TRAIGE

AI-enabled Triaging System
PROJECT REORT

Tng Sian Soo (Shawn)
Kumaravelu Varadharajan (Varad)

6 May 2020

—

Practice Module – Intelligent Reasoning Systems

—

Gu Zhan

Table of Contents

1. INTRODUCTION 3
2. PROJECT OBJECTIVES 4
3. KNOWLEDGE MODELLING 4
   - 3.1 Knowledge Representation - Intent Detection 4
   - 3.2 Knowledge representation – Production Rules 5
4. SOLUTION 7
   - 4.1 System Architecture 7
   - 4.2 System Scope 7
   - 4.3 System Scope 7
5. FUTURE IMPROVEMENTS 11
6. CONCLUSION 12
7. APPENDICES 13
   - Appendix A 13
   - Appendix B 13
   - Appendix C 14
   - Appendix D 14
   - Appendix E 16

1. INTRODUCTION

Customer service across all industries, struggles with the high volume of incoming tickets at one point or other. For organizations without the ability to manage spikes in response demand, the service level and satisfaction of customers goes down rapidly and results in unnecessary escalations which could have been avoided efficiently.

While computer systems have evolved to offer an elastic mode in processing power and storage space, the human workforce is fixed. This is due to the cost involved in hiring and the turn-around time to get someone onboard and train (typically 8-12 weeks). These customer service agents need to deal with tickets which fall outside the scope of self-help FAQs and the popular chatbots.

Traditionally, this process is reactive. There are a lot of administrative steps in that process, causing layers of inefficiencies. Urgent cases get piled up behind in the queue since the end user will perceive even a simple problem as a major issue.

The support queues require a triaging process like an accident & emergency center in a hospital. All new incoming patients are screened using standard practices and a decision is made by the triage nurse on the criticality so that they can be placed in the queue appropriately.

The proposed solution is to build an intelligent triage process which can take in some rules input by the relevant experienced customer service managers to classify the severity of the incoming tickets before assigning to the customer service agent. The system will read through the content of the ticket and flag them with various criticalities as defined by the customer service managers.

2. PROJECT OBJECTIVES

With limited staff resources, the support organization are likely to suffer from poor customer satisfaction since there is always a pile of issues that needs to be addressed and insufficient number of staff available to deal with them. A high priority issue comes up and gets stuck behind other tickets until an escalation from the customer happens. Normal tickets also become critical as time passes.

The support organization needs a robust triaging system to prioritize cases based on their current severity and predict their immediate future severity. The system should be pluggable into existing ticketing systems and be flexible to allow manual overrides.

The project's objective is to help the customer service managers better prioritize based by developing the following cognitive capabilities:

a) Understand sentiments of the ticket contents to infer some level of severity
b) Matching mentions of pre-defined words to organization's known words to each level of severity
c) Supervised learning of historical data to predict level of severity

