ARCADA

Predicting the Duration of User Stories

Machine learning for agile planning

Asif Raza

Master's Thesis
Master of Engineering - Big Data Analytics
December 5, 2020

MASTER'S THESIS	
Arcada University of Applied Sciences	
Degree Programme:	Master of Engineering - Big Data Analytics
Identification number:	
Author:	Asif Raza
Title:	Predicting the Duration of User Stories Machine learning for agile planning
Supervisor (Arcada):	Leonardo Espinosa Leal
Commissioned by:	

Abstract:
Effective effort estimation in project planning is vital, because it helps organizations to build product plans which they can stick to, have shorter turn-around time and better cost discipline. In this thesis, a series of supervised machine learning models were studied, analyzed and implemented to solve the problem of predicting effort estimate in Agile Scrum. The main approaches used were, Term Frequency - inverse document frequency (TF-idf), fastText, Neural Networks (Recurrent Neural Network (RNN), Long Short Term Memory (LSTM)) and Bidirectional Encoder Representations from Transformers (BERT). The models were fitted with two publicly available datasets. The fastText (with pre-trained model) significantly performed better in predicting the story-points of user-stories. The second-best performing model was bidirectional-LSTM. distilBERT performs poorly among all the models analyzed. This study can pave a way for organizations to benefit from these machine learning models and predict accurately the project deadlines and schedules. This could help organizations to get a head of their competitors and have happy customers.

Keywords:	effort estimate, agile, scrum, NLP, machine learning, fastText, TF-idf, BERT, distilBERT, bi-LSTM, user-stories
Number of pages:	53
Language:	English
Date of acceptance:	1.10.2054

CONTENTS

1 Introduction . . . 8

1.1 Motivation and Aim of the Study . . . 11
1.2 Definitions . . . 12
1.3 Data and Methods . . . 14
1.4 Structure of the Thesis . . . 17

2 Related work . . . 18

2.1 TF-idf for text classification . . . 18
2.2 LSTM and RHN for predicting story-points . . . 18
2.3 BERT for text classification . . . 20
2.4 Hierarchical Attention Networks (HAN) for Document Classification . . . 20

3 Research Methodology . . . 22

3.1 Data . . . 22
3.1.1 Data cleanup . . . 22
3.1.2 Splitting the Dataset . . . 24
3.2 Methods . . . 24
3.2.1 TF-idf . . . 24
3.2.2 fastText . . . 26
3.2.3 Neural Network . . . 28
3.2.4 RNN . . . 29
3.2.5 LSTM . . . 31
3.2.6 BERT . . . 32
3.2.7 Limitations . . . 33
3.2.8 Research Design . . . 33

4 Results . . . 35

4.1 Result of TF-idf . . . 35
4.2 Result of fastText . . . 37
4.3 Result of RNN . . . 40
4.4 Result of LSTM . . . 41
4.5 Result of BERT . . . 43
4.6 Result Comparison . . . 45

5 Discussion . . . 46

5.1 Conclusion . . . 47
5.2 Future work . . . 48

References . . . 49

FIGURES

Figure 1. Agile Scrum Framework (Wikipedia 2020) 8
Figure 2. User-Story view from Agile-Scrum tool (Yodiz 2018) 15
Figure 3. User-story's title, description and story-point in corpus 15
Figure 4. User-stories and story-points. 16
Figure 5. User-story length in words 17
Figure 6. Joining title and description of User-stories into one column 23
Figure 7. User-stories with regards to number of words in dataset. 23
Figure 8. fastText architecture (Joulin et al. 2016) 26
Figure 9. RNN architecture (Olah 2015) 29
Figure 10. RNN 3 steps overview (Kim 2017) 29
Figure 11. DAN model overview (Iyyer et al. 2015) 30
Figure 12. LSTM architecture (Olah 2015) 32
Figure 13. BERT overview of text classification (Devlin et al. 2019) 33
Figure 14. FI-Score of SVM model with TF-idf 36
Figure 15. SVM model validation accuracy confusion matrix 36
Figure 16. SVM model validation accuracy confusion matrix 37
Figure 17. fastText test accuracy - parameter approach 38
Figure 18. fastText (CV-5) Test accuracy- with pre-trained model 39
Figure 19. RNN validation and training accuracy of 100 epochs 40
Figure 20. RNN validation and training loss of 100 epochs 41
Figure 21. bi-LSTM training and validation accuracy of 100 epochs 42
Figure 22. bi-LSTM training and validation loss of 100 epochs 42
Figure 23. DistilBERT validation accuracy of 100 epochs 44
Figure 24. DistilBERT training and validation loss 44

TABLES

Table 2. Attributes in User-stories 16
Table 3. Number of words in user stories 23
Table 4. Dataset split and target classes 24
Table 5. fastText parameter and values 28
Table 6. List of libraries and tools used 34
Table 7. TF-idf results(SVM, LR) 35
Table 8. fastText best performing parameters 37
Table 9. fastText training and test accuracy - parameter approach 38
Table 10. fastText pre-trained model- accuracy result 39
Table 11. RNN training and validation accuracy results 40
Table 12. LSTM training and validation accuracy results 41
Table 13. BERT validation accuracy result 43
Table 14. Consolidated results view of all the models 45

ACKNOWLEDGEMENT

I would like to thank many people who were involved during this journey and without their support this thesis would have not been possible. My sincere gratitude to Dr. Leonardo Espinosa Leal, for supervising the project and for mentoring. His continuous motivation helped me to finally finish the project. I would also like to thank Dr. Magnus Westerlund for his support and guidance through out my studies. Last but not least, special thanks to my family and friends for all the support and help.

ABBREVIATIONS

API	Application Programming Interface
ATLM	Automatically Transformed Linear Model
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bang of Words
bi-LSTM	bi-directional Long Short Term Memory
CBoW	Continuous Bag of Words
CNN	Convolution Neural Network
Conv-GRNN	Gated Recurrent Neural Network
CPU	Central Processing Unit
CV	Cross Validation
DAN	Deep Averaging Network
DT	Decision Tree
FAIR	Facebook AI Research
GPU	Graphics Processing Unit
HAN	Hierarchical Attention Network
hs	skipgram hierarchical softmax
HTML	Hypertext Markup Language
IMDB	Internet Movie Database
IT	Information Technology
JSON	JavaScript Object Notation
k-NN	k-Nearest Neighbors
LD-RNN	Long Deep - Recurrent Neural Network
lr	Learning rate
LR	Linear Regression
LSTM	Long Short Term Memory
LSTM-GRNN	Long Short Term Memory - Gated Recurrent Neural Network
MMRE	Mean Magnitude of Relative Error
MSE	Measn Squared Error
n/a	Not Applicable
NB	Naive Bayes

NLP	Natural Language Processing
ns	skpgram negative sampling
PB	Product Backlog
PO	Product Owner
QA	Quality Assurance
RF	Random Forests
RNH	Recurrent Network Highway
RNN	Recurrent Neural Network
SM	Scrum Master
ST	The Scrum Team
SVM	Support Vector Machine
SW	Software
TF-idf	Term Frequency - Inverse Document Frequency
U-S	User-Story
URLs	Uniform Resource Locator
UX	User experience
ws	size of the context window

1 INTRODUCTION

In today's rapidly changing and competitive environment, organizations are looking for ways to be more flexible and efficient when comes to building products and services. Business leaders realize that the pace of competition is increasing, so they need to find ways to respond to change and deliver value faster to their customers. In response to these challenging environments, organizations are adopting Agile project management methodology to keep pace with customer demand and to develop efficient high-quality product faster (Figure 1). Agile is becoming a popular project management approach because of its emphasis on efficiency and quality (Ramin et al. 2020).

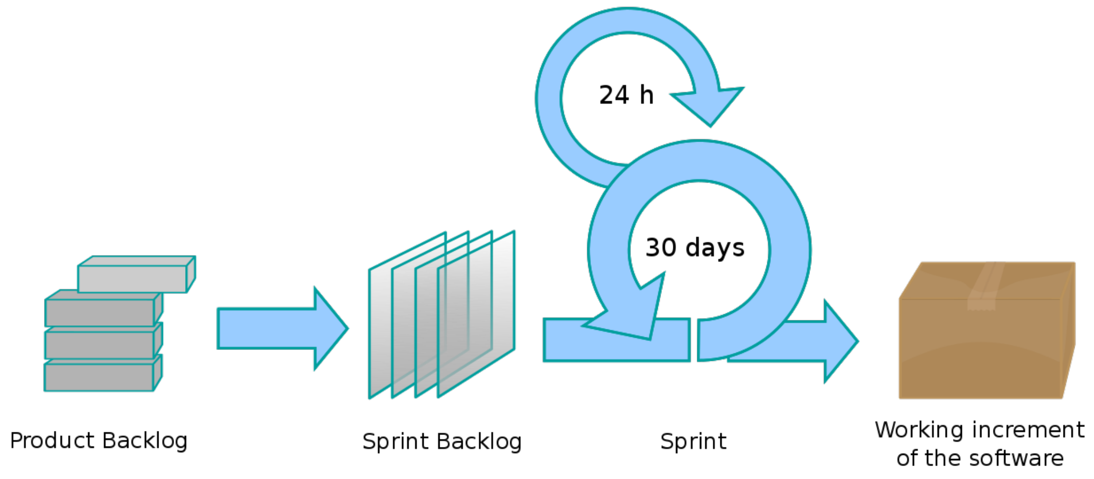


Figure 1. Agile Scrum Framework (Wikipedia 2020)

In the Information Technology (IT) sector alone, Gartner (2019) reported that around 87% of organizations are using Agile for all or some of their application development work. Agile is defined as a development approach that delivers software in increments by following the principles of the Agile Manifesto. The Agile Manifesto (Beck et al. 2001) was put together in 2001 by a group of experts who were looking for better ways to deliver the software (SW). Agile methodology consists of few frameworks: Kanban, eXtreme Programming (XP), Lean, Scrum, etc., but the most common one is Scrum (Andrei et al. 2019). It is a lightweight process framework for Agile development. Agile Scrum methodology has now become a de-facto standard in the Information Technology (IT) industry (ScrumAlliance 2018). Agile Scrum (Vargas et al. 2018) deal with the estimate and schedule of project planning, by answering questions like, how big is the effort? when it will be done? and when it will delivered? These questions circle around during life cycle of a project, but due to the iterative nature of Scrum, these questions are raised often, and team adapt to change.

In Agile Scrum, a project is first decomposed into smaller requirements to understand the bigger picture of architecture, user experience (UX), quality assurance (QA), development and deployment. This helps to understand the risk, duration, schedule and cost of a project. The smaller steps are taken to achieve the big picture. In agile, this work is done by using a series of repeated cycles (sprint) which is a short timebox iteration (ideal duration of sprint is 2-4 weeks). Project is sub-divided into small sub-projects which is represented by sprints. In each sprint, a potentially shippable product increment is delivered, which consists of design, implementation, testing and deployment stages. Thus, in every sprint new features represented by user-stories are added to the product, which results in the gradual project growth. The user-story selected in each sprint is based on business priority to make sure that most important user-stories are developed first.

As the user-stories are being tested and validated by the scrum team at the end of each sprint, they are likewise accepted by the customer. There is a high chance that end product is very close to user need. Scrum keeps a short customers feedback loop. At the end of each sprint, customer's feedback is collected which helps to keep the product, customer centric.

Determining effort estimate using story-point is a common practice in Agile Scrum. Story-point is a relative measure of the size of a user-story. Story-point is purely an estimate of time that will take to finish a user-story with quality product. For example, a user-story with 10 story-point is twice as big, risky and complex as a user-story estimated as five story-point.

Assuming a project's requirement is sub-divided into 20 user-stories in total. The project team, after careful analysis, estimates that it will take 4 sprints to deliver the project. The team will work on 4 user-stories on each sprint and after the 5th sprint, a finish product will be delivered to the customer. If the sprint duration is 2 weeks, it will roughly take 10 weeks (2.5 months) to finish the project.

Before starting each sprint, the project team meet to plan the work for the sprint and set its scope and objective, this meeting is called *sprint planning meeting*. The main characteristic of sprint planning meeting is to understand the user-stories well and estimate it precisely. So that user-stories which are actually planned for the current sprint can be finished within a desired sprint duration. The importance of sprint planning and estimation of a user-story is due to the fact that, effort estimate is concluded from the collective wisdom of the team. The *planning poker* is an effort estimation technique used in Agile Scrum. It combines expert opinion, analogy and dis-aggregation into an enjoyable approach, which results in a quick and reliable estimate. The Planning poker works because it brings together multiple expert opinion from cross-functional team members.

