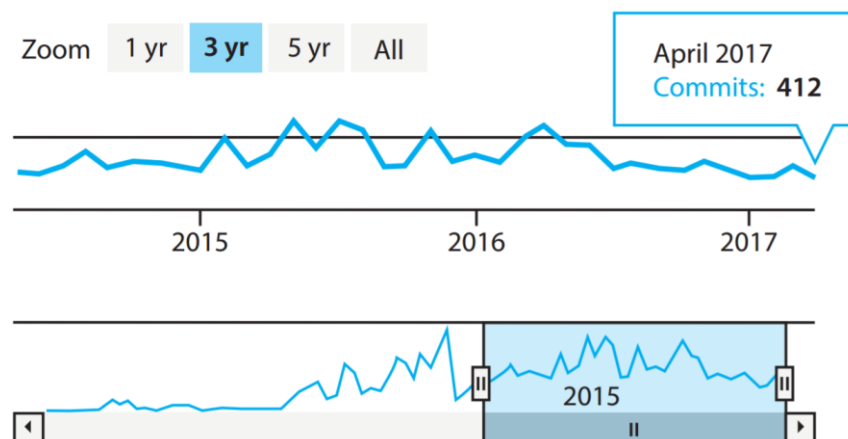


Figure 5 Apache Spark development activity: Commits per month (Open HUB, 2017)

Apache Spark is mostly written in Scala which makes it possible to have a concise API for users (Zaharia, M. and , Chowdhury, M., Franklin, M.J., Shenker, S. and Stoica, I, 2010). Therefore, proficiency in Scala is needed to understand and modify the core of Apache Spark. Java and Python are also used to a lesser extent.



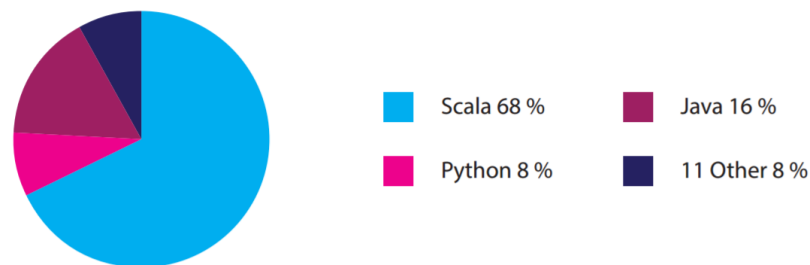


Figure 6 Programming languages used to write Apache Spark (Open HUB, 2017)

### 3.3 Data collection and pre-processing

Data preparation is an important and time-consuming phase in the data mining process. In the real world, data is not usually presented to us in the feature vector representation that the majority of data mining procedures take as input. To use most of the available machine learning algorithms, either data representation must be engineered to match the algorithms' input representation, or new algorithms are built from scratch to match the data's representation. Leading data scientists use both of these strategies. It is, however, in most cases, easier to first attempt to engineer the data to match the existing algorithms, since the latter underwent extensive testing and have been shown to work.

The following section describes the way in which the text of the FOSS mailing list archive was extracted, analysed and turned into a form that could be used by the machine learning analysis.

#### 3.3.1 Mining text

The challenging task was to get enough labelled data as quickly as possible for our machine learning algorithm. Usually, the practical way to address this challenge is to exploit an ongoing process where data is labelled by humans as part of their regular work or routines. In our instance, the process exploited was the labelling of Q&As by the Stack Overflow community, this action offer information about the classes used for the training data set. To build the database of the training and testing data to run the supervised machine learning algorithm on it, I text mined Stack Overflow and Apache Spark Issue Tracker.

Web scraping is systematically mining data from the Internet. Although web scraping can be done manually by a developer, the term normally refers to automated implementations done by using a bot or web crawler.

The legality of web scraping differs from country to country. Overall, web scraping can be forbidden by the terms of use (ToS) of some websites, but the enforceability of these terms is often ambiguous. Thus, it is necessary to review the website's ToS and respect the robots.txt file prior to initiating any scraping tasks.

For this research, I chose to use an open source web scraping framework called Scrapy, which is written in Python. I used it to extract the data from Stack Overflow with the help of selectors based on XPath and Regular Expressions. Scrapy settings help adhering to ethical scraping practices by not flooding the website with frequent requests over a short period of time. Otherwise, there is a high risk of being blocked from scraping the website permanently.

The scraped data was the body text of Q&A tagged “Scala”, “Java” and “Python” sorted by descending order of the number of votes. The output of each crawling job is summarised in Table 3.

*Table 3 Summary of scraped data from Stack Overflow*

<b>Tag</b>	<b>Number of scraped tagged Q&amp;A</b>	<b>Word count</b>	<b>Stack Overflow total tagged Q&amp;A</b>	<b>Margin of error (%) *</b>
Java	1143	584485	1276503	± 2,9
Python	1594	579910	771122	± 2,5
Scala	1147	369121	67381	± 2,9

\* 95% confidence level (The level of confidence in whether the true figure for the population lies within the confidence interval for the sampling)

Scrapy outputs a .json file for each crawling job, these are transformed into line separated .txt files which are in turn concatenated into a .txt file as seen in Figure 7.

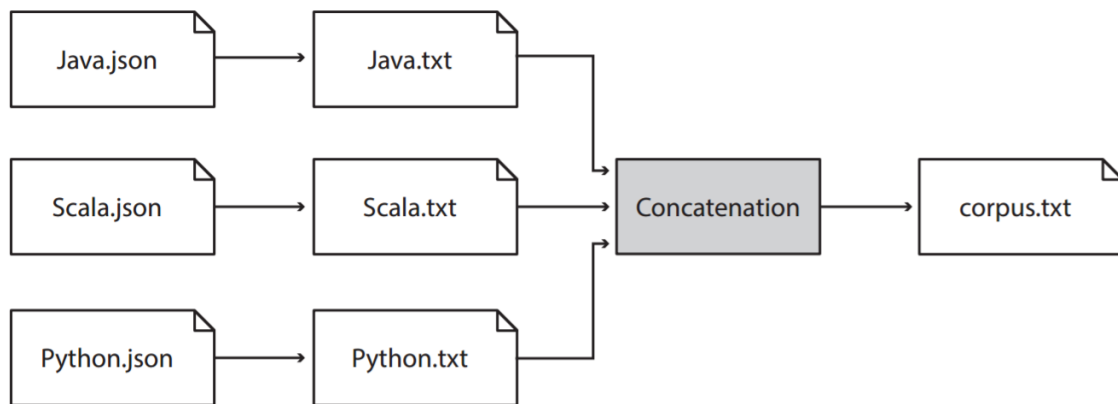


Figure 7 Workflow of building the training corpus

The concatenated corpus.txt file is given as an input to a python script for data extraction, the script takes the input path of the corpus from the user, extracts relevant data and writes it into a CSV file named corpus.csv. The output of the said extraction can be seen in Figure 8.

```

jami@jami-VirtualBox ~/SVMClassifier
File Edit View Search Terminal Help
(py27) jami@jami-VirtualBox ~/SVMClassifier $ python data_extraction.py
Enter the Path to the Corpus
corpus.txt
The corpus contains 3880 articles

Processing the corpus and writing it to CSV...

The training data has been loaded into corpus.csv

Time to extract data was: 18.7465019226 seconds
(py27) jami@jami-VirtualBox ~/SVMClassifier $
  
```

Figure 8 Terminal output for the data extraction process

To generate the target variable, raw mbox textual files were downloaded from Apache Spark Issue Tracker mailing list archives. Mbox is a generic name for a group of related file formats utilised for storing sets of email messages. All these messages are stored in concatenated plain text in a single mbox file.

Given the large number of emails, I sampled 10000 emails, out of a total of 161666 existing emails (as of May 2017). The emails selected were those posted by the top 50 most active contributors. This is a suitable sample to obtain the standard confidence level of 95% and margin error of  $\pm 0.95\%$ .

### 3.3.2 Data pre-processing

Text is usually termed “unstructured” data. This denotes the fact that textual data lacks the kind of structure that is normally associated with data, namely tables of records (i.e. collections of feature vectors) having links between them. Nevertheless, text is amply structured, but it is linguistically structured, meaning that it is intended for human understanding, not for machines. Text data is somewhat a dirty type of data (Provost, 2013). In FOSS mailing lists developers tend to type misspelled words, use ungrammatical structures, they concatenate words as one word, they use unpredictable abbreviations and random punctuations. In the rare cases where they write flawlessly, the text can include synonyms and homographs, therefore non-trivial to process.

In order to use all the mined unstructured texts in machine learning, it needs to be pre-processed. The first step was to filter out content generated automatically and which did not contain any actual natural language. Such content includes automatic confirmation emails sent by the web-based version control repository GitHub. These emails consist of source code and bug reports originating from other platforms. I manually inspected the sampled emails to discern the patterns these automated messages have and then proceeded to filter them out using regular expressions. Another issue was the duplicate contents inside emails resulting from quoting previous emails in a conversation. Only the original emails were kept for consideration and the quoted ones removed. Thus, reducing the size of the data to be processed.

I used a Python script to handle mbox files and convert them into plain text files. This script uses regular expressions to search for lines with special characters and keywords to filter out irrelevant content (i.e. source code, email headers and stack traces).

Also, a list of email addresses was retrieved from the processed mbox files. These collected emails addresses were used later to distribute the digital survey.

## 3.4 Survey

To validate my algorithm for the detection of the expertise of the team members, I conducted a digital survey. As discussed in the literature review, it is possible to reveal the development stage of the TMS in a group by using a survey. The web-based survey was sent to a total of 50 top contributing developers to assess the level of TMS

development within the Apache Spark community. The objective of the survey was to quantify the level of awareness each developer had about their peers' technical skills.

The measure (see Appendix A for the full set of questions used) was adapted from the 15-item scale by (Lewis, 2003) measuring the three constituents of TMS: specialisation, credibility, and coordination. Participants in the survey responded to each statement using a five-point scale (1 = strongly disagree, 5 = strongly agree).

The survey asks team members about their source of information and how they go about assigning responsibility for certain knowledge to other teammates. The survey is used to identify the specialist knowledge of the virtual team, the location of important information, the way in which the information is found, and the hurdles facing its location.

To quantify the level of development of the TMS in the studied virtual team and by following the collective method, TMS was first measured individually and subsequently the individual scores were collected into a team-level measure.

## 4 Data analysis and Results

I will present in this chapter the most relevant results from the empirical research according to the main aims stated in the research design. However, I will elaborate more on the classification process. The first section (4.1) presents the considerations that were followed when choosing the right classification technique for our case. In the next sections (4.2 and 4.3), I examine more specifically how the support vector machines and text feature extraction were used in our case. In the following section (4.4), the results of the data analysis, cross-validation and external validity analysis will be presented and I also summarise the whole data analysis in a table form. And finally, the survey report will be discussed.

### 4.1 Considerations when choosing the right classification technique

#### 4.1.1 Accuracy

While accuracy is an important indicator of a good prediction, it is not always necessary to achieve a high level of it. Often, an approximation is sufficient, this is determined by the use case. The approximate method has the advantage of shorter processing times and it tends to prevent overfitting (more on overfitting in next section), which is ideal for our case.

#### 4.1.2 Training time

The time it takes to train a model can vary widely from few minutes to many hours, depending on the algorithm used. Accuracy and training time, usually, affect each other. Moreover, certain algorithms tend to be oversensitive to the amount of data points more than other algorithms. The choice of the algorithm also depends on the time available, particularly if the dataset is very large. Our dataset is relatively small; thus, training time was not an issue. As a matter of fact, it takes just few seconds with the developed algorithm to train and predict on a new testing set.

#### 4.1.3 Linearity

Many machine learning algorithms utilise the linearity property (i.e. property of a linear function which is graphed as a straight line). For instance, linear classification assumes that classes are separable by one or more straight lines or planes. The algorithms that fall

into this category include logistic regression and SVMs. The assumption of linearity has no negative impact on accuracy in the case of textual inputs in SVMs but they can be problematic for certain problems.

#### 4.1.4 Parameters

The parameters are the way through which an algorithm can be fine-tuned. They are adjustable factors used to specify, for example, the penalty parameter of the error term, the loss function and the norm used in the penalisation. The more an algorithm has parameters the more it is flexible. Usually, the algorithms with numerous parameters need a lot of testing to achieve the best settings for their parameters. Optimal settings, in turn, have an impact on the accuracy and training time. In our case the number of parameters was low.

#### 4.1.5 Number of features

In some instances, the data at hand can contain a huge number of features. In the case of textual data, for example, there are as many features as there are terms. This high number of features can slow down certain machine learning algorithms and make the training time unrealistically long. SVMs are remarkable at handling data with a high number of features, making them a suitable candidate for text classification in our case.

#### 4.1.6 Summary

The choice of the right machine learning algorithm depends on: first, the data structure and the assumptions made regarding it; second, on the results sought. The learning model that best fits our textual data is SVM, which has the advantages of good accuracy, short training times and a limited number of parameters and is not memory hungry. Additionally, SVMs can divide classes faster and avoid overfitting more than most other algorithms. The chart in Figure 9 was used to help make this choice.

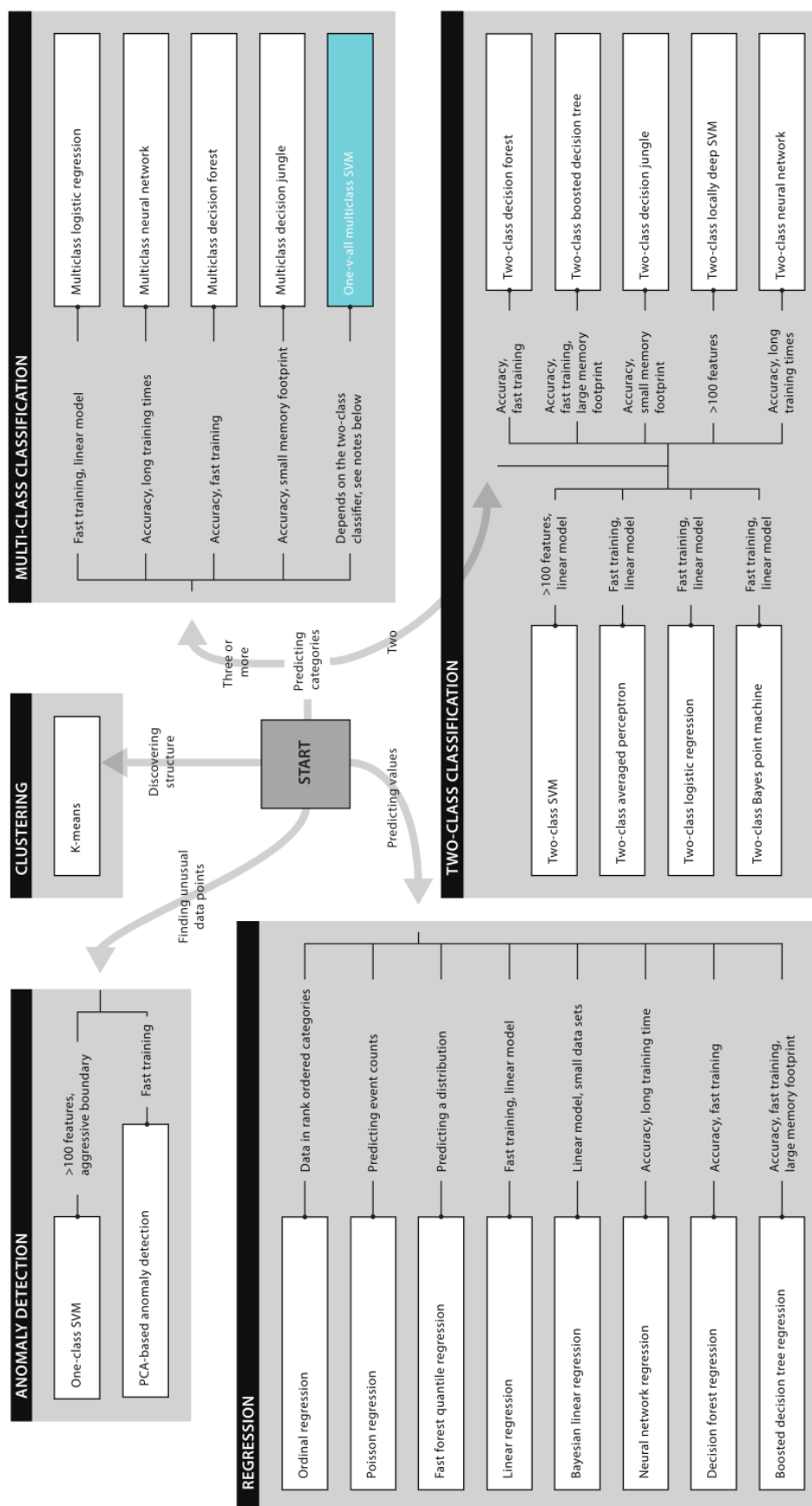


Figure 9 Chart guide for choosing the right machine learning algorithm



## 4.2 Support Vector Machines

If a sizeable corpus of text is at hand, then choosing any classifier would probably result in very similar classification performance and there might be no clear winning classifier accuracy-wise. In this case, it is advisable that the selection of a classifier be determined by its scalability and/or run-time efficiency. Roughly, as the training data doubles in size, the classifier's performance would linearly increase. However, with huge corpus size, the increase is sub-linear. I had at hand an acceptable amount of labelled data from Stack Overflow, thus I was in the perfect position to use an SVM.

Choosing which classification technique to use in our case depended also on the kind of patterns that I wanted to detect. Since the input data is a textual data with high-dimensionality (i.e. the number of variables is the number of words), SVM techniques are the most adequate in this instance. SVMs are even suitable in instances where the number of dimensions is higher than the number of samples. Cortes and Vapnik (1995) demonstrated that SVMs outperformed other techniques in handling this kind of data.