It is also a fact that, even after careful planning, projects do get delayed in Agile Scrum. Sometimes it is difficult to hold on to the original plan, as scope of the sprint changes due to additional unplanned work, unexpected bugs to be fixed from previous sprint or complexity of the work which was under estimated. These are few of the most common causes of sprint delays which the project team tries to optimise by rescheduling the sprint. The team in this scenario reduces the sprint scope to complete the work as sprint is time-boxed and tries to focus on high priority and high value tasks. This practice makes the sprint successful but impacts the overall duration of the project, as now it requires additional sprints to finish the work, which causes delays in projects. For example, If a team takes twice of the sprints originally planned to finish the work, the project delivery would be in 5 months not in 2.5, a 100% delay in the schedule. The underlying cause of project delay and cost overrun are mainly for two reasons: underestimating the complexity and overestimating the development team's productivity (McKinsey & Company 2017).

A research conducted by McKinsey & Company (Bloch et al. 2012) in collaboration with University of Oxford suggested that on average, large IT projects run 45% over budget and 7% over time while delivering 56% less value than predicted. Other study (McKinsey & Company 2017) indicated that, while analyzing more than 1,100 software projects, it was found that only 30% met their original delivery deadline, with an average overrun of around 25%. Delay in launching or delivering project means lost sales, advantage for competitor to get ahead, and potential long lasting damage to reputation and a very unhappy customer.

1.1 Motivation and Aim of the Study

IT projects are inherently unpredictable and complex due to their design and features. When embarking on journey to design complex products, organizations have often little idea how long the project will take, what the cost will be, and when would be the delivery to their customers. Initial project plans are based on intuition, analogy or a guess work.

The motivation of this thesis is to help organisations to be effective and precise in project planning when comes to effort estimation. Machine learning can play an important role in planning and estimating the project schedule. In Agile-Scrum, project teams leveraging the data from past sprints or projects to estimate the upcoming sprint, the effort estimates from team is often incorrect due to intuition, inexperience, over confidence or not understanding the complexity. This gives a perfect reason to apply machine learning approach in agile project planning. A machine learning model can be trained to learn the past historic data and use that knowledge to predict the upcoming sprints. It is important to understand that the data used in the stated scenario must have correct label which is the effort estimate, and it must be the real effort. If project is new and organisations don't have any similar data from other projects, open source data can be used provided that domain (mobile app, html, web services, etc.) is similar.

The aim of this thesis is to propose possible machine learning models to predict effort estimate based on current or past project data to predict the realistic project delivery schedule. This could help organizations build plans which they can stick to and have shorter lead time and better cost discipline. Estimating the project's effort correctly will potentially improve organization revenue.

In recent years, many research have been done in Agile Scrum on planning and effort estimation domain. This thesis is inspired by the latest research in this area and try to explore new trends. Machine learning research in project management and product design has come a long way. This research will investigate how Agile Scrum team can improve effort estimate during sprint planning by using machine learning techniques. The automated process of machine learning model can provide user-stories estimates based on past sprint data. The aim is to equip organizations with a tool, that can be used in their early project planning and during product development phase to provide realistic schedule.

This thesis will be dealing with supervised learning method, because the data is labelled. Its a multi class text classification problem, where text from the user-stories is used as input feature [X] and model has to predict value [y], which an integer value referring to the effort estimate of a user-story.

The research questions of the thesis are:

1. How to use current project data in estimating the project duration?
2. What machine learning techniques are most suitable for effort estimation in Agile Scrum projects?

The aim of this study is to answer these research questions and to set a path for future studies and applications. The thesis will conclude after analysing the existing work in this domain, and then experimenting with existing and new ideas to produce novel results. The other motivation of this research is to utilize the learning outcomes by making it available as a new feature in Yodiz Agile Project Management tool (www.yodiz.com) for predicting effort estimate in Agile Scrum projects.

1.2 Definitions

This section covers all the definitions, terminologies and concepts used in this thesis. Agile Scrum is a simple framework and it is easy to implement. The core of Agile Scrum framework is composed of **three roles**: i) Product Owner (PO), ii) Scrum Master (SM), iii) The Scrum Team (ST), **two artifacts:** i) Product Backlog (PB), i) User-Story (U-S), and **three ceremonies:** i) Daily Scrum, ii) Sprint Planning, iii) Sprint Demo

Product Owner: Primary duty of the PO is to make sure all team members are pursuing a common vision for the project. To set priority for the work, so that team knows the project priority. PO is also responsible for the return on investment on project.
Scrum Master: Main task of Scrum Master (SM) is to facilitate the work without having any authority to set project priority. Main task of SM is to remove impediments faced by the team and to protect the team from external influence and distraction so that team can focus on work prioritized by PO.
Scrum Team: Agile Scrum team is a self-organized team, comprises of members from multiple domains (architect, UI/UX, development, QA, etc.). Scrum team works autonomously and without much supervision. The scrum team is self-organized team whose main task is to achieve the goal set by PO in the most efficient way possible.
Customer: The customer is the one who funds or buy the product. If SW build for internal use, customer is usually from another group or division.
Product Backlog: it's a list of items relating to a product in priority order. The items can be a new feature, change request of existing feature, bugs and non-functional requirements. The items in backlog are always kept up to date and with right priority order.
User-Story: Is a short, simple description of feature of product.
Daily Scrum: It is a 15 min daily stand-up meeting, before team starts the day. This meeting is to synchronize team members about each other's work. Each team member answer three questions Q1: What I did yesterday? Q2: what is the plan for today? and Q3: What problems I am facing?
Sprint planning: Is an important ceremony to plan the upcoming sprint. SM facilitates the meeting and the team makes sure that they understand the requirement well, ask any question from PO to clear the doubts and estimates the effort of each user-story using planning poker.
Sprint Demo: The Sprint demo is a ceremony held at the end of each sprint, where team demo the work done during the sprint to customer/stakeholder. PO, together with customer accepts or rejects the user-stories. Sprint demo concept is to get feedback from the customer on work which is done so far on product before moving on to new features.
Planning Poker: In the planning poker, each team member is given a deck of cards of Fibonacci series. After a user-story is explained and discussed, each team member selects

and shows the card representing his or her estimates. All cards should be shown at the same time by all team members. This step is repeated until agreement on the estimate is reached.

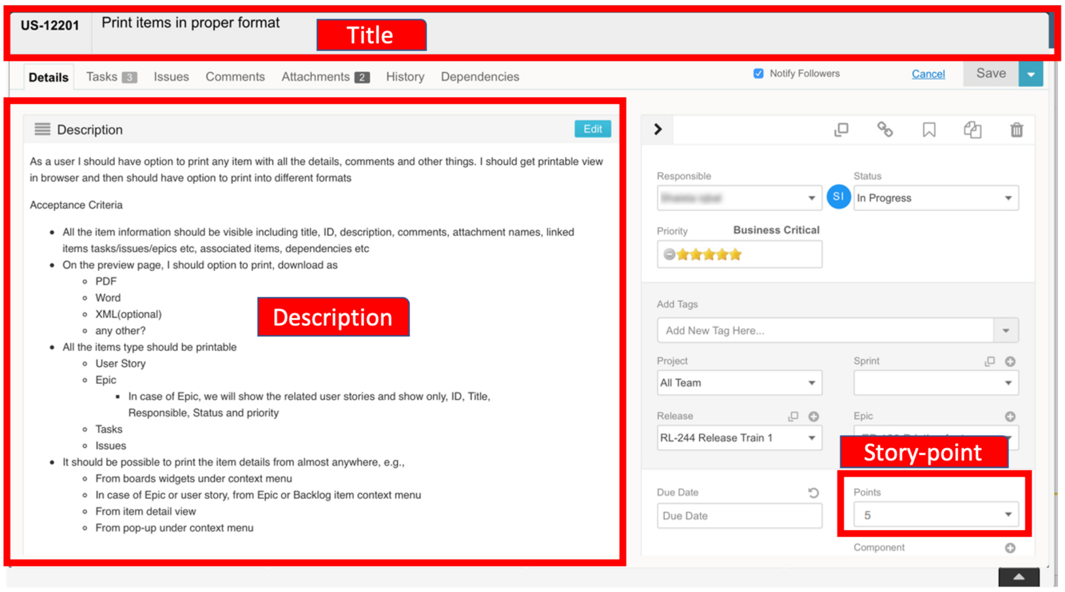
Story Point: Story point is a unit of measure for expressing the overall size of the user-story. Numbers used for story-points are from Fibonacci series, commonly used are 1, 2, 3, 5 & 8. These numbers associate with the size of a user-story.

1.3 Data and Methods

The data used in this thesis are from open source projects. These are user-stories of functional and non-functional requirements. All the user-stories have title, description and commit-logs. The attributes of user-stories are describe in Table 2. All the user-stories have story-point which is the label. Figure 3 shows the data in Pandas (Table 6) data-frame, the three columns represent the title, description and story-point in data-frame.

The story-point in the dataset is the real effort, estimated by the team regarding a user-story, so they are the actual duration in which a user-story was completed in a sprint. The Figure 4 shows user-stories count in dataset and their story-points. The Figure 2 shows the user-story view from Agile-Scrum tool (www.yodiz.com).

The dataset used in this thesis comprises of data used in earlier similar studies (**?**Porru et al. 2016) respective git repository (Choetkiertikul 2017), additionally some more data collected from other public git repository (RandulaKoralage 2020). The data used in this thesis is from 25 different projects which was fetched from 3 stated sources. The format of the data is csv or excel format. The length of the text in user-stories is important, as longer the text, larger the requirement could be. Figure 5 shows the word count in user stories



US-12201 Print items in proper format

Notify Followers

Description

As a user I should have option to print any item with all the details, comments and other things. I should get printable view in browser and then should have option to print into different formats

Acceptance Criteria

- All the item information should be visible including title, ID, description, comments, attachment names, linked items tasks/issues/epics etc, associated items, dependencies etc
- On the preview page, I should option to print, download as
  - PDF
  - Word
  - XML(optional)
  - any other?
- All the items type should be printable
  - User Story
  - Epic
    - In case of Epic, we will show the related user stories and show only, ID, Title, Responsible, Status and priority
  - Tasks
  - Issues
- It should be possible to print the item details from almost anywhere, e.g.,
  - From boards widgets under context menu
  - In case of Epic or user story, from Epic or Backlog item context menu
  - From item detail view
  - From pop-up under context menu

Responsible: [redacted]

Status: SI In Progress

Priority: Business Critical ★★★★★

Add Tags: Add New Tag Here...

Project: All Team

Sprint:

Release: RL-244 Release Train 1

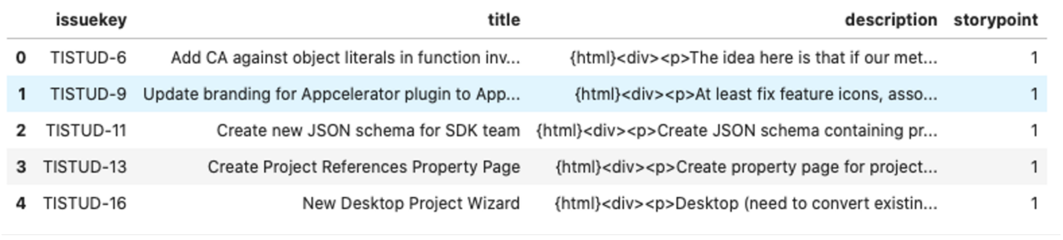
Epic:

Due Date: Due Date

Points: 5

Component

Figure 2. User-Story view from Agile-Scrum tool (Yodiz 2018)



	issuekey	title	description	storypoint
0	TISTUD-6	Add CA against object literals in function inv...	{html}<div><p>The idea here is that if our met...	1
1	TISTUD-9	Update branding for Appcelerator plugin to App...	{html}<div><p>At least fix feature icons, asso...	1
2	TISTUD-11	Create new JSON schema for SDK team	{html}<div><p>Create JSON schema containing pr...	1
3	TISTUD-13	Create Project References Property Page	{html}<div><p>Create property page for project...	1
4	TISTUD-16	New Desktop Project Wizard	{html}<div><p>Desktop (need to convert existin...	1