SVMs can be considered as kernel machines. Thus, their behaviour can be adjusted by specifying various kernel functions for the decision function, adding to their versatility. Very commonly used kernel functions include linear, polynomial, (Gaussian) radial basis function, and string kernels.

Additionally, since textual data is high-dimensional, linearly separating the data becomes less difficult. Accordingly, the majority of text classification problems are linearly separable (Joachims, 1998). Thus, the most applicable classifier for our type of data is a linear kernel.

scikit-learn offers an implementation of the linear kernel of support vector classification inside the `sklearn.svm` module. This class is called `LinearSVC` and can perform multi-class classification on a text dataset. `LinearSVC` focuses on classifying large amounts of documents and variables at a faster run-time than the standard `SVC`. These capabilities allow `LinearSVC` to be a good choice for text classification.

## 4.3 Text feature extraction

It is not possible to feed machine learning algorithms directly with raw textual data since the majority of these algorithms require a matrix of numerical feature vectors that

represent the text. The general process of creating this matrix is called text feature extraction.

Scikit-learn provides a module called `sklearn.feature_extraction` that performs feature extraction. When fed with raw text, it outputs a format accepted by machine learning algorithms. In order to do so, this module offers useful NLP functionalities, including:

- Tokenisation, which is the process of converting textual data into words, phrases or other significant items referred to as tokens and assigning an identifying number to each resulting token. White-space characters and/or punctuations are used as token separators to split text into individual elements.
- Occurrences of each token within every document are counted. Every token occurrence frequency is thus considered as a feature.
- Occurring tokens are normalised and weighted according to their importance in the documents.

By following these steps (i.e. tokenisation, counting and normalisation) I created a matrix that represent the text mined from Stack Overflow and Apache Spark. This matrix is called a Bag-of-Words, where textual data are represented simply by word frequencies regardless of the location of the word inside the document or its syntax.

A Bag-of-words or Bag of n-grams is a scheme used in NLP as a representation of documents as matrix of unigrams occurring in it. An n-gram is a sequence of contiguous words. For example, in Figure 10, are unigram, bigram and bag-of-words representations of the same sample text. Including features of higher n-grams in the feature vector increases the chances of detecting multi-word sentences occurring in the text.

The class `CountVectorizer` in the module `sklearn.feature_extraction` have been used in combination with an n-gram range of (1-2) to implement both tokenisation and occurrence counting in a single go.

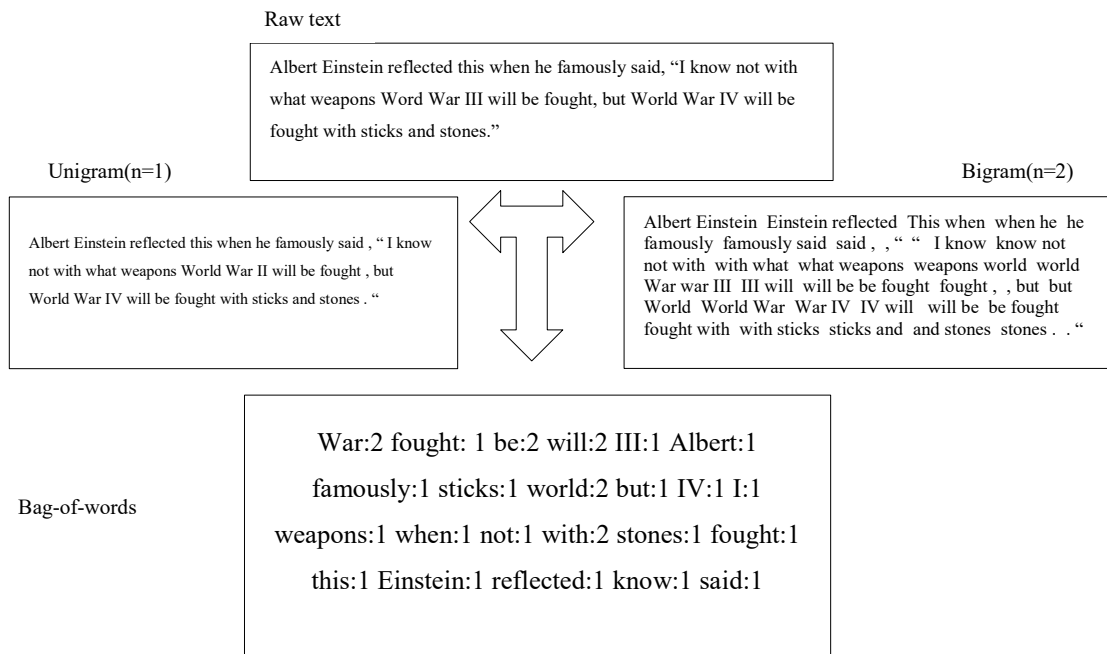


Figure 10 Sample text with unigram, bigram and bag-of-words

In sizable corpora of text, certain kind of words will be occurring very frequently (e.g. “they”, “an”, “and”, etc...), these stop-words thus are likely offering empirically insignificant information that cannot be used to distinguish between different classes of documents. Assuming that I fed the classifier directly with raw frequencies of occurrence of a token in a given document, those high-frequency tokens would overshadow the sparser yet more meaningful tokens that reflect the actual content of the documents.

To counterbalance the shadowing effect of these stop-words there are two strategies to follow, either outright remove them or use TF-IDF transform to reweigh the count data. scikit-learn implements this functionality through the class `TfidfTransformer`, inside the module `sklearn.feature_extraction`. `TfidfTransformer` transforms a given count data to a normalised TF or TF-IDF matrix. The purpose of utilising TF-IDF as a substitute to raw count features is to weigh-down the influence of terms that appear frequently in documents and which are therefore less descriptive than terms that appear sparsely in the training data.

The class `TfidfTransformer`, however, computes TF-IDF somewhat differently than the standard approach discussed earlier in the literature review. The following equations were used by scikit-learn to calculate IDF and TF-IDF:

$$IDF_{(t,d)} = \log (1 + n_d / 1 + DF_{(d,t)})$$

And

$$TF - IDF_{(t,d)} = TF_{(t,d)} \times (IDF_{(t,d)} + 1)$$

Another difference worth noting is that typically raw count data is normalised before computing TF-IDF, `TfidfTransformer`, however, normalises the TD-IDF directly.

#### 4.4 Training the Support Vector Machine classifier

Supervised machine learning is called classification when the data are utilised to predict categories. When, predicting two categories the classification is called binomial or two-class classification. When classifying instances into more than two categories then the classification is called multinomial or multiclass classification. The multiclass classification assumes that each data point is mapped to a unique label. In our instance, given the text documents making up our corpus, a multiclass classification is classifying them by labels: Scala, Java or Python.

Although few classification algorithms naturally allow for instances to be classified into three classes or more, the basic support vector machines, however, support only binomial classification. Yet, there are variety of strategies to extend binary classification to deal with multiclass classifications. The strategy adopted in our case is to reduce our multiclass classification problem to several binomial classifications. The methods proposed to reduce the multiclass problem into several binomial problems is also referred to as problem transformation methods. The problem transformation can be achieved by either a One-vs-One or One-vs-Rest schemes. In our instance, it is the latter scheme that is chosen because it is the most typically used scheme and is a reasonable default choice.

The One-vs-Rest is a scheme that fits a single binomial classifier on each individual class. Then, for each of these binomial classifiers, another fitting is done, this time the classes are assessed against all other classes in the model, as if it was an ordinary binomial classification case. Subsequently, predictions are carried out through these individual binomial classifiers, and the prediction that achieves maximum confidence score is then chosen.

The advantage of the One-vs-All scheme is that it is computationally efficient, since the number of required classifiers is n-classes. Additionally, this scheme is useful for

acquiring information about the classes by examining their mapped classifier, since every class corresponds to one and only one classifier. Basically, a collection of individual binomial classifiers is built and the resulting predictions are then aggregated to build a single classifier that predicts all classes as shown by Figure 11. Therefore, any binomial classifier can be utilised to create a One-vs-Rest classifier.

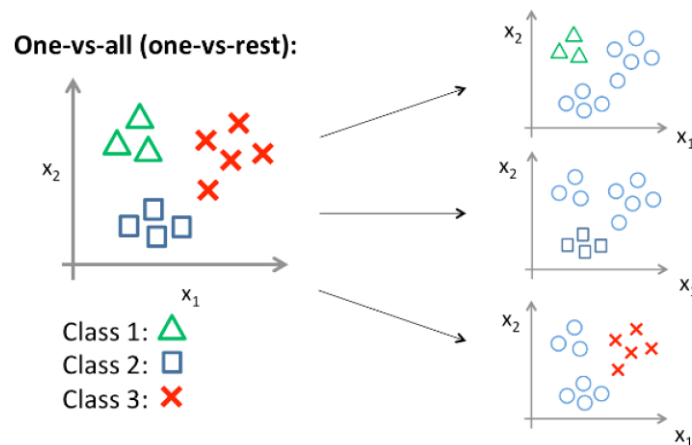


Figure 11 One-vs-Rest scheme

scikit-learn implements One-vs-Rest strategy through the class `OneVsRestClassifier`, as part of the module `sklearn.multiclass`. This class was used to configure a binomial LinearSVC SVM classifier which in turn was fed to the One-vs-Rest classifier. This classifier creates the binomial LinearSVC SVM for all the target classes and then runs the One-vs-Rest class to merge the resulting predictions for all the classes.

When training the classifier, one binomial classifier is trained per class. To do so, the multiclass labels are converted to binomial labels indicating whether it belongs or not to the class. To binarize the labels in a One-vs-Rest way, the class `LabelBinarizer` from the module `sklearn.preprocessing` is used. `LabelBinarizer` makes the conversion of labels handy with the `fit_transform` method. Finally, when making predictions, those classes whose associated classifier has the highest confidence score are assigned. `LabelBinarizer` implements this process through the `inverse_transform` method.

#### 4.4.1 Evaluation of the Classification's Performance: Cross-validation

I discussed previously, the steps followed to fit our model to the training dataset so that I can reliably extend the predictions made by the machine learning algorithm to new unseen data. To do this in practice, the already shuffled data is partitioned into two parts,

with each portion containing the same distribution of samples: 80% of the data is assigned to train the classifier and the remaining 20% is unseen by the classifier and is called a test set.

**80% training set** - this is the data for which the algorithm gets acquainted with the labels and which is fed to the training process to build the classifier.

**20% testing set** - this is the data that is kept hidden from the algorithm until after the training process finishes. It is used to compute metrics on the algorithm's behaviour. For each entry in the testing set an attempt to predict its *value* is made by using the built classifier then a comparison is made with the real *value*.

While this technique gives satisfactory results, it has two disadvantages: first the classifier is not trained and tested on all the samples in the dataset; second, there is a high risk of the model learning erroneous patterns that are valid only within the training set. This risk is referred to as overfitting.

In overfitting, a machine learning algorithm, instead of uncovering the underlying structure of the data, it points to the random error or noise within the data. Overfitting may occur when an algorithm is very elaborate, like when it has an excessive number of parameters compared to the number of data points. In such instance, the generated model performs poorly in predictions, because it reacts disproportionately at the slightest fluctuation in the training data set.

To prevent overfitting, while adjusting the classification's parameters, it is indispensable to hold out a portion of the data set for validation besides the training and testing sets. In this case, the training set is utilised to train the classifier, the I validate against the validation set, and lastly use the test set to get the model's performance estimations (e.g. accuracy, precision, recall and f-score). The validation set works as a hybrid between the training and test data sets as it is a training data used as well for testing, yet, it is not considered a part of the proper training nor test sets.

So instead of splitting the data into 2:8 ratio, I used a more advanced technique referred to as K-fold cross-validation. This technique splits the data into k portions, holds one and combines the remaining portions and use them as a training set while validating against the portion held earlier. This process is then repeated k times (i.e. the number of folds) as a different fold is held each time.

Table 4 Classification performance of different k-fold sizes of the dataset

Number of folds	% of Accuracy	Precision	Recall	F1-score	Support
2	87,78	0,91	0,88	0,88	1940
4	88,96	0,91	0,89	0,89	970
6	89,93	0,91	0,90	0,90	646
<b>8</b>	<b>91,33</b>	<b>0,93</b>	<b>0,91</b>	<b>0,91</b>	<b>485</b>
10	89,43	0,91	0,89	0,90	388
12	88,54	0,91	0,89	0,89	323

scikit-learn implements k-fold cross-validation through the class `sklearn.cross_validation.KFold`. This class was used to split the dataset into various values of k folds and cross-validate against them. The k value with the highest accuracy was chosen, which is in this case 8 folds as can be seen in Table 4. The `KFold` fold class generates k pairs of bitmasks—vectors of Booleans that are utilised to choose randomised parts of the dataset for the training and validation sets. Then the performance scores for every portion are averaged to obtain a fairly accurate approximation of the algorithm's overall performance.

#### 4.4.2 External Validity: Classification Performance in an Independent Test Set

The challenge in creating new predictive models is to build a model that is performing very well at predicting classes on completely new and unseen dataset. Thus, it is very crucial to utilise robust methods for training and evaluating the classifier on the existing training data set. As the reliability of the model's performance estimation increases, the confidence in the improved performance of the model increases as well. This in turn has a positive effect on the operational use of the built algorithm.

The ultimate test of the accuracy of the predictions made by the LinearSVC SVM classifier I built is how it performs on an independent test set which remained unseen by the algorithm so far. This set must be from a similar domain on which the algorithm has been trained, namely Stack Overflow Q&A body of text.

In essence, a classification's accuracy is the ratio of the number of correctly predicted classes to all predictions it made. This is the most commonly used evaluation metric for classifications. However, it is only appropriate to use when the number of data points in each class is approximately the same and that all the predictions made and the related prediction errors are equally significant, which is the case in our data set.

In practice, the accuracy of this classifier was calculated by using the class `sklearn.metrics.accuracy_score`. In the case of multiclass classification, the accuracy score is the same as the Jaccard similarity coefficient score. The Jaccard index assesses the similarity between two or more label sets. It is referred to as Intersection over Union since it is measured by the size of the intersection divided by the size of the union of two or more sample sets. It is useful when comparing a set of actual labels to their mapped set of predicted labels. For our LinearSVC SVM algorithm the achieved accuracy was 91,33%.

Accuracy should be considered just an initial assessment of the predictive power of the algorithm, as well as an intuitive measure for prediction. However, an algorithm can have a good accuracy but still be of no use due to the accuracy paradox.

In the field of predictive analytics, the accuracy paradox refers to the phenomenon by which a classifier with a given degree of accuracy can make better predictions than a classifier with greater accuracy. To further insure the high quality of the predictive power of our model, other metrics were computed such as precision, recall, F1-score and support.

scikit-learn offers a practical way to compute the aforementioned metrics all at once through the class `sklearn.metrics.precision_recall_fscore_support` which computes precision, recall, F1-score and support for each class.

The precision, also referred to as the positive predictive value, is the ratio of true positives to the total number of true positives and false positives. It indicates, intuitively, how the algorithm is good at not labelling a negative sample as being positive. In our case, it is the sum of the positive predictions made divided by the sum of positive label values predicted. The model I built achieved a precision of 93%. If it would have been a low precision it would have indicated that there is an excessive number of false positives.



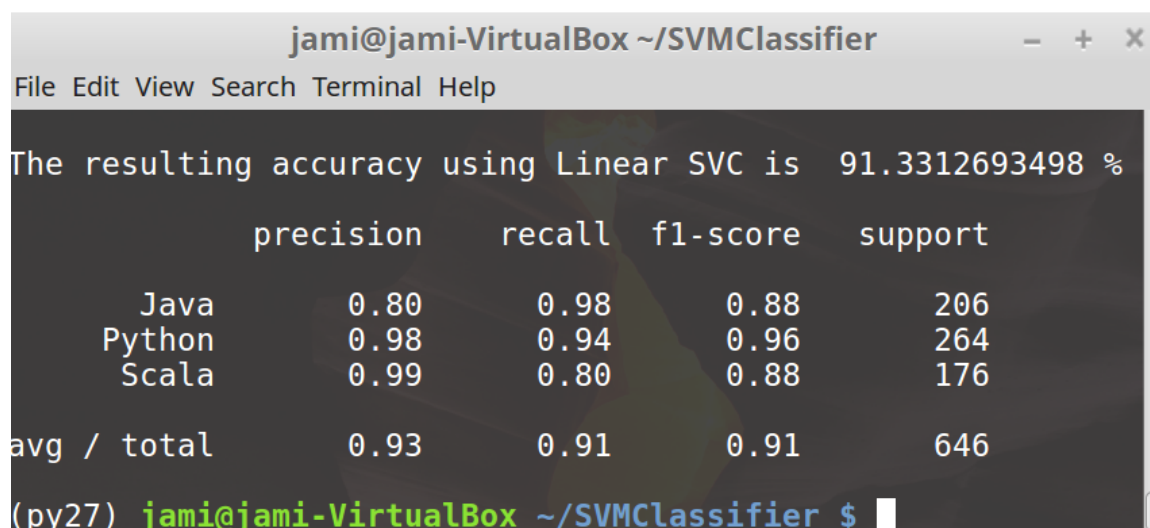
The following metric to compute was the recall. The recall, also referred to as true positive rate, is defined as the ratio of true positives to the total number of true positives and false negatives. It gives the intuition of how good is the algorithm at finding all the positive samples. In the case of our classifier, it is the total sum of the positive predictions made divided by the sum of label values in the testing data set. The LinearSVC SVM classifier achieved a recall of 91%. If it were a low recall value it would have indicated an excessive number of false negatives.

The final metric measured was the  $F_1$  score. This metric takes into account both the precision and recall, discussed above, to calculate the measure. The  $F_1$  score can be considered as a weighted average of the precision and recall. Traditionally, it is the harmonic mean of precision and recall and calculated as follows:

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

The  $F_1$  score can range from 0 (worst) to 1 (best). The classifier I built has an  $F_1$  score of 0,91.