Figure 3. User-story's title, description and story-point in corpus

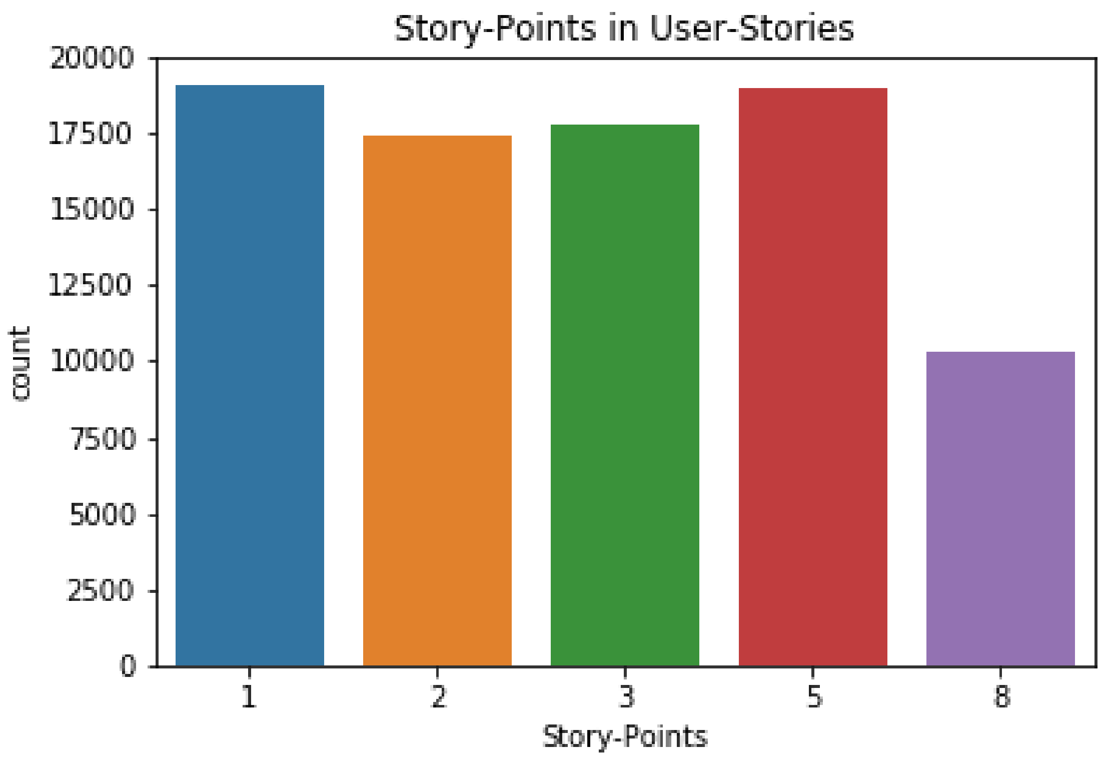


Figure 4. User-stories and story-points.

Table 2. Attributes in User-stories

User-Story Attribute	**Explanation**	**Data Type**
Title	1-3 line of text describing the title	String
Description	Describing requirement or enhancement in detail	String
Commit-Logs	SW code commit	String
Story-Point	Fibonacci series [0,1,2,3,5,8,..], which describes the effort estimate of work	Integer

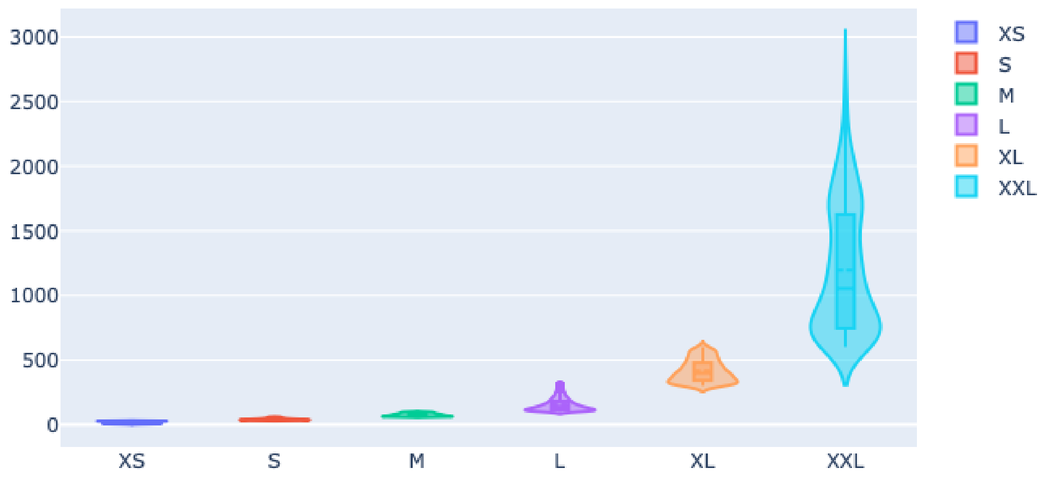


Figure 5. User-story length in words

1.4 Structure of the Thesis

This thesis is divided into four chapters. The chapter two presents, new concepts, current research and related work in this area. Chapter three will cover the methods, techniques and algorithms used to solve the problem. In chapter four, results of the experiments will be presented. Finally, chapter five provides the discussion and conclusion of the results along with closing remarks and future work.

2 RELATED WORK

2.1 TF-idf for text classification

In the study made by Porru et al. (2016), the authors proposed the Term Frequency - inverse document frequency (TF-idf) based model for estimating the story-point in Agile Scrum project for issue reports. The evaluation was done on one industrial and eight open source projects. The authors used attributes from issue reports like "Title", "Summary" and "Description". TF-idf is used to create a vector representation by converting each issue report as a row in matrix, where each word in a issue report is represented as wight in the matrix. On the TF-idf vector following models were used: Support Vector Machine (SVM), k-Nearest Neighbor (k-NN), Decision Tree (DT), and Naive Bayes (NB). The authors claim that the reason for choosing SVM, k-NN and DT is because, they are generally well suited for text classification applications. Model evaluation was done using Mean Magnitude of Relative Error (MMRE) and the accuracy. The 10-fold cross-validation was used to avoid over-fitting.

Scott & Pfahl (2018) further extended Porru et al's study by investigating if more features can be utilized along with TF-idf feature. They claimed to use extra feature on top of Porru's work and called it developers' features. Scott & Pfahl described the developers' features as the contribution and achievement of developer (development team member) towards the completion of project. Results from their study showed that the model based on developers' features outperformed the model with text only features (Porru's work). They also used the same method for evaluation as used by Porru et al. (2016) but used only SVM model, as it was proven well in Porru's study.

2.2 LSTM and RHN for predicting story-points

Choetkiertikul et al. (2019) stated that deep learning neural network is very effective in predicting story-points for user-stories. The authors proposed the predicting model by combining two deep learning architecture: long short-term memory (LSTM) and recurrent highway network (RHN). Data from 16 open source projects was used, which included around 25000 user stories.

LSTM (Hochreiter & Schmidhuber 1997) which is a special variant of RNN was used in this work. It has advantage over RNN because it stores accumulated information over time and works well on sequential data. LSTM is an ideal choice according to the author, due to its short-term memory cell, a vector that stores accumulate information over time. The information stored in the memory is refreshed at each time step, by accepting and refreshing new input and partially forgetting old and irrelevant information.

LSTM uses the forget gate [f_t] to control how much information from the previous context [c_{t1}] from the memory cell should be removed. The previous output state [h_{t1}] based on forget gate and current state, produce output value [0] or [1] where [1] indicates that all the past memory is preserved while [0] means completely forget everything. The [i_t] gate control which new information will be stored in memory.

In this approach, LSTM was using word-embedding as an input layer, which is a low dimension vector space, size of vector is the vocabulary size in corpus. This model also unitizing pre-trained model which was trained using project data without labels. After LSTM is trained, author used Random Forest (RF), Support Vector Machine (SVM), Automatically Transformed Linear Model (ATLM) and Linear Regression (LR) to validate the network. Results show that LR performed better, followed by SVM. Authors further continued by adding one additional layer as Recurrent Highway Network (RHN) on top of LSTM. RHN is a improved version of feed-forward neural network to achieve deep representation of model by enabling learning in very deep networks with hundreds of layers (Srivastava et al. 2015), which traditional neural network can not achieve (Pham et al. 2016). That was the main motivation to use RHN behind this approach as mentioned by the author.

Results showed that the use of recurrent networks highway (RNH) on top of LSTM did not improve the performance of the model significantly when compared with the LSTM alone.

2.3 BERT for text classification

In another research aiming for text classification, González-Carvajal & Garrido-Merchán (2020) compared the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. 2019) against the traditional machine learning approach the TF-idf (Porru et al. 2016, Scott & Pfahl 2018). The authors described that BERT model has arisen to popularity in recent year as start-of-the-art machine learning model which is able to cope with the multiple NLP task, specially the text classification.

Authors started with following hypothesis: i) how BERT perform on different sizes of data? ii) how BERT handle different languages (English, Chinese, etc.)? iii) and how BERT performs on data from different domains?. To test these hypothesis, different kind of data sets were used. There were four data sets: i) IMDB movie reviews, ii) RealOrNot tweets written in English, iii) Portuguese news dataset (the classified advertisement in seven different classes) iv) Chinese hotel reviews for sentimental analysis. After performing four different NLP scenarios on the data sets, Gonzalez's study claimed that BERT has outperformed the traditional NL approach the TF-idf, adding empirical evidence of its superiority over classical methodologies.

2.4 Hierarchical Attention Networks (HAN) for Document Classification

In the study, Yang et al. (2016) proposed the Hierarchical Attention Network (HAN) for document classification. Hierarchical structure mirrors the structure of a document, as document has three level of hierarchy (words-sentence-document): document has words, these words combine together to create sentence, and sentences create a document. HAN architecture consist of several parts i) word sequence encoder ii) word level attention layer iii) sentence encoder iv) and, a sentence-level attention layer.

This study tested the hypothesis that how to incorporate the knowledge of document structure in the model architecture for better representation of document. The authors described that not all parts of a document are equally relevant or important. Determining the sections that are relevant, involves modeling the interactions of the words, not just their presence in isolation. Primary contribution of this research was a new neural architecture, the Hierarchical Attention Network (HAN).

Since a document has hierarchical structure as mentioned earlier, the authors constructed a document representation by first building representations of sentences and then aggregating those into a document representation. The model proposed in this research included two level of attention mechanisms, one at word level and one at the sentence level. This let the model to pay more or less attention to individual words and sentence when constructing the presentation of the document.

To test the hypothesis, the HAN architecture was tested against other baseline methods like SVM, LSTM, word/character based CNN and with several baseline methods. These baseline methods and results are reported in (Zhang et al. 2016, Tang et al. 2015). Experimental results demonstrated that the HAN model preformed significantly better than previous methods.

3 RESEARCH METHODOLOGY

This chapter describes the methods and algorithms used in this thesis. Different NLP approaches were analyzed to solve the story-points estimation problem in agile projects. The goal of this chapter is to provide comparisons, advantages and dis-advantages among the different approaches used. This thesis will be using following approaches: TF-idf, fastText, Neural Network RNN, bi-LSTM and distilBERT.

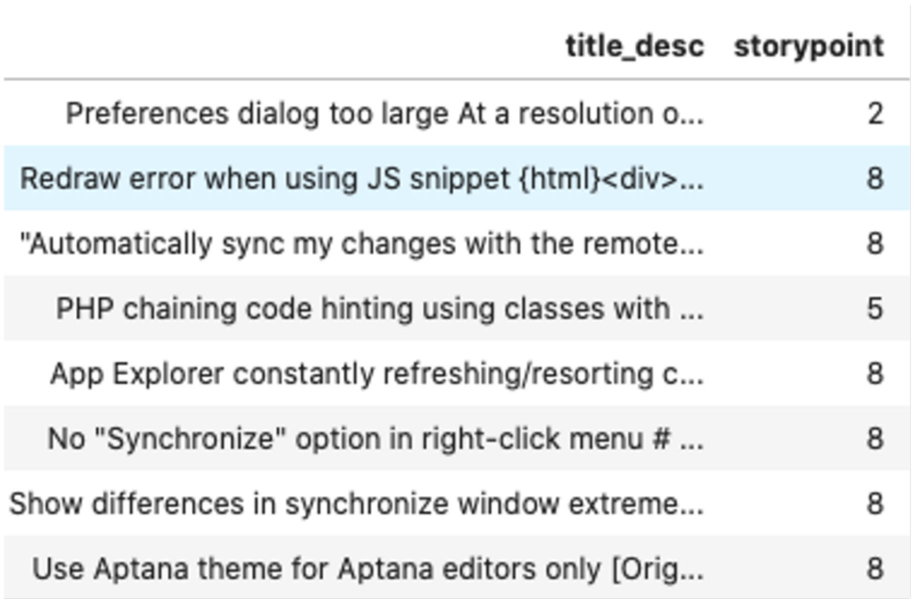
3.1 Data

This section will discuss in detail the data cleaning process and the steps used to prepare the data for the prediction models.