The classification report, as shown in Figure 12, was outputted at the end of the training and in it figure al the aforementioned measures.



The terminal window shows the output of a classification report for a Linear SVC classifier. The window title is 'jami@jami-VirtualBox ~/SVMClassifier'. The output text is as follows:

```
The resulting accuracy using Linear SVC is 91.3312693498 %
```

	precision	recall	f1-score	support
Java	0.80	0.98	0.88	206
Python	0.98	0.94	0.96	264
Scala	0.99	0.80	0.88	176
avg / total	0.93	0.91	0.91	646

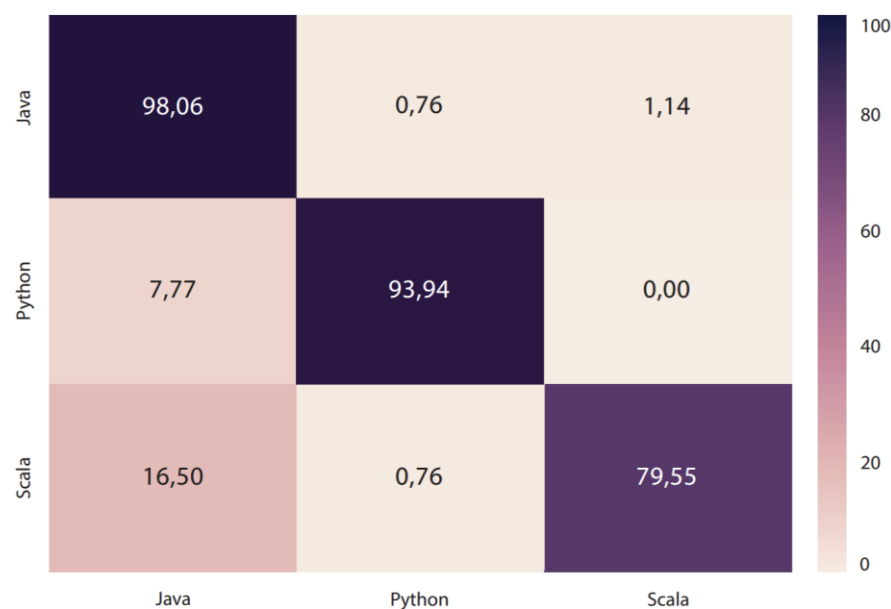
The prompt at the bottom is '(py27) jami@jami-VirtualBox ~/SVMClassifier \$'.

Figure 12 Terminal output for the classification

Also indicated in the classification report is the support, which is the number of occurrences of each class in the actual labels.

Using the classification accuracy and no other complementary measures may be misleading when the number of data points differs from class to class or when the classification problem is a multiclass case. To address this issue a confusion matrix was computed to show which predications the classifier got right and which were wrong. The confusion matrix is a convenient representation of the predictive powers of a multiclass classifier. It is also referred to as an error matrix, which has a table layout that summarises the correct and incorrect predictions of a supervised machine learning algorithm by class and with either count or percentage values. It is named a confusion matrix because it easily shows the ways in which the system confuses classes and labelling one as another when making predictions.

The class `sklearn.metrics.confusion_matrix` computes the confusion matrix for our classifier to evaluate its accuracy. Figure 13 shows a heatmap of predicted classes on the x-axis and actual classes on the y-axis.



*Figure 13 Heatmap of the confusion matrix (percentage)*

### 4.4.3 Summary

The following pipeline, shown in Table 5, was implemented in order to build the supervised LinearSVC multiclass classifier model for our computational framework.

*Table 5 Pipeline implemented for classification*

Step	Description of the Method	scikit-learn module
Pre-processing	Tokenising and removal of stopwords are handled by a high-level module that can build a dictionary of features and transform documents to feature vectors	CountVectorizer
Feature extraction	Transform a count matrix to a normalized TF-IDF representation	TfidfTransformer
Label transformation	Label Binarization	LabelBinarizer
Training the classifier	Train a classifier to make multiclass predications of the class of a document	OneVsRestClassifier (LinearSVC())
Building a pipeline	a module that acts like a compound classifier is provided to make (vectorizer=>transformer=>classifier) easier to work with	pipeline
Cross-validation	Cross-Validation	KFold
Evaluation metrics	precision, recall f1-score and support	precision_recall_fscore_support
	accuracy	accuracy_score
	Confusion matrix	confusion_matrix

## 4.5 Survey report

A digital survey was conducted at the end of the first phase of this research, and after collecting the relevant data about the Apache Spark virtual community members (i.e. username, email address and body of conversations). The digital survey was distributed to top contributing developers to assess issues related to the community's TMS. The survey has three objectives, the first one was to get results that would suggest a correlation between the results of our classifier's predictions and the actual skillsets of the participants. The second objective was to get feedback from participants on the usefulness of the computational framework developed by this research. The final objective was to assess the level of development of TMS in the Apache Spark virtual community.

Out of the 1687 contributing developers to the Apache Spark Issues Tracker 1068 had more than 2 emails from which 312 had more than 10 emails which in turn had only 60 contributors that had more than 50 emails. The survey was sent to these 60 top contributors, 47 emails made it to the final recipients from which only 4 responded.

Making a corpus from each contributor's emails is labour intensive and time consuming endeavour, it involves cleaning the data manually and matching the names of the authors with their email addresses. Due to time constraint, I couldn't go down the list and do this process to include more contributors.

Even though the response rate to the survey was low, here are highlights of the findings:

Members were asked to tell whether it would be valuable for them to be able to automatically identify the members with expertise in any given technology within their team.

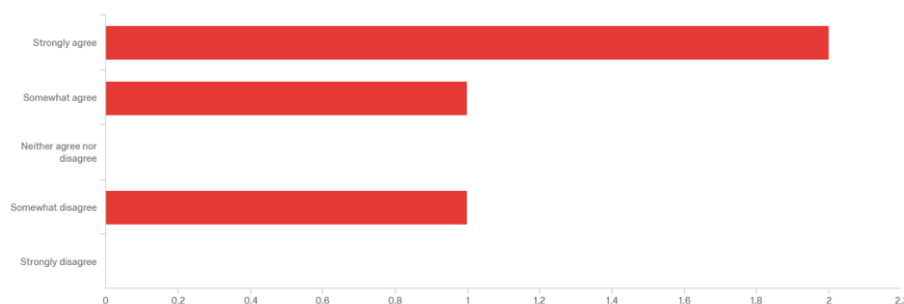


Figure 14 Answers to the question “Q4 - It would be valuable for me to be able to automatically identify the members with expertise in a given technology.”

According to the information collected through the questionnaire (See Appendix B for the full measure) most of the participants thought that a tool that helps with locating expertise within their virtual team would be valuable. However, this inference must be taken with caution and should be verified by future research in a more active community with members that are more willing to respond to surveys.

Next, the participants were asked to rate their expertise on top three programming languages that are used to write Apache Spark; namely, Scala, Java and Python. Half of the respondents said that they were experts in both Scala and Java, the other half said that they were either beginners or advanced in these technologies (Figure 15). The goal of this question was to find a correlation between the classifier's predications and the actual skills of the members of the virtual team. The results of the survey however are inconclusive. Moreover, for future research the case of a member being expert in more than one technology should be taken into consideration.

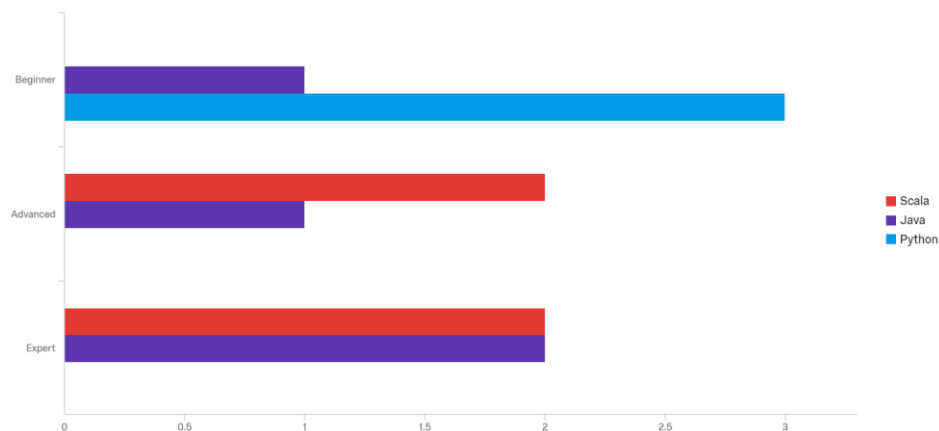


Figure 15 "Q2 - How would you rate your expertise in the following technologies?"

Individual scores were collected with Lewis' (2003) scale (Appendix B) according to the collective method, discussed in the literature review, which requires, first, that individual measures be gathered and then combined into a team-level measure (Cooke et al., 2000). Thus, after calculating individual scores the following step was to aggregate them to the group level. Aggregated team scores are presented in Table 6. Knowledge specialisation was, according to the scale, very high. However, coordination was poor between team members.