3.1.1 Data cleanup

Data cleansing steps are important aspects in machine learning specially when dealing with text data. Special characters, null values and other values can seriously affect the model performance. So first step was to analyze the data and prepare it for predictive modelling. By analyzing the data, it was realized that there were a number of items in the data which needed to be removed. These items consisted of NaN values, special characters, stop-words, single characters, numbers, URLs, HTML and JSON tags, punctuation, etc. After cleaning all of these items, data was converted to lower cases. "Title" and "Description" of user-stories were combine together (Figure 6), to have a single input as feature [X].

Other observation regarding dataset was that, it has story-points from 1-100 scale, its due to the reason that sometimes team just put any large number in the beginning regarding the effort estimations. Story-points in the data are grouped along the values 1, 2, 3, 5 & 8. These effort estimate represents the classes in our data, which the model needs to predict. Table 3 shows the user-stories and the word count. Figure 7 shows percentage of user-story size based on word count.



	title_desc	storypoint
	Preferences dialog too large At a resolution o...	2
	Redraw error when using JS snippet {html}<div>...	8
	"Automatically sync my changes with the remote...	8
	PHP chaining code hinting using classes with ...	5
	App Explorer constantly refreshing/resorting c...	8
	No "Synchronize" option in right-click menu # ...	8
	Show differences in synchronize window extreme...	8
	Use Aptana theme for Aptana editors only [Orig...	8

Figure 6. Joining title and description of User-stories into one column

Table 3. Number of words in user stories

Total Words in User-Story	User-Story size	No. of User-Stories
10-30	XS	38883
31-60	S	27493
61-100	M	8677
101-300	L	8677
301-600	XL	1032
601 and >	XXL	594

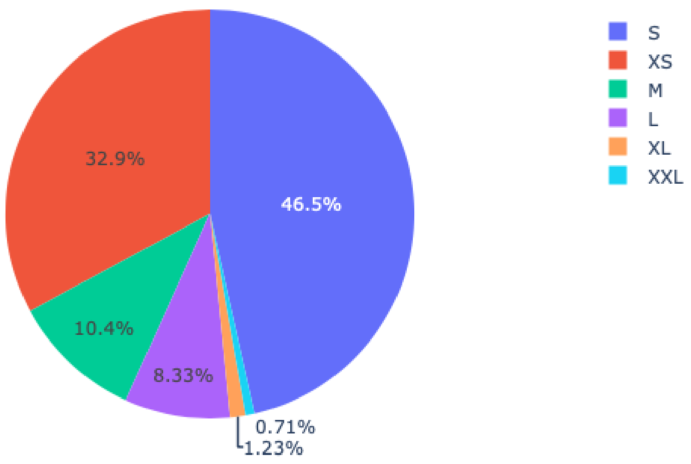


Figure 7. User-stories with regards to number of words in dataset.

3.1.2 Splitting the Dataset

Finally, dataset is split randomly with the ratio of 70% training set and 30% test set (Table 4). There are five target classes. Training dataset is used for model training and test dataset is used for model validation.

Table 4. Dataset split and target classes

Target Class	Total data	Train data	Test data
5 classes(1,2,3,5,8)	90,000	85000	15000

3.2 Methods

3.2.1 TF-idf

Term Frequency - inverse document frequency (TF-idf) is a very common algorithm to transform text into a meaningful representation of numbers in form of a vector. This vector representation is then used to fit machine learning algorithm for prediction. TF-idf is a proven classical approach in NLP for text classification. This thesis applied a method proposed by Porru et al. (2016), the goal is to predict the story-point using the "Title" and "Description" provided in text format. TF-idf is a simple way to calculate the 'score' or a 'weight' of the words in a document relative to a corpus. This will give each word based on its occurrence a weighting factors which is represented as wight and a score within a corpus. The application of TF-idf varies from spam filtering, sentiment analysis, key word search, recommendation systems, etc. TF-idf comprises of two components: TF and idf. TF (Term Frequency) represents the number of times a term/word appear in all the document in a given corpus. So higher the occurrence of a word in a document within corpus, higher the TF score would be.

$$TF = \left(\frac{\text{Number of repetitions of word in a document}}{\text{Total number of words in a document}}\right) \tag{1}$$

The second part of TF-idf is idf (Inverse document frequency), it is a logarithmic inverse fraction coefficient. This value decrease if frequency of a word in a corpus increases. It is calculated by diving the total number of documents by the number of documents containing the term and then taking the logarithmic of that value.

$$idf = log\left(\frac{\text{Number of documents}}{\text{Number of documents containing the word}}\right) \tag{2}$$

$$\text{idf}(t, D) = \log\frac{|D|}{1 + |\{d \in D : t \in d\}|} \tag{3}$$

$$w(t,d) = tf(t,d) \times idf(t,D) \tag{4}$$

number of documents where the term t t appears

In Eq. 3 the |D| is described as the total number of document, the [t] is the term which appear in the document [d]. Eq. 4 provides the score of each word in corpus, if the frequency of occurrence of a term is too high, it will be minimize or become zero by calculating the inverse frequency of the term, that is the reason denominator is adjusted to plus one in Eq. 4.

TfidfVectorizer (Pedregosa et al. 2011) will be used to process the raw text to a matrix of TF-idf features. Different n-gram values will be used to build the TF-idf matrix. In the matrix each row represents a user-story in corpus and each value in a row of a matrix represents the score of the word in a user-story.

To use TF-idf matrix for predicting the story-point in user-stories, two prominent machine learning models were chosen, which produced good results in stated study.

a) Support Vector Machine

Support Vector Machine (SVM) is very successful in statistical learning theory. SVM (Joachims 1998) represents data in high or infinite dimensional space. It learn the hyperplane that can separate the data points. The learned hyperplane is used to predict an output of the new (unknown) point. SVM is very effective in text classification data, specially when data is very high dimensional. The number of dimensions is higher then number of samples, meaning highly sparse data.

b) Logistic Regression

Logistic regression (Menard 2002) is a regression model, where the dependent variable is categorical which allow it to be used in classification also. Its a generalise logistic regression to multi class problem. Multi-nominal Logistic Regression predicts the probabilities of the various outcomes of categorically distributed dependent variable using a combination of independent variables of any class.

3.2.2 fastText

The fastText (Bojanowski et al. 2017) is a machine learning library for "text classification" and "word representation". The fastText library was released by Facebook AI Research (FAIR) in late 2019. It helps to create a vector representation of word in both supervised and unsupervised learning method. High level architecture if fastText is shown in Figure 8. This method provides both CBoW & Skip-gram (Mikolov et al. 2013) word embedding and support both word and character n-gram tokens as input to the model. The main advantages of fastText is speed and competitive performance. fastText exploits subword information to construct word embeddings. Word representations is created by first learning character n-grams representations and then summing them up to get the word representation. Bojanowski et al. (2017), argued that popular models, that learn

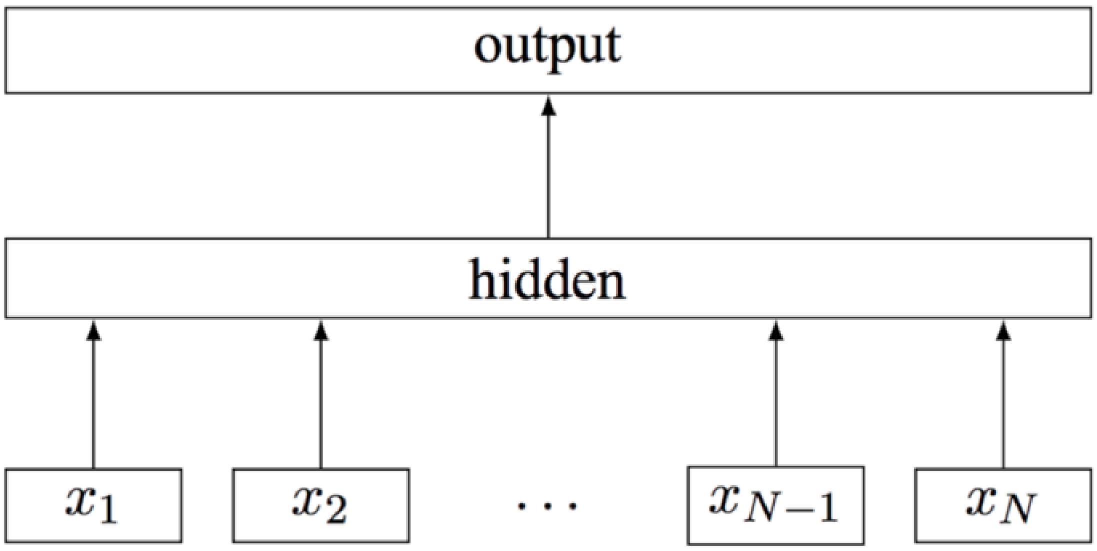


Figure 8. fastText architecture (Joulin et al. 2016)

word representations ignore the morphology of words. As in French or Spanish, most verbs have more then forty inflected forms, Finnish language for e.g has fifteen cases for nouns, which makes its difficult to learn good word representation, fastText proposed vector representation for morphologically rich languages by using character level information.

Internals of fastText

1. Labelled data with text provided for classification to fastText.

2. In fastText CBoW or skip-gram can be chosen for creating token, which can be further used to create matrix for learning embedding.

3. fastText is internally a shallow neural network where the embedding layer (CBoW or skip-Gram) is fed to the hidden layer, which act as a lookup layer for each word embeddings.

4. Text representation as a hidden state that can be shared among features and classes.

5. The embedding layer of text representation is fed to the classifier (multi-nominal logistic regression)

6. Then the softmax is applied to the outer layer to compute the probability distribution over the predefined classes.

Two of the different approaches of fastText as proposed by Joulin et al. (2016) will be applied in this thesis: fastText with Hyper parameter and fastText with Pre-trained model.

i. fastText with Hyper parameters

First approach is to tune parameters values provided by fastText, the default parameter values of fastText performs well, but to optimising the model performance, additional values and parameters are used as shown in Table 5. Following parameters will be used for tuning the values: i) learning rate (lr), ii) N-gram, iii) number of epochs, iv) window size (size of the context window) and v) loss. These stated parameter values are initialize in fastText by looping through different parameter values. There are total of 324 ($lr \times$

$3 \times ngram \times 4 \times epoch \times 3 \times ws \times 3 \times loss \times 3)$ iterations each provided with different accuracy based on parameter and values used, then the high performing parameter values are picked on which the model performs best.

Table 5. fastText parameter and values

Parameters name	Values
lr (learning rate)	0.05, 0.1, 0.2
n-gram (word)	1, 2, 3, 5
epoch (number of epochs)	5, 10, 15
ws (window size)	5, 10 , 15
loss	ns, hs, softmax

ii. fastText with Pre-trained model

Second approach is to use pre-trained model, the pre-trained model is trained using project's data without labels. Data used for training the model is not used in either train or validation process. There is also a possibility to use pre-trained model trained on wikipedia data or google's pre-trained model on books. But since model trained on domain specific data generally performs better, that is the reason to train own model using fastText. The advantage of having pre-trained model is to reduce the training time and have a better quality vector. That is why we build pre-train the model with domain specific data with fastText and used it as input.

3.2.3 Neural Network

Neural networks are widely used in building language models like finding POS tagging, language translation, word similarities, auto correction and suggestion, text generation, etc. Neural network/Deep learning approach is also getting very popular in solving text classification problem like sentimental analysis, email spam detection, question answering, news categorization, etc. Neural network is the other approach that will be utilized in this thesis.

In this work three types of neural network will be analysed:

- RNN

- LSTM

- BERT

3.2.4 RNN

Recurrent Neural Network (RNN) (Goodfellow et al. 2016) is a special type of neural network suitable for processing sequential data. It sees text as a sequence of words and memorise the sequence for text at each time step for classification. Figure 9 shows high level architecture of the model. The main characteristic of RNN is to memorize the previous stage, that is the reason it can predict the future stage based on past memory.

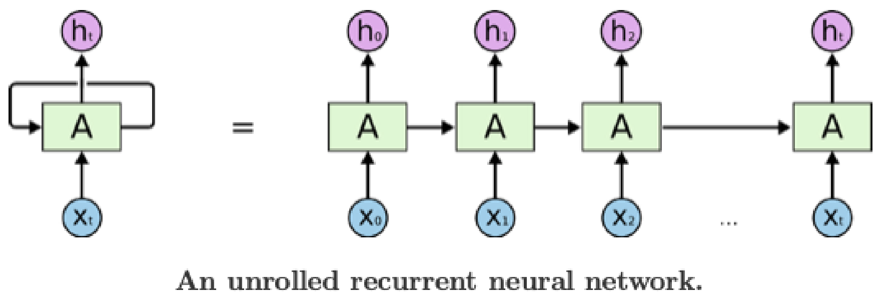
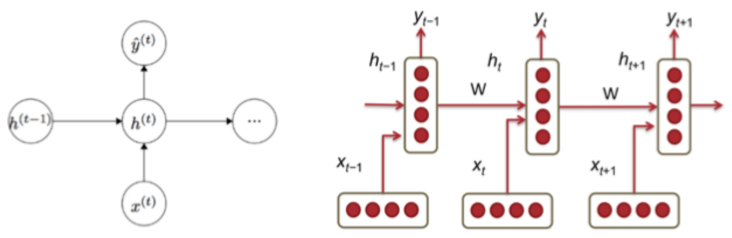


Figure 9. RNN architecture (Olah 2015)



$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\ \hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(\boldsymbol{o}^{(t)}), \end{aligned}$$

Figure 10. RNN 3 steps overview (Kim 2017)

This thesis applied similar technique as proposed by Iyyer et al. (2015). Deep Averaging Network (DAN), is the continuous bag-of-words (CBoW) or neural bag of words representation, computes a sum or mean of word embeddings over a document, as shown in Eq 5.

$$\mathrm{CBoW(D)} = \frac{1}{L}\sum_{i=1}^{L} \mathrm{embedding(w}_i) \tag{5}$$

Figure 10 illustrates input vector $X = \{x_{(1)}, ..., x_{(t)}\}$, hidden state at $h_{(t-1)}$(previous state), current state $h_{(t)}$ and the cell output $y_{(t)}$. As described by the author, DAN is deep unordered model that can obtain near state-of-the-art accuracy on a variety of sentence and document level tasks with just minutes of training time on an average laptop computer.

DAN works in three simple steps (shown in Figure 11)

1. Take the vector average of the embeddings associated with an input sequence of tokens
2. Pass the average through one or hidden layers
3. Perform (linear) classification on the fully connect layer.

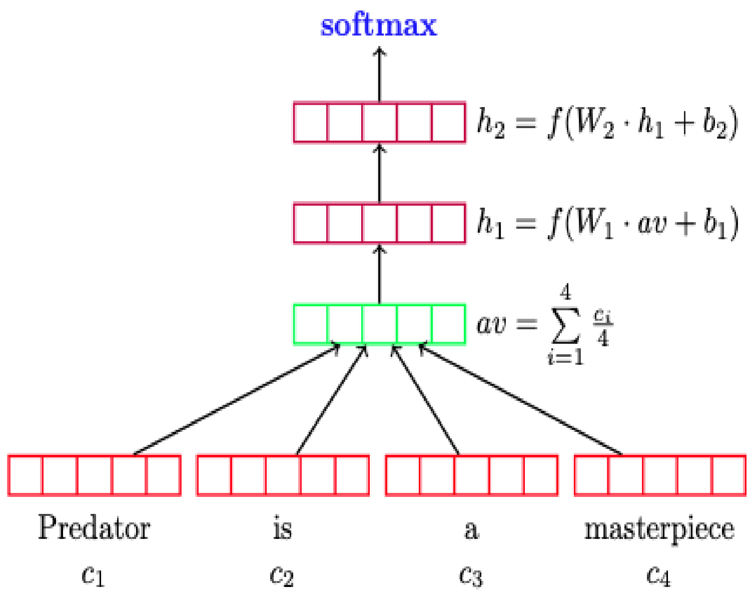

softmax
$h_2 = f(W_2 \cdot h_1 + b_2)$
$h_1 = f(W_1 \cdot av + b_1)$
$av = \sum_{i=1}^{4} \frac{c_i}{4}$
Predator
c_1
is
c_2
a
c_3
masterpiece
c_4

Figure 11. DAN model overview (Iyyer et al. 2015)

Pytorch model training steps

1. The model is trained with 100 epochs.

2. After corpus is encoded as integers from which embedding layer was created.

3. On top of embedding layer fully connected linear layer was added. Its input size is same as the output from embedded layer. The output size is the size as number of classes, in our case its 5.

4. Torchtext is used, it is a Pytorch (Paszke et al. 2019) helper function to prepare training and validation data set.

5. BucketIterator, another Pytorch helper function used to group sentences in similar length size, this function take care of creating training and validation data to small batches, batch size 16 is used when preparing data. Then model and batches are placed on to GPU, as Pytorch supports GPU.

6. CrossEntropyLoss is used as loss function and Adam optimizer for tuning the learning rate during training.

7. Model start training by initializing it which also initialize the dropout layer.

8. Training the model is started by iterating through the batches for each batch, score and gradient computed with respect to the loss, parameters of the model updated by initializing backward propagation at each step.

9. At each training epoch, model loss and accuracy on the validation set was computed.

3.2.5 LSTM

The next choice of neural network model is Long Short Term Memory (LSTM), as shown in Figure 12. LSTM works well to capture the long term dependencies. It introduces a memory cell to remember values over arbitrary time. Input, output and forget gate regulate well, the flow of information in and out of the cell. Bi-directional LSTM (bi-LSTM) is used in this study, in bi-LSTM forward and backward hidden layers are concatenate and apply drop out.

Parameter used in bi-LSTM

Input dimension: size of vocabulary of corpus
Embedding dimension: size of embedding dimension is 100
Hidden dimension: size of embedding dimension is 256
Output dimension: Output dimension is same as number of classes, which is 5.
Number of layers: 2

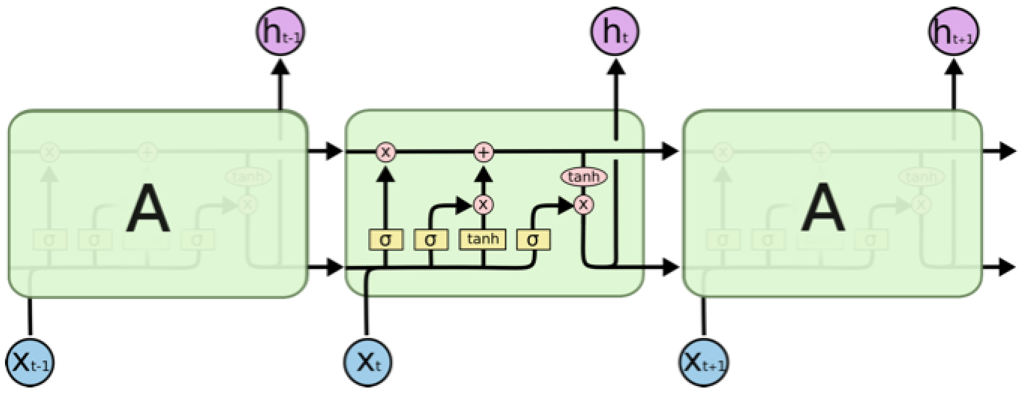


Figure 12. LSTM architecture (Olah 2015)

3.2.6 BERT

This thesis will be using DistilBERT by Sanh et al. (2020), which is a lighter, smaller and faster version of BERT (Devlin et al. 2019), as shown in Figure 13. DistilBERT is pre-trained model used in this work and is provided by *HuggingFace*, which is a library of Transformers (Wolf et al. 2019)

DistlBERT model is used by interface class DistilBertForSequenceClassification (Wolf et al. 2020) by HuggingFace, which provide model for text classification. DistilBertTokenizer is used for tokanization.
Training the DistilBERT model is similar as LSTM (Section 3.2.5) and RNN (Section 3.2.4) as it can be access via Pytorch API.

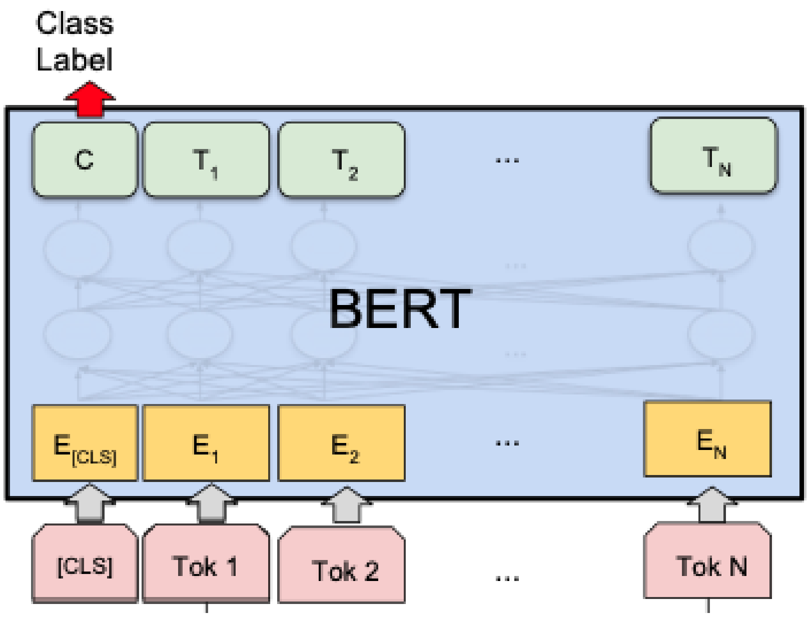


Figure 13. BERT overview of text classification (Devlin et al. 2019)

3.2.7 Limitations

The methods proposed in this study are limited to the Information Technology (IT) project domains namely, mobile application development, web application development, software development, etc. It is assumed that Agile-Scrum methodology is used in managing those stated projects. Since, story-point is the empirical estimation method in Agile-Scrum, it is mandatory that story-point reflects the duration of user-story and scrum team must have followed scrum principles (Yodiz 2019) to estimate the effort estimate of user-stories. The production level implementation is out of scope of this thesis.

3.2.8 Research Design

The libraries and other systems that are used in this thesis listed in Table 6. The data analysis was performed on Google Colab (Google Colab notebook), as it provides free GPU support for Jupyter notebook environment in the cloud. The model training and validation is done using CSC (CSC – IT Center for Science).

Table 6. List of libraries and tools used

Name	Purpose	Version
fastText	library for word representations and sentence classification	0.9.2
HuggingFace	Open source Transformers library	3.5.0
matplotlib (2019)	plot graphs	3.0.3
numpy (2019)	mathematical computing	1.16.2
pandas (2019)	data analytics	0.24.2
PyTorch	Deep learning research platform	1.7.0
scikit-learn (2019)	machine learning library	0.20.3
Google Colab	free GPU supported Jupyter notebook environment	n/a
www.csc.fi	Distributed computing	n/a

4 RESULTS

This chapter describe and analyse the results of this study and also compare them with each other.

In order to validate the results, the data is visualized by using confusion matrix, precision and accuracy from the models.

4.1 Result of TF-idf

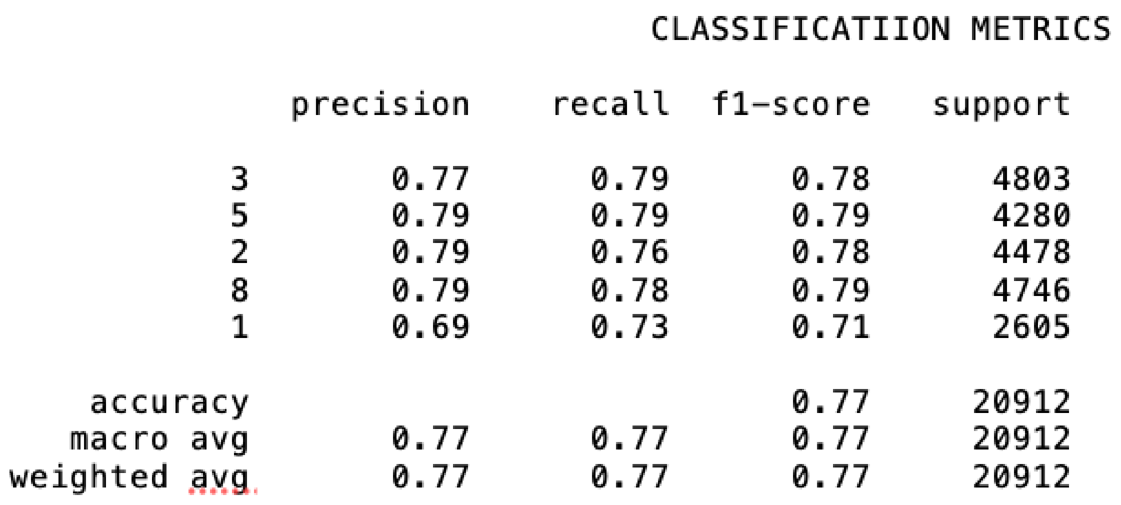
This Section describes the result of TF-idf from two different models, the method is described in Section 3.2.1. The results of Support Vector Machine (SVM) (Joachims 1998) and Logistic Regression (LR) (Menard 2002) will be analyzed and compared. Table 7 shows the train and test accuracy of LR and SVM models.