*Table 6 Team scores on three dimensions of TMS according to the scale adapted from Lewis*

<b>Dimension</b>	<b>Team aggregated score</b>
Specialisation	4,25/5
Credibility	3,91/5
Coordination	1,5/5

The aggregated scores specify the level of TMS development on three dimensions on a scale from 1 to 5. Since these dimensions are equally important for a well-developed TMS, it was concluded that TMS of the Apache Spark virtual team was not well developed. However, this deduction must be taken with caution and should be further verified by future research.

## 5 Discussion

In this section, the findings of this study will be discussed and put into their theoretical perspective. Moreover, the analytical process presented here could benefit an organisation in uncovering, describing and evaluating the quality of their TMS. It offers indications of the order of events in the development of TMS in virtual teams and functions as a tool from which the a more in-depth implementation of sociotechnical frameworks that support and improve organisational TMS could be derived.

### 5.1 Effects of virtualness on TMS

This research started with the proposition that a developed TMS is an essential microfoundation to dynamic capabilities that are important for an organisation to grow and sustain its competitive advantage. By offering support for TMS I intended to create awareness among team members regarding who knows what in their team.

The first research question asked: How TMS manifests itself in virtually distributed teams and how can we locate who knows what within these teams? This question was brought to my mind by the not so well understood assumed effect of distributed work arrangements on TMS development. It is more difficult for a virtual team to build its TMS than a traditional face-to-face team because of the diminished or even absent real interaction opportunities and the lack of contextual work information and physical colocation. Moreover, virtual teams' members cannot train collectively and this has a negative impact on the development of TMS (R. Moreland, L. Myaskovsky, 2000). Also, these members rely on a less rich media for their communication, such as email and chat which have less power to catch teammates' attention like would other visual cues do (i.e. gender, age, etc...). The lack of these real-world indicators results in difficulties in coordination between teammates since information about who knows what is "less visible". The absence of past collaboration between team members and lack of diversity in their expertise also limits the development of TMS in virtual teams.

Furthermore, the characteristics of dynamic structure and cultural diversity are having a negative impact on TMS in Apache Spark virtual community. Hollingshead (1998) notes that there is a negative correlation between the duration a team has spent together and the ability of a member of that team to locate the expertise of other teammates.

Repeating work-related exchanges between teammates have been proven by Lewis et al. (2005) to positively influence the development of TMS and the related knowledge transfer. TMS development depends, consequently, on a stable team composition. A virtual team like the one studied here tends to have varying team members through time which could lead to the disruption of the encoded knowledge, misallocation of where it is stored and non-coordinated retrieval.

Cultural diversity, like the one seen in the Apache Spark community has most probably an influence on the development of its TMS. Indeed, miscommunication and conflicts can arise from different cultural identities that may result in considerable loss in processes related to TMS development.

From all the above issues put together, the following is proposed:

**Proposition 1:** Teams operating in an environment subject to virtualness (i.e. geographically and temporally distant, computer mediated communication, changing membership and culturally diverse) will see the development of their TMS hindered.

## 5.2 Effects of the developed framework on TMS in virtual teams

Although I have founded my research based on the hypothesis that augmenting TMS artificially could benefit its development in virtual teams, thus far I have not examined the possibility of using externally derived expertise location to actually support TMS in this type of teams. To know who knows what and making this information easily accessible to anyone who needs it would possibly spawn substantial spike in performance for members of a virtual team. These members would then focus solely on other tasks instead of constantly trying to update the information about their needed external expertise sources.

R. Moreland, L. Myaskovsky (2000) set up an experimental examination of teams who were offered generated assessments of teammates' levels of performance that were derived externally. They showed through their experimentation that these teams were capable to achieve tasks as efficiently as teams who did train collectively to perform the task. This can be considered as strong indication that TMS can be artificially formed through technological means that would generate the building blocks of TMS. Indeed, Griffith et al. (2003) indicated that poor TMS development in virtual teams can be



mitigated to the degree to which information technology or organisational systems are utilised to support TMS development.

Consequently, the following proposition is suggested:

**Proposition 2:** The anticipated negative effect of virtualness on the development of TMS can be mitigated with a technological tool that assists virtual team members in knowing what others in their team know.

To achieve this objective, data were collected and analysed through machine learning and NLP techniques. I have predicted different classes of skills solely through the developers' contributions in the Apache Spark mailing lists. The results presented in Chapter 4 support the hypothesis that skillsets of members of a virtual team can be extracted and mapped.

To validate my algorithm, I conducted a digital survey. The survey has three objectives, the first one was to get results that would suggest a correlation between the results of my classifier's predictions and the actual skillsets of the developers. The second objective was to get feedback from the developers on the usefulness of an information technology tool that would assist them in locating expertise within their team. The final objective was to assess the level of development of TMS in the Apache Spark virtual community. However, the data from the survey is inconclusive due to very low response rate.

Finally, the findings of my research have also implications to managers. The ultimate goal of this research is to give tools to managers for detecting early signs of anomalies affecting TMS in a distributed work arrangement. Thanks to this computational framework, they could address those issues in a timely manner, before they affect the collaboration and productivity within their teams and consequently hinder one of the dynamic capabilities in their organisation.

## 6 Conclusion

Altogether, research on TMS in distributed work arrangements is at its development stage. This thesis represents research in the broad area of leveraging big data to benefit organisations. Specifically, it sheds light on innovative ways to use machine learning and NLP techniques to augment TMS in virtual teams by improving knowledge location.

This study developed a computational framework and a survey that use both mined text and, to a certain degree, answers from responding developers to uncover underlying patterns that can be employed to assist individual members in finding who knows what within their teams. This research designed and built algorithms that were used to extract data from open source communities and use them to identify and locate members' skillsets. In an organisational setting, this framework can improve employees' collaboration and productivity by supporting the search for experts within the organisation.

The main contribution of this research, however, lies chiefly in the development of a machine learning artefact that classifies textual data and labels them, and the use of that model to provide intelligent support for TMS.

The present chapter offers a summary of the outcomes of this study. First, I will discuss what was achieved. Second, I will reflect on the limitations and difficulties encountered throughout this research journey. Last, I will suggest research directions for the short and long term.

### 6.1 Summary of the accomplished work

The current study made the proposition and implementation of a computational framework for determining the skillset of individual members in virtual teams that use FOSS mailing lists as a mean for interaction. The application of this framework allowed me to offer intelligent support for the development of TMS through a novel algorithmic approach.

In the introduction of this thesis I stated three research questions, they were addressed as follows:

**RQ1:** How TMS manifests itself in virtually distributed teams and how can we locate who knows what within these teams?

I have (a) reviewed the literature about TMS in general and TMS in virtual teams then I focused on supporting one of the components of TMS which expertise location in virtual teams. (b) Skillsets have been chosen to inform a better tailored support adapted for expertise location in the Apache Spark community. I have formalised the input information to catch important data about team members and their conversations;

**RQ2:** How can we map the skillset of the individual members of a virtually distributed team? And how can we extract the information that represent these skillsets?

I have (b) developed algorithms that mine Stack Overflow Q&As for labelled data and (c) Apache Spark archived mailing lists for unlabelled data and (d) I developed a supervised machine learning algorithm to analyse and classify the unlabelled data based on the labelled data and NLP techniques. A total of 3100 labelled Q&As from Stack Overflow and 800 unlabelled text documents from Apache Spark Issues Tracker were used in this research. The results show that the approach adopted in the research classifies conversations with high accuracy, and that it is effective at indicating members' skillsets based on their conversation history.

**RQ3:** How would the developed computational framework support the development of TMS within virtually distributed teams?

I run a digital survey (e) among the members of the Apache Spark virtual team based on Lewis's field scale. Although the response rate was low, the partial results show that pinpointing who knows what can be beneficial for the development of TMS in virtually distributed teams.

## 6.2 Evaluation of the research and its limitations

The skillsets extracted were based on the mailing list archived data. This method let me capture a member's skill based only on the conversations between members of the virtually distributed team. Subsequently, I succeeded to predict a member's

programming skills that other members were not aware of. While using archived data from mailing lists in this way can have its benefits as we have seen, notably economy in time and labour, there are also limitations. For instance, all the data mined from the mailing list archives captured only the virtual interplay between members of the team. However, all the data about the real-world team members' interactions were totally missing from the mailing list. For example, there was offline team work when the Spark project has been created and transferred to Apache. Moreover, when new members join the team, there are no data relevant to these members' skillset in the archived mailing list. Therefore, skillset of that member would not be captured. Filling these gaps was one of the main reasons for using a survey at the end of the study.

A validation of the predictive powers of the computational framework have been made. To achieve this, I have employed a huge corpus of authentic data from Stack Overflow. Using already available authentic archival data is a common practice for validating supervised machine learning algorithms.