Analysing the precision and recall (shown in Figure 14), all classes have very stable and equal threshold of accuracy except the class "1", it could be the reason that data for class 1 is in-balance then others or the class threshold boundary from class "1" or other classes are not clearly classify be SVM. For all classes precision and recall values very consistent, class "2", "5" and "8" have precision of 79% while class precision is slightly less. The average accuracy of the SVM model is 77%.

As described by Joachims (1998), SVM is very well suited for text categorization. SVM consistently achieve good performance on text categorization task. Furthermore, SVM do not require any parameter tuning, since it can find good parameter setting automatically, all this makes SVM a very promising and easy to use method for text classifier. LR on other hand doesn't perform so well comparing to the SVM. Confusion matrix (normalized and non-normalized) of SVM validation accuracy show in Figure 15 & 16

Table 7. TF-idf results(SVM, LR)

Model Name(TF-idf+3-Gram)	Train Accuracy	Test Accuracy
Support Vector Machine(SVM)	96.8%	77.4%
Logistic Regression	83.8%	72.1%



```
                              CLASSIFICATIIION METRICS

              precision    recall  f1-score   support

           3       0.77      0.79      0.78      4803
           5       0.79      0.79      0.79      4280
           2       0.79      0.76      0.78      4478
           8       0.79      0.78      0.79      4746
           1       0.69      0.73      0.71      2605

    accuracy                           0.77     20912
   macro avg       0.77      0.77      0.77     20912
weighted avg       0.77      0.77      0.77     20912
```

Figure 14. FI-Score of SVM model with TF-idf

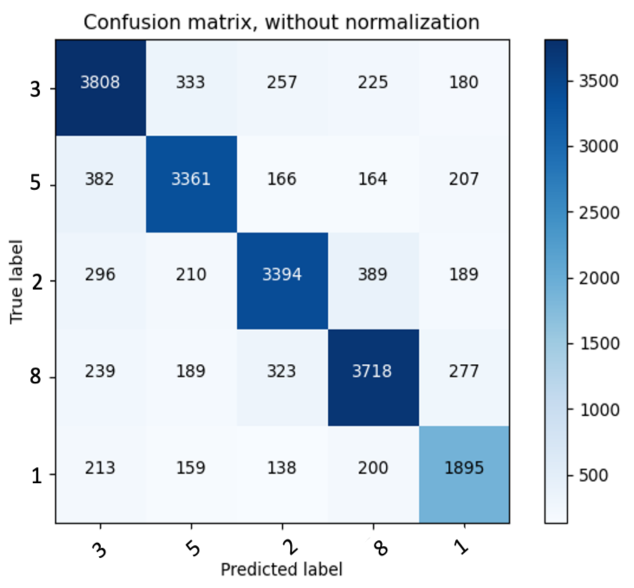


Figure 15. SVM model validation accuracy confusion matrix

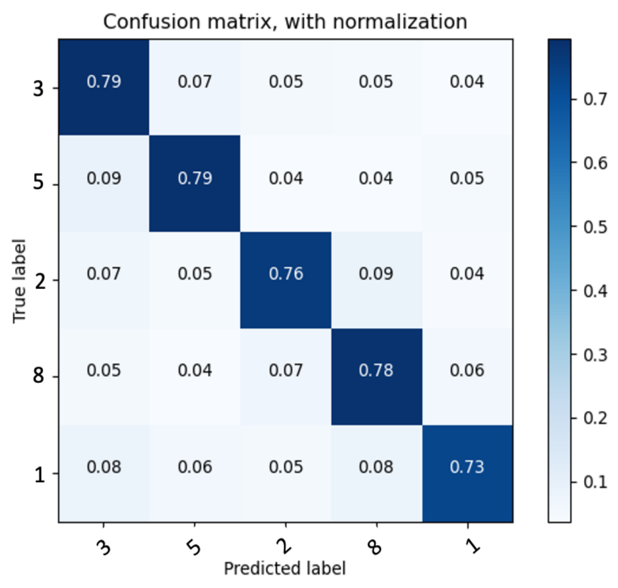


Figure 16. SVM model validation accuracy confusion matrix

4.2 Result of fastText

This Section describes the result of fastText with two different approaches, these approaches are described in Section 3.2.2. The results of two approaches i) fastText approach with parameter selection and ii) fastText with pre-trained model will be analyzed and compared.

Approach i) fastText Result with hyper-parameter values

Dataset was split as mentioned in Section 3.1.2. In first approach the model was trained with different parameter values to identify the best values on which model performed well. Table 8 shows the parameter values and best performing values. The training and test accuracy (shown in Table 9) with this method was 96.54% and 85.92 %.

Table 8. fastText best performing parameters

Parameters name	Values Applied	Best Performing Values
lr (learning rate)	(0.05, 0.1, 0.2)	0.2
N-gram (word)	(1, 2, 3, 5)	3
epoc (number of epochs)	(5, 10, 15)	5
ws (window size)	(5, 10 , 15)	10
loss	(ns, hs, softmax)	softmax

The Figure 17 shows the confusion matrix of fastText test accuracy.

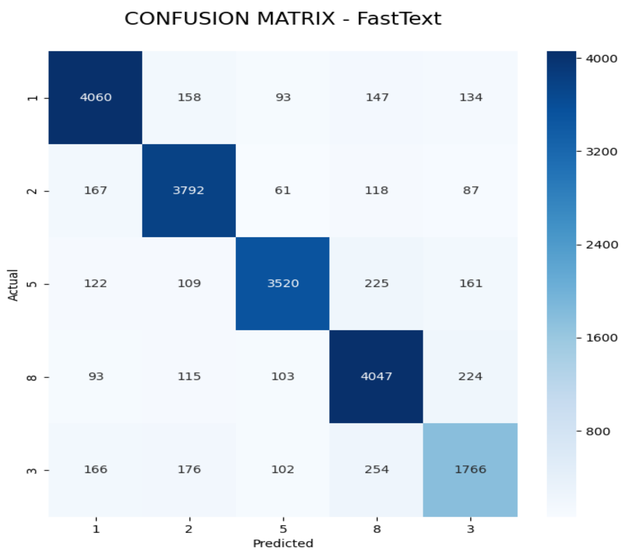


Figure 17. fastText test accuracy - parameter approach

Table 9. fastText training and test accuracy - parameter approach

Model name	Train accuracy	Test accuracy
fastText	96.54%	85.92%

Approach ii) Result from fastText pre-trained model approach.

fastText model was using 5-cross-validation for training the data. The data is divided in five folds and on each fold mean accuracy was calculated on validation set. Figure 18 shows the validation accuracy of all 5 folds. Table 10 shows the 98.32% train and 87.4% test mean accuracy of this approach.

pre-trained model was used in this approach. It help the accuracy, as model learn faster and better. This thesis is applying similar technique proposed by Joulin et al. (2016) and the result obtained in this thesis is aligned with the results from Joulin et al. (2016). The author quoted "fastText is often on par with deep learning classifiers in terms of accuracy, and many orders of magnitude faster for training and evaluation". Also author mentioned that, they were able to achieve approximate 1% higher accuracy in their study when using pre-trained model. In this thesis, using the pre-trained model approach, the accuracy is increased by 1.48% higher, comparing to the model which does not use pre-trained model (hyper-parameter approach i). Which shows that the result achieved is similar to the fore mentioned study.

Table 10. fastText pre-trained model- accuracy result

Model name	Train mean accuracy	Test mean accuracy
fastText	98.32%	87.4%

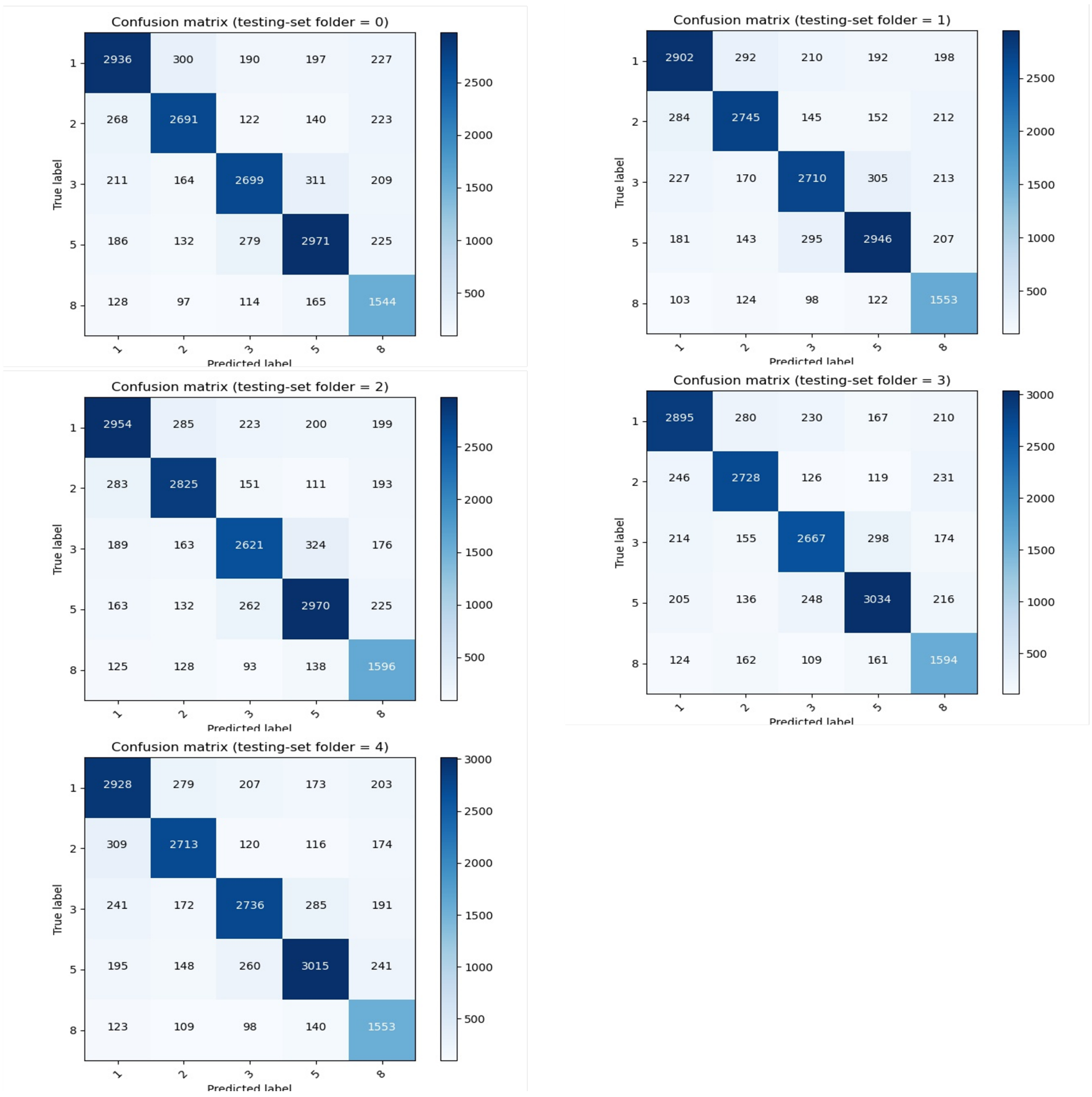


Figure 18. fastText (CV-5) Test accuracy- with pre-trained model

4.3 Result of RNN

This Section describes the result of RNN, the method is described in Section 3.2.4. The technique followed is similar as proposed by Iyyer et al. (2015). Its a DAN network, which is a special variant of RNN, 100 epochs is used to train the network, training accuracy, validation accuracy and loss is calculated after each 5 epoch and mean accuracy is calculated after all the epochs. Training mean accuracy 84.18% and validation mean accuracy 69.75% shows in Table 11.

Training the network was fast, but it doesn't produce as good result as fastText or TF-idf approach. After 50 epochs the validation loss doesn't decrease much. Training loss shows a good decrease, but not significantly in validation. Figure 19 shows the training and validation accuracy, Figure 20 show training and validation loss.