Another limitation of using archived data, which became apparent only after administering the survey, was the low response rate. Over the last year, Apache Spark has experienced a considerable decline in its development activity. This could be a telling sign that the developer's interest in this project is diminishing, or it might be a sign that Apache Spark's code base is mature enough and that it requires fewer bug fixes and modifications. I could not make any correlation with the results achieved by the algorithm since members of the virtual team were not responding. Choosing a more active project than Apache Spark would have been more judicious.

One more weakness of the digital survey is how the questions in the questionnaire (see Appendix B) were formulated. Team members were required to choose answers from a list as their reply to each question. A more suitable approach would have been to place checkboxes next to a member's identifier and let members choose the skill or skills they think were associated with that person.

I would also, point to the fact that Austin (2003) provides a survey setup for capturing TMS in teams, but unlike Lewis's scale, it measures the construct indirectly by comparing team members' assessments of competence areas. Actually, it is usually better to measure constructs indirectly rather than directly asking team members to assess their ability, knowledge, or behaviours.

## 6.3 Improvements and Recommendation for future research

The current research showed promising results in using NLP and machine learning techniques to support the development of TMS in virtually distributed teams. However, there are improvements to make to the current computational framework and the companion digital survey. The following directions for future studies are suggested:

### 6.3.1 Short-term research direction

#### 6.3.1.1 *Changing skills of the members of the virtually distributed team*

Even though the current research can pinpoint a member's programming skills by classifying their conversation into classes, the current computational framework does not consider the changing nature of a member's skillset through time. There are modelling technologies that can detect how expertise evolve in time. For instance Song et al. (2005) developed such a tool that dynamically describes and updates a member's expertise profile. But instead of using keyword extraction they used relational and evolutionary graph models that are used to mine, retrieve and visualise experts.

Similarly, in our case, functionality can be added to the supervised machine learning algorithm to describe the evolution of members' skillsets and the timestamps associated with these changes.

#### 6.3.1.2 *Add multi-labelling capability to the algorithm*

The algorithms developed for labelling the conversation's body of text can assign only one label per class, and this could be a problem. For example, when the member has many programming skills the algorithm can predict only one of those skills. A possible way to solve this issue is to consider using multiclass multilabel classification technique. In this way, a set of a member's skills will be described. A different approach would be to add to the current algorithm a functionality that shows all the labels with their prediction probabilities. This is possible by using `predict_proba` method from `scikit-learn`.

#### 6.3.1.3 *Increasing the sizes of the corpora and adding more classes*

In this research, for each of the three classes, an average of 1000 Q&A were used for training and another 200 were used for testing in the 8-fold cross-validation. Even though the relatively medium sizes of the training and testing sets, they are considered suitable

for supervised machine learning. Usually, the bigger the data the better. By utilising more Q&As for training and testing we would capture more data which would increase the classifiers' predictive powers. To address this issue, we need to assess by how much the classifier's performance metrics improve as the size of the corpus of Q&As for the training and test increases. It is very probable that there is a limit beyond which increasing the number of Q&As would not have any further positive impact on the performance of the classifier. Once that limit is reached I can rest assured that the predictions made by the algorithm reflect the actual skills of the members of the Apache Spark virtual team with very high confidence.

Additionally, it is important to add more classes for more programming skills, to make the results more generalisable and reflect the common real-world scenario where a developer masters many programming languages at once.

#### **6.3.1.4 Build an integrated expertise detection application**

The last part of the short-term research direction would be to compile the presently discrete components of the source code into a fully integrated and automated application. These components comprise the algorithms that mine Stack Overflow and Apache Spark mailing list archives to generate raw text from the Q&As and emails and also, the algorithm that performs the classification per se. This integrated application can be user friendly and provide the user with a graphical user interface (GUI) with options to filter members with a certain skill or a combination of skills.

### **6.3.2 Long-term research direction**

#### **6.3.2.1 Continuous monitoring of TMS**

As an extension of the approach adopted in this research, we can consider real time continuous monitoring of the virtually distributed teams for indicators of TMS by mining both unvarying and changing patterns that would permit the early detection of eventual expertise location anomalies with respect to TMS. This will inform the initiation of appropriate corrective measures. Moreover, it will be of interest to examine the phases through which a TMS in a virtually distributed team goes.

#### **6.3.2.2 Application in a different type of virtual teams**

To be able to apply the approach of this research in diverse types of distributed work arrangement is an interesting extension of this work. A community of researchers is one

potential applications of the present framework. Communities of practice are, also, an appealing possible use of this framework. It would be of interest also to consider whether this approach could assist communities of practice members and expand their practice. Particularly, when these members are geographically distant, it would be thought-provoking to study what benefits (if any) this framework could have on the TMS in such communities.

### **6.3.2.3 Including the other two components of TMS**

Lastly, the experimental design can be improved further to include all the dimensions of TMS, namely: Group knowledge stock; Consensus about knowledge sources; Specialisation of expertise and accuracy of knowledge identification.

## **6.4 Concluding remarks**

I had begun this research journey with the understanding that by offering support for TMS in a distributed work arrangement, I would be supporting a microfoundation of dynamic capability. Thus, assisting organisations in sustaining their competitive advantage. I have argued that by augmenting the TMS through machine learning algorithms and NLP techniques I would be facilitating expertise location and improve the experience of virtual teams' members by being aware of who knows what in their environment. Eventually, the goal of this type of research is to give tools to managers for detecting early signs of anomalies affecting TMS in a project. Thus, they could address those issues in a timely manner, before they affect the collaboration and productivity within their teams.

This research only scratched the surface of TMS analysis in FOSS archived mailing lists, therefore further research is necessary on other types of systems and tools used to facilitate the interaction of the members of a virtually distributed team. I am assured that future research on TMS would profit from the work presented in this thesis. This work proved that the approach presented can be deemed a valuable addition to the TMS research field. The range and variety of the aspects I researched and the knowledge I have acquired through examining numerous theoretical and technical areas resulted in a truly challenging and stimulating research journey.

## References

- Ackerman, M. S. and Malone, T. W.: Answer Garden: A Tool for Growing Organizational Memory, in *Proceedings of the ACM SIGOIS and IEEE CS TC-OA Conference on Office Information Systems*, pp. 31–39, ACM, New York, NY, USA., 1990.
- Agostini, A., Albolino, S., De Michelis, G., De Paoli, F. and Dondi, R.: Stimulating Knowledge Discovery and Sharing, in *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*, pp. 248–257, ACM, New York, NY, USA., 2003.
- Anand, V., Manz, C. C. and Glick, W. H.: An Organizational Memory Approach to Information Management, *Academy of Management Review*, 23(4), 796–809, doi:10.5465/AMR.1998.1255639, 1998.
- Apache Spark: Apache Spark, Apache Spark [online] Available from: <http://spark.apache.org/docs/latest/> (Accessed 30 May 2017a), 2017.
- Apache Spark: Apache Spark Mail Archives, [online] Available from: [https://mail-archives.apache.org/mod\\_mbox/spark-issues/](https://mail-archives.apache.org/mod_mbox/spark-issues/) (Accessed 30 May 2017b), 2017.
- Argote, L. and Ren, Y.: Transactive Memory Systems: A Microfoundation of Dynamic Capabilities, *Journal of Management Studies*, 49(8), 1375–1382, doi:10.1111/j.1467-6486.2012.01077.x, 2012.
- Austin, J. R.: Transactive Memory in Organizational Groups: The Effects of Content, Consensus, Specialization, and Accuracy on Group Performance, *Journal of Applied Psychology*, 88(5), 866, 2003.
- Boudreau, J. W. and Ramstad, P. M.: Talentship, talent segmentation, and sustainability: A new HR decision science paradigm for a new strategy definition, *Hum. Resour. Manage.*, 44(2), 129–136, doi:10.1002/hrm.20054, 2005.
- Brandon, D. P. and Hollingshead, A. B.: Transactive Memory Systems in Organizations: Matching Tasks, Expertise, and People, *Organization Science*, 15(6), 633–644, doi:10.1287/orsc.1040.0069, 2004.
- Continuum Analytics: Anaconda, Continuum Analytics [online] Available from: <https://www.continuum.io/anaconda> (Accessed 30 May 2017), 2017.
- Cooke, N. J., Salas, E., Cannon-Bowers, J. A. and Stout, R. J.: Measuring Team Knowledge, *Human Factors*, 42(1), 151–173, doi:10.1518/001872000779656561, 2000.
- Cortes, C. and Vapnik, V.: Support-Vector Networks, *Mach. Learn.*, 20(3), 273–297, doi:10.1023/A:1022627411411, 1995.
- Davenport, T. H., Harris, J. and Shapiro, J.: Competing on Talent Analytics, *Harvard Business Review*, 88(10), 52–58, 2010.



Davies, J. and A. Duke: OntoShare - An Ontology-based Knowledge Sharing System for virtual Communities of Practice, *JUCS*, 10(3), 262–283, 2004.

Dinsmore, T. W.: *Disruptive Analytics : Charting Your Strategy for Next-Generation Business Analytics*, Apress, [Berkeley, CA]. [online] Available from: <http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1174413&site=ehost-live> (Accessed 7 February 2017), 2016.