Table 11. RNN training and validation accuracy results

Model name	Training accuracy mean	Validation accuracy mean
DAN	84.178%	69.75%

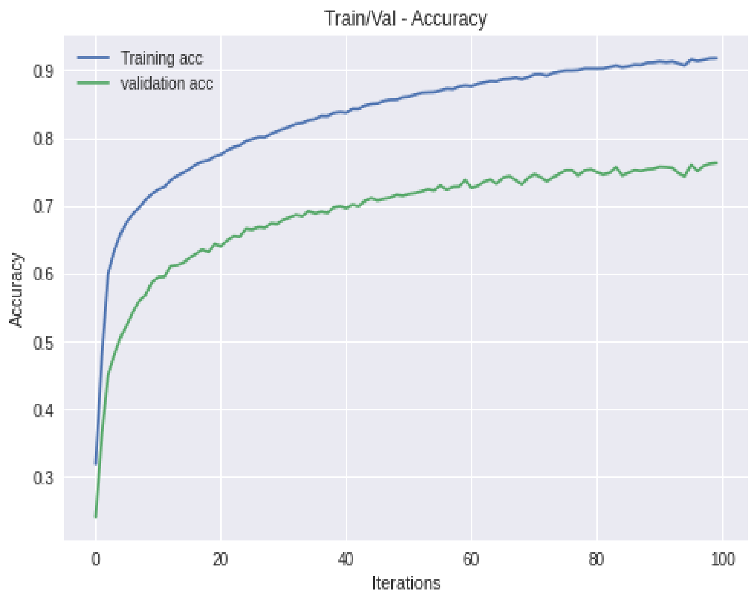


Figure 19. RNN validation and training accuracy of 100 epochs

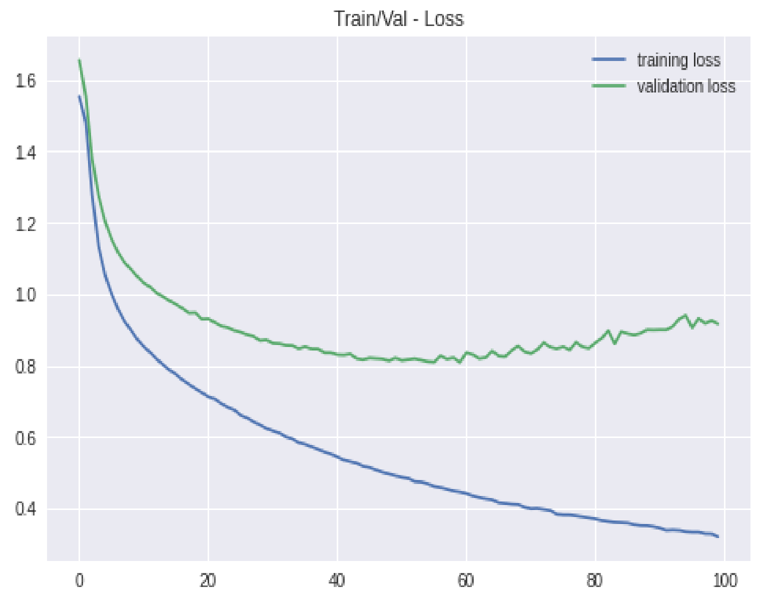


Figure 20. RNN validation and training loss of 100 epochs

4.4 Result of LSTM

This Section describes the result of LSTM, as shown in Table 12, the method is described in Sections 3.2.5 & 2.2. The technique followed in this thesis is similar to proposed by Choetkiertikul et al. (2019), but not exactly the same, the main difference is, in this thesis bi-LSTM is used which is a special variant of LSTM, the method is described in Section 3.2.2. Other difference is, in this thesis is, no additional RHN layer is added on top of bi-LSTM. The concept is similar to extent that LSTM is used for text classification. bi-LSTM perform significantly well with training mean accuracy of 93.1% and validation mean accuracy of 83.2% but still lack behind the fastText pre-trained model (Section 3.2.2) approach. The bi-LSTM is trained using similar method described in RNN (Section 4.3). Training loss shows a good decrease but not significantly in validation set. Figure 21 shows the training and validation accuracy, Figure 22 show training and validation loss.

Table 12. LSTM training and validation accuracy results

Model name	Training mean accuracy	Validation mean accuracy
bi-LSTM	93.1%	83.2%

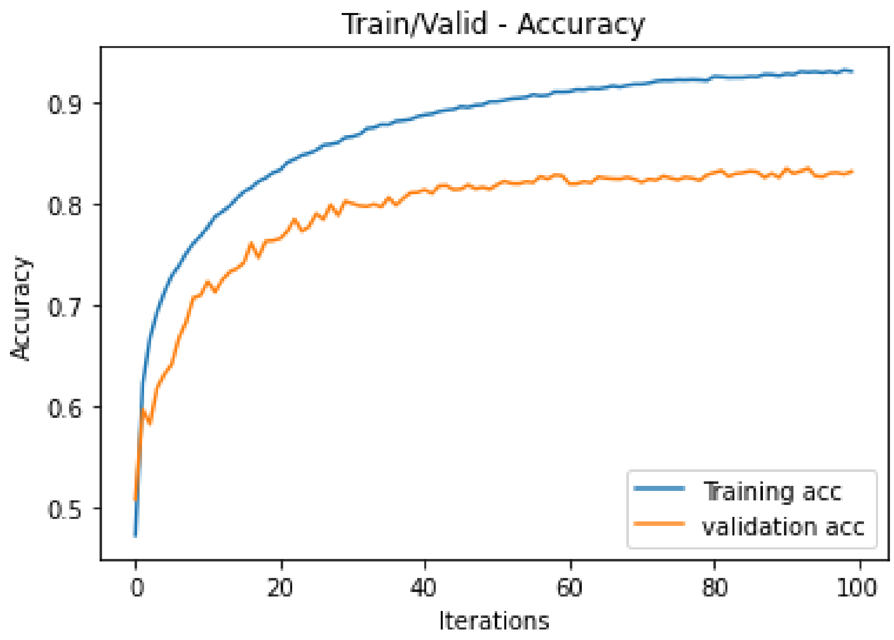


Figure 21. bi-LSTM training and validation accuracy of 100 epochs

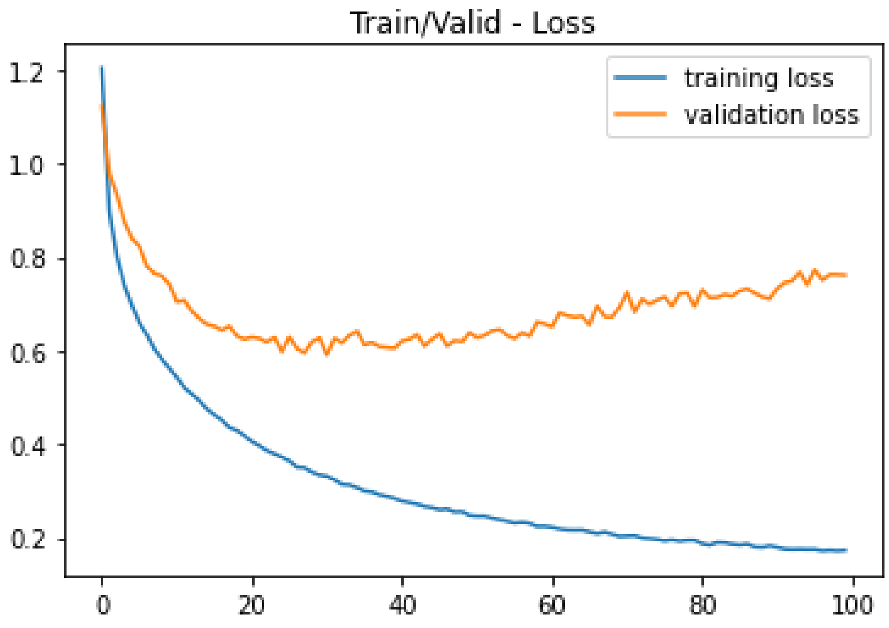


Figure 22. bi-LSTM training and validation loss of 100 epochs

4.5 Result of BERT

This Section describes the result of BERT, the method is described in Section 3.2.6. The technique followed in this thesis is similar to proposed by (Sanh et al. 2020, Wolf et al. 2019). BERT model has been creating tremendous popularity. But it doesn't produce good result in this study. Table 13 shows the validation accuracy results. In fact it is one of the least performing model in our methods. distilBERT perform worst than TF-idf approach. There could be two explanation, lighter variant of BERT which this study is using. Another explanation is that in NLP problem relating to text classification it depend on domain of the dataset in investigation. BERT has proven excellent result for different kind of NLP problem, like generic email spam detection, sentiment analysis in public domain, etc. The study showed that distilBERT is not suitable for the dataset used in this study. The validation accuracy of distilBert model was 70.7%

Training loss shows a good decrease but not as significantly in validation. Figure 23 shows the training and validation accuracy, Figure 24 shows training and validation loss.

Table 13. BERT validation accuracy result

Model name	**Validation accuracy mean**
BERT	70.7%

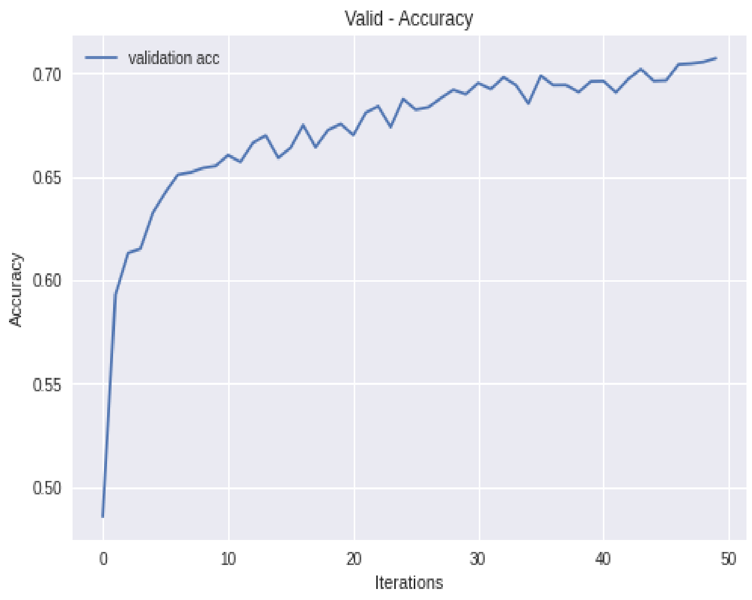


Figure 23. DistilBERT validation accuracy of 100 epochs

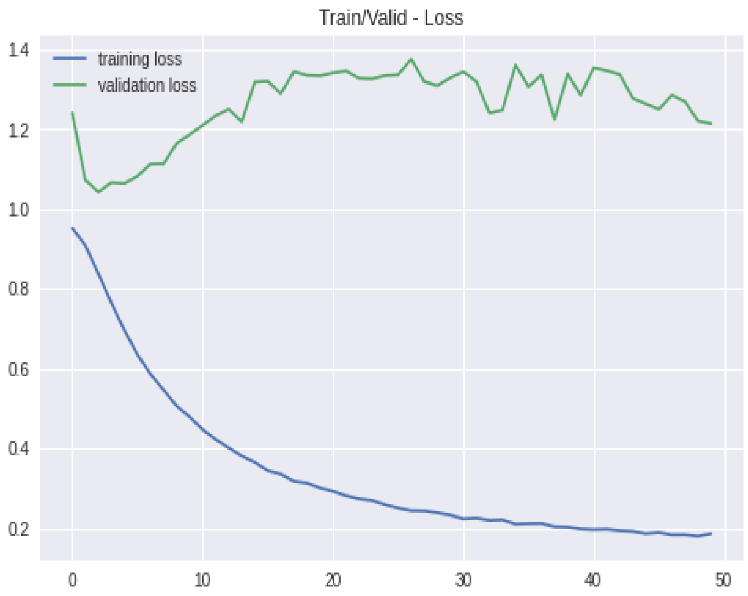


Figure 24. DistilBERT training and validation loss

4.6 Result Comparison

Here are the results (Table 14) from all the techniques used in this thesis: TF-idf, fastText, RNN, bi-LSTM and BERT.

Table 14. Consolidated results view of all the models

Techniques	Validation Acc. %
fastText with pre-trained model	**87.4%**
fastText without pre-trained model	85.92%
bi-LSTM+word embedding	83.2%
TF-idf+SVM+3-gram	77.4%
TF-idf+LR+3-gram	72.1%
DistilBERT	70.7%
DAN+word embedding	69.7%

Finally, when looking at the results of all the methods, its evident that the fastText (pre-trained model approach) performs best with 87.4% accuracy, followed by second approach of fastText with accuracy of 85.92%. So we can clearly say that both approaches of fastText performed best in our study and is the state-of-the-art. Furthermore, in neural network domain, bi-LSTM was the top performing model with 83.2% accuracy. 4th in our list is the TF-idf+SVM model with the accuracy of 77.4% followed by TF-idf+LR model with accuracy of 72.1%. The two models which were at the bottom of list are distilBERT with accuracy of 70.7% and list one DAN with accuracy of 69.7%

5 DISCUSSION

In this thesis, we try to solve the problem faced by Agile project teams to correctly predict effort estimate of user-stories. The aim of this thesis was to provide a machine learning model which can predict the effort estimate to facilitate the team in planning phase.

To solve this multi-classification problem, we tried different approaches, our ambition was to test the classical approach (TF-idf) and compare it against more recent and advance approaches available today like fastText, RNN, bi-LSTM and BERT.