Dresch, A., Lacerda, D. P. and Antunes Jr, J. A. V.: *Design Science Research*, Springer International Publishing, Cham., 2015.

Griffith, T. L., Sawyer, J. E. and Neale, M. A.: Virtualness and Knowledge in Teams: Managing the Love Triangle of Organizations, Individuals, and Information Technology, *MIS Quarterly*, 27(2), 265–287, 2003.

Hamel, L.: Linear Decision Surfaces and Functions, in *Knowledge Discovery with Support Vector Machines*, pp. 49–59, John Wiley & Sons, Inc., 2009a.

Hamel, L.: Support Vector Machines, in *Knowledge Discovery with Support Vector Machines*, pp. 89–132, John Wiley & Sons, Inc., 2009b.

Hollingshead, A. B.: Communication, Learning, and Retrieval in Transactive Memory Systems, *Journal of Experimental Social Psychology*, 34(5), 423–442, doi:10.1006/jesp.1998.1358, 1998.

Hollingshead, A. B.: Perceptions of Expertise and Transactive Memory in Work Relationships, *Group Processes & Intergroup Relations*, 3(3), 257–267, doi:10.1177/1368430200033002, 2000.

Jarvenpaa, S. L. and Leidner, D. E.: Communication and Trust in Global Virtual Teams, *Organization Science*, 10(6), 791–815, doi:10.1287/orsc.10.6.791, 1999.

Joachims, T.: Text categorization with Support Vector Machines: Learning with many relevant features, in *SpringerLink*, pp. 137–142, Springer, Berlin, Heidelberg., 1998.

Jurafsky, D. Martin, J.H.: *Speech and language processing (Vol. 3)*, Harlow: Pearson Education., 2014.

KDnuggets: R, Python, Data Science software, [online] Available from: <https://kdnuggets.com/> (Accessed 30 May 2017), 2017.

Kleanthous, S. and Dimitrova, V.: Towards a Holistic Personalised Support for Knowledge Sharing in Virtual Learning Communities, *Conference on technology enhanced learning* [online] Available from: <http://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=2F101DC73A7C18F47E792FBE7138E3C4?doi=10.1.1.86.6836> (Accessed 15 June 2017), 2006.

Lewis, K.: Measuring Transactive Memory Systems in the Field: Scale Development and Validation, *Journal of Applied Psychology*, 88(4), 587–604, 2003.

Lewis, K.: Knowledge and Performance in Knowledge-Worker Teams: A Longitudinal Study of Transactive Memory Systems, *Management Science*, 50(11), 1519–1533, doi:10.1287/mnsc.1040.0257, 2004.

Lewis, K., Lange, D. and Gillis, L.: Transactive Memory Systems, Learning, and Learning Transfer, *Organization Science*, 16(6), 581–598, doi:10.1287/orsc.1050.0143, 2005.

Lindstaedt, S. N.: Towards Organizational Learning: Growing Group Memories in the Workplace, in *Conference Companion on Human Factors in Computing Systems*, pp. 53–54, ACM, New York, NY, USA., 1996.

Manning, Christopher D., and Hinrich Schütze.: *Foundations of statistical natural language processing.*, Cambridge: MIT press., 1999.

March, S. T. and Storey, V. C.: Design Science in the Information Systems Discipline: An Introduction to the Special Issue on Design Science Research, *MIS Quarterly*, 32(4), 725–730, 2008.

Marr, B.: *Data Strategy: How to Profit from a World of Big Data, Analytics and the Internet of Things*, Kogan Page, Limited., 2017.

Merali, Y. and J. Davies: Knowledge Capture and Utilization in Virtual Communities, *Proceedings of the First International Conference on Knowledge Capture*, 2001.

Mohammed, S., Klimoski, R. and Rentsch, J. R.: The Measurement of Team Mental Models: We Have No Shared Schema, *Organizational Research Methods*, 3(2), 123–165, doi:10.1177/109442810032001, 2000.

Moreland, R. L.: Transactive memory: learning who knows what in work groups and organizations, in *Small Groups: Key Readings*, Psychology Press., 1999.

Moreland, R. L., Krishnan, R. and Argote, L.: Socially shared cognition at work: Transactive memory and group performance., *What's social about social cognition*, 57–84, 1996.

Open HUB: Open HUB Spark development activity, Open HUB [online] Available from: <https://www.openhub.net/p/apache-spark> (Accessed 30 May 2017), 2017.

Provost, F.: *Data science for business: What you need to know about data mining and data-analytic thinking*, O'Reilly & Associates., 2013.

Python.org: Python, Package Index [online] Available from: <https://pypi.python.org/pypi> (Accessed 30 May 2017), 2017.

R. Moreland, L. Myaskovsky, (first): Exploring the performance benefits of group training: transactive memory or improved communication?, *Organizational Behavior and Human Decision Processes*, 82(1), 117–133, 2000.

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

Salton, G., Wong, A. and Yang, C. S.: A Vector Space Model for Automatic Indexing, *Commun. ACM*, 18(11), 613–620, doi:10.1145/361219.361220, 1975.

Scikit-learn: Scikit-learn, Scikit-learn [online] Available from: <http://scikit-learn.org/> (Accessed 30 May 2017), 2017.

Sebastiani, F.: Machine Learning in Automated Text Categorization, *ACM Comput. Surv.*, 34(1), 1–47, doi:10.1145/505282.505283, 2002.

Song, X., Tseng, B. L., Lin, C.-Y. and Sun, M.-T.: ExpertiseNet: Relational and Evolutionary Expert Modeling, in *SpringerLink*, pp. 99–108, Springer, Berlin, Heidelberg., 2005.

Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval, *Journal of documentation*, 28(1), 11–21, 1972.

Stack Exchange: Stack Exchange, Data Explorer [online] Available from: <http://data.stackexchange.com> (Accessed 30 May 2017), 2017.

Stack Overflow: Stack Overflow, Stack Overflow [online] Available from: <https://stackoverflow.com/> (Accessed 30 May 2017), 2017.

Teece, D. J.: Explicating Dynamic Capabilities: The Nature and Microfoundations of (Sustainable) Enterprise Performance, *Strategic Management Journal*, 28(13), 1319–1350, 2007.

Teece, D. J., Pisano, G. and Shuen, A.: Dynamic Capabilities and Strategic Management, *Strategic Management Journal*, 18(7), 509–533, 1997.

Wegner, D. M.: Transactive Memory: A Contemporary Analysis of the Group Mind, in *Theories of Group Behavior*, edited by B. Mullen and G. R. Goethals, pp. 185–208, Springer New York., 1987.

Wegner, D. M., Giuliano, T. and Hertel, P. T.: Cognitive Interdependence in Close Relationships, in *Compatible and Incompatible Relationships*, edited by D. W. Ickes, pp. 253–276, Springer New York., 1985.

Wong, E., Yang, J. and Tan, L.: AutoComment: Mining question and answer sites for automatic comment generation, in 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 562–567., 2013.

Zaharia, M. and , Chowdhury, M., Franklin, M.J., Shenker, S. and Stoica, I: Spark: Cluster Computing with Working Sets, *HotCloud*, 10, 95, 2010.

## Archived code repositories

Armel, J (2017): A web scraping tool built with Scrapy to scrape Q&A from Stack Overflow by tags and most votes. doi: <https://doi.org/10.5281/zenodo.833663>

Armel, J (2017): SVM-Classification: Multiclass Linear SVC SVM classification of text files into categories. doi: <https://doi.org/10.5281/zenodo.833655>

Armel, J (2017): A Python script that takes an mbox file and converts it into a text file. doi: <https://doi.org/10.5281/zenodo.833677>

## Appendix A: Transactive Memory Systems Survey

Based on the scale developed by Lewis (2003)

General questions:

1. How would you rate your expertise in the following technologies: Python, Scala, Java?  
(scale: Beginner, Advanced, Expert)
2. It would be valuable for me to be able to automatically identify the members with expertise in a given technology.

Specialisation questions:

1. Team members have specialised knowledge in specific aspects of the project.
2. I have knowledge about a specific aspect of the project that other team members don't have.
3. Different team members are responsible for expertise in different areas.
4. The specialised knowledge of many team members was needed to complete project related tasks.
5. I know which team members have expertise in specific areas.

Credibility questions:

6. I was comfortable accepting suggestions from other team members.
7. I trusted that other members' knowledge was credible.
8. I was confidently relying on the information that other team members provided.
9. When other team members gave some information, I wanted to double-check it for myself.
10. I had a lot of faith in other team members' expertise.

Coordination questions:

11. Our team worked together in a well-coordinated way.
12. The team members had very few misunderstandings when completing tasks.
13. The team members usually completed the tasks on the first attempt.
14. The team accomplished tasks smoothly and efficiently.
15. There was not much confusion about how we would accomplish the task.

NB: All items in this survey use a 5-point disagree-agree response (1=Strongly agree, 2=Somewhat agree, 3=Neither agree nor disagree, 4=Somewhat disagree, 5=Strongly disagree).