Our first approach was TF-idf, we use SVM and LR. SVM is simple model which works well with text classification, it was very easy to implement but slow in performance because of high parameters and sparsity in matrix. But overall performance was very good, it was much better then some of the advance neural network (DAN and BERT). The reason that TF-idf technique work so well in text classification is because of n-gram approach which is empowered by SVM model which provide the best combinations from all different domains. TF-idf is good in providing term scoring of words in corpus, and using n-gram (n-item long arrays of words of skip-gram) approach, increase the chances to identify the words belong to a certain class, because presence of certain words can strongly indicate the class of a user-story. The ideal value of n-gram in this kind of approach is 2-3 n-gram, the higher value then 3-gram leads to very sparse vector, because the occurrence of word increase due to higher n-gram value, this leads to lower score of potential word due to the characteristic of TF-idf, as higher occurrence of words penalized in TF-idf, hence decreases the performance.

Our second approach was fastText, it is relatively new approach and gain much popularity due to this performance and accuracy. Its performance was among the best in all the model analyzed. This model make it to the most suitable choice as it offer performance, speed and is easy to implement. The only drawback was it doesn't support GPU yet.

Our third choice was DAN, which is a simple variant of RNN which make it suitable for text classification. DAN is utilizing Bag of Words (BoW) approach as embedding layer, one possible reason for its poor performance could be due to Bag of Words (BoW) technique, as in BoW the sequence of sentence is lost, its focus is on counts of word, its a order-less document representation.

Our fourth method was bi-LSTM, this neural network performs really well, the second best after the fastText. LSTM is good in sequence problem due to its internal gating mechanism that can handle the information well. Our last model to analyse was distil-BERT. It does not perform well as compared to other models, it falls second last in the list when comparing the performance and accuracy of other models.

To compare our results with the studies we used as a benchmark, we can summarize that in TF-idf (Section 3.2.1) approach we were able to achieve better accuracy (18.4% improvement) with SVM then to that of Porru et al. (2016). Similarly our LSTM (Section 3.2.5) result with the approach from Choetkiertikul et al. (2019) showed improved accuracy by almost 24% while comparing it with fore mentioned study.

Moreover, we also identified a novel approach by using fastText (Section 3.2.2) method to predict the duration of user-stories in Agile-Scrum. This state-of-the-art approach produced accuracy of around 87.4% and has not not been reported before in this context.

5.1 Conclusion

To conclude, the fastText prove to be much powerful text classification library, it is much easier to use and provide much better result then other approaches. The classical approach (TF-idf) is faster but still lacking behind in accuracy and speed. On other hand, Neural networks (LSTM) was harder to train as it required longer training time and GPU power, but still offered reasonably good performance when comparing to other approaches.

The fastText with pre-trained model is state-of-the art in our model list by providing the best solution to this problem, its excel is all level of performance, speed and simple to implement.

To summarize, in problems relating to NLP, the most important factor is the dataset itself, depending on the data domain and characteristic of data it is hard to tell which approach works well. As some model or technique may work really well with one type of data, but may not work on data with same type but different domain. In NLP domain, the context of words and semantics of document plays an important role. As morphology of word is different in languages and domain it belong to. Similarly, the approaches used in this thesis may or may not work in other user-stories, if they belong to different domains for e.g health, social science or marketing.

With this study we can conclude that document presentation and word embedding plays an import role in NLP problems, the clear explanation is fastText works due to use of skipgram mechanism and LSTM works over DAN because of handling the information effectively due to it excellent gating mechanism.

5.2 Future work

The future work will be to investigate other models in detail. Also it will be good to further study fastText and try to improve the accuracy even higher. One interesting angle would be to productise this feature and made it available with Yodiz Agile Project Management tool (www.yodiz.com) as an additional feature.

REFERENCES

Andrei, Bogdan-Alexandru; Casu-Pop, Andrei-Cosmin; Gheorghe, Sorin-Catalin & Boiangiu, Costin-Anton. 2019, A STUDY ON USING WATERFALL AND AGILE METHODS IN SOFTWARE PROJECT MANAGEMENT, *Journal of Information Systems & Operations Management*, pp. 125–135.

Beck, Kent; Beedle, Mike; van Bennekum, Arie; Cockburn, Alistair; Cunningham, Ward; Fowler, Martin; Grenning, James; Highsmith, Jim; Hunt, Andrew; Jeffries, Ron; Kern, Jon; Marick, Brian; Martin, Robert C.; Mellor, Steve; Schwaber, Ken; Sutherland, Jeff & Thomas, Dave. 2001, *Manifesto for Agile Software Development*. Available: http://www.agilemanifesto.org/.

Bloch, M.; Blumberg, S. & Laartz, Jürgen. 2012, Delivering large-scale IT projects on time, on budget, and on value, *McKinsey Quarterly*, vol. 27, , pp. 2–7. Available: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value.

Bojanowski, Piotr; Grave, Edouard; Joulin, Armand & Mikolov, Tomas. 2017, *Enriching Word Vectors with Subword Information*.

Choetkiertikul, M. 2017, *Datasets*. Available: https://github.com/morakotch/datasets.

Choetkiertikul, M.; Dam, H. K.; Tran, T.; Pham, T.; Ghose, A. & Menzies, T. 2019, A Deep Learning Model for Estimating Story Points, *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 637–656.

CSC – IT Center for Science. Available: https://docs.csc.fi/computing/overview/.

Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton & Toutanova, Kristina. 2019, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.

Gartner. 2019, *Results Summary: Agile in the Enterprise*. Available: https://circle.gartner.com/Portals/2/Resources/pdf/Agile%20in%20the%20Enterprise%202019%20-%20Results%20Summary%20(updated).pdf.

González-Carvajal, Santiago & Garrido-Merchán, Eduardo C. 2020, *Comparing BERT against traditional machine learning text classification*.

Goodfellow, Ian; Bengio, Yoshua & Courville, Aaron. 2016, *Deep Learning*, MIT Press, http://www.deeplearningbook.org.

Google Colab notebook. Available: https://colab.research.google.com.

Hochreiter, Sepp & Schmidhuber, Jürgen. 1997, Long Short-Term Memory, *Neural Comput.*, vol. 9, no. 8, p. 1735–1780. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

Iyyer, Mohit; Manjunatha, Varun; Boyd-Graber, Jordan & Daumé III, Hal. 2015, Deep Unordered Composition Rivals Syntactic Methods for Text Classification, In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China: Association for Computational Linguistics, pp. 1681–1691. Available: https://www.aclweb.org/anthology/P15-1162.

Joachims, Thorsten. 1998, Text categorization with Support Vector Machines: Learning with many relevant features, In: Claire Nédellec & Céline Rouveirol, eds., *Machine Learning: ECML-98*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 137–142.

Joulin, Armand; Grave, Edouard; Bojanowski, Piotr & Mikolov, Tomas. 2016, *Bag of Tricks for Efficient Text Classification*.

Kim, Sung. 2017, *PyTorchZeroToAll*. Available: https://github.com/hunkim/PyTorchZeroToAll.

McKinsey & Company. 2017, *RD that's on time and on budget? Yes, with predictive analytics*. Available: https://www.mckinsey.com/business-functions/operations/our-insights/rd-thats-on-time-and-on-budget-yes-with-predictive-analytics.

Menard, Scott. 2002, *Applied logistic regression analysis*, vol. 106, Sage.

Mikolov, Tomas; Chen, Kai; Corrado, Greg & Dean, Jeffrey. 2013, *Efficient Estimation of Word Representations in Vector Space*.

Olah, C. 2015, *Understanding LSTM Networks*. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Paszke, Adam; Gross, Sam; Massa, Francisco; Lerer, Adam; Bradbury, James; Chanan, Gregory; Killeen, Trevor; Lin, Zeming; Gimelshein, Natalia; Antiga, Luca; Desmaison, Alban; Kopf, Andreas; Yang, Edward; DeVito, Zachary; Raison, Martin; Tejani, Alykhan; Chilamkurthy, Sasank; Steiner, Benoit; Fang, Lu; Bai, Junjie & Chintala, Soumith. 2019, PyTorch: An Imperative Style, High-Performance Deep Learning Library, In: H. Wallach; H. Larochelle; A. Beygelzimer; F. d'Alché-Buc; E. Fox & R. Garnett, eds., *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., pp. 8024–8035. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M. & Duchesnay, E. 2011, Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, vol. 12, , pp. 2825–2830.

Pham, Trang; Tran, Truyen; Phung, Dinh & Venkatesh, Svetha. 2016, *Faster Training of Very Deep Networks Via p-Norm Gates*.

Porru, Simone; Murgia, Alessandro; Demeyer, Serge; Marchesi, Michele & Tonelli, Roberto. 2016, Estimating Story Points from Issue Reports, In: *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering, PROMISE 2016, Ciudad Real, Spain, September 9, 2016*, ACM, pp. 2:1–2:10. Available: https://doi.org/10.1145/2972958.2972959.

Ramin, Frederike; Matthies, Christoph & Teusner, Ralf. 2020, More than code: Contributions in scrum software engineering teams, In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pp. 137–140.

RandulaKoralage. 2020, *AgileScrumSprintVelocityDataSet*. Available: https://github.com/RandulaKoralage/AgileScrumSprintVelocityDataSet.

Sanh, Victor; Debut, Lysandre; Chaumond, Julien & Wolf, Thomas. 2020, *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*.

Scott, Ezequiel & Pfahl, Dietmar. 2018, Using Developers' Features to Estimate Story Points, In: *Proceedings of the 2018 International Conference on Software and System Process*, ICSSP '18, New York, NY, USA: Association for Computing Machinery, p. 106–110. Available: https://doi.org/10.1145/3202710.3203160.

ScrumAlliance. 2018, *Scrum Alliance Scaling and Agile Transformation*. Available: http://info.scrumalliance.org/State-of-Scrum-2017-18.

Srivastava, Rupesh Kumar; Greff, Klaus & Schmidhuber, Jürgen. 2015, *Training Very Deep Networks*.

Tang, Duyu; Qin, Bing & Liu, Ting. 2015, Document Modeling with Gated Recurrent Neural Network for Sentiment Classification, In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal: Association for Computational Linguistics, pp. 1422–1432. Available: https://www.aclweb.org/anthology/D15-1167.

Vargas, Diego Armando Diaz; Xue, Rui; Baron, Claude; Esteban, Philippe; Vingerhoeds, Rob; Citlalih, Y & Liu, Chao. 2018, Implementing SCRUM to develop a connected robot, *arXiv preprint arXiv:1807.01662*.

Wikipedia. 2020, *Agile Scrum*. Available: https://en.wikipedia.org/wiki/Scrum_(software_development).

Wolf, Thomas; Debut, Lysandre; Sanh, Victor; Chaumond, Julien; Delangue, Clement; Moi, Anthony; Cistac, Pierric; Rault, Tim; Louf, Rémi; Funtowicz, Morgan; Davison, Joe; Shleifer, Sam; von Platen, Patrick; Ma, Clara; Jernite, Yacine; Plu, Julien; Xu, Canwen; Scao, Teven Le; Gugger, Sylvain; Drame, Mariama; Lhoest, Quentin & Rush, Alexander M. 2019, HuggingFace's Transformers: State-of-the-art Natural Language Processing, *ArXiv*, vol. abs/1910.03771, .

Wolf, Thomas; Debut, Lysandre; Sanh, Victor; Chaumond, Julien; Delangue, Clement; Moi, Anthony; Cistac, Pierric; Rault, Tim; Louf, Rémi; Funtowicz, Morgan; Davison, Joe; Shleifer, Sam; von Platen, Patrick; Ma, Clara; Jernite, Yacine; Plu, Julien; Xu, Canwen; Scao, Teven Le; Gugger, Sylvain; Drame, Mariama; Lhoest, Quentin & Rush, Alexander M. 2020, Transformers: State-of-the-Art Natural Language Processing, In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, pp. 38–45. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Yang, Zichao; Yang, Diyi; Dyer, Chris; He, Xiaodong; Smola, Alex & Hovy, Eduard. 2016, Hierarchical Attention Networks for Document Classification, In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California: Association for Computational Linguistics, pp. 1480–1489. Available: https://www.aclweb.org/anthology/N16-1174.

Yodiz. 2018, *Effective agile sprint planning*. Available: https://www.yodiz.com/blog/writing-user-stories-examples-and-templates-in-agile-methodologies/.

Yodiz. 2019, *Effective agile sprint planning*. Available: https://www.yodiz.com/blog/effective-agile-sprint-planning/.

Zhang, Xiang; Zhao, Junbo & LeCun, Yann. 2016, *Character-level Convolutional Networks for Text Classification*.