3. KNOWLEDGE MODELLING

3.1 Knowledge Representation - Intent Detection

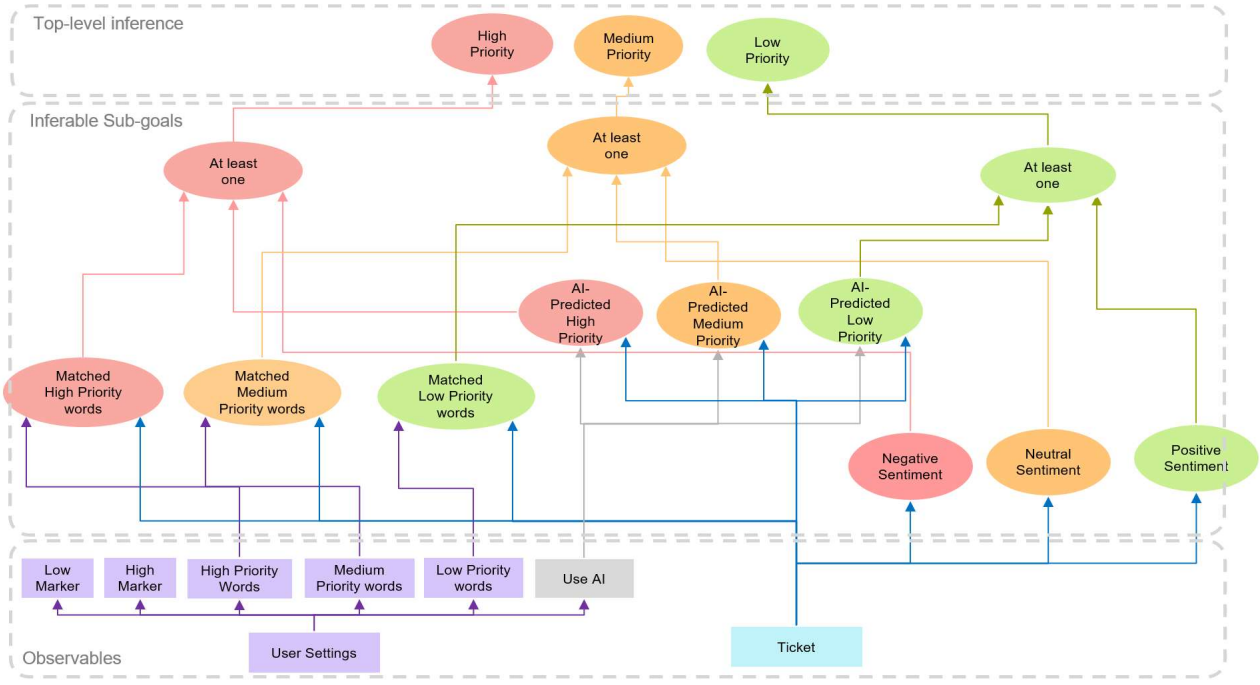
The system implemented intent detection through supervised machine learning models. In this case, the intents are known as the severity levels, or priority levels.

Level of Severity	Numeric Representation
High Priority	2
Medium Priority	1
Low Priority	0

The severities are classified into 3 classes of intent as too many categories will create more cognitive overload rather aiding the human. This numeric representation also helped to standardizes the required labelling for supervised learning of historical data

3.2 Knowledge representation – Production Rules

The production rules are represented by the following dependency graph.



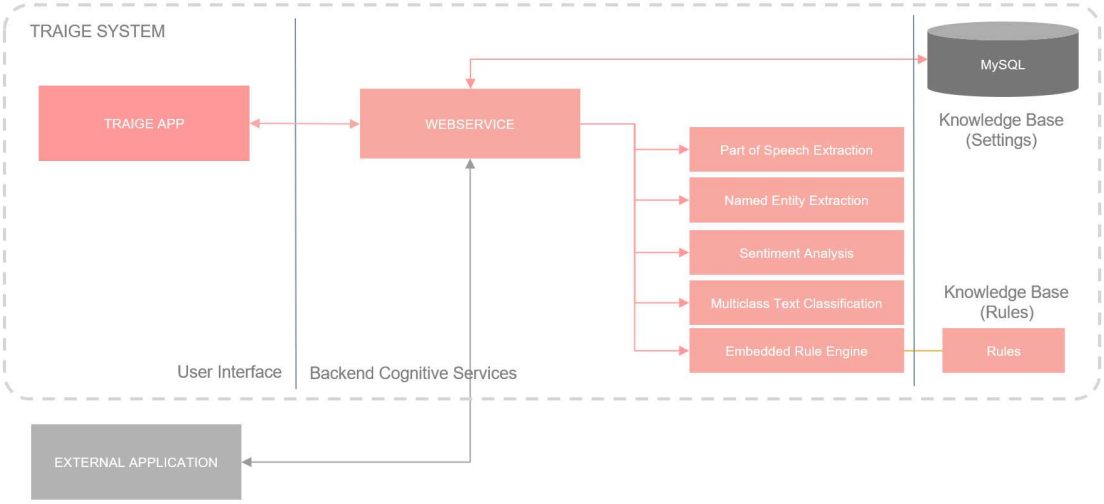
The dependency diagram arranges the factors affecting inference of the severity in a hierarchical tree structure. The top most level node represents the decision of the proposed system, which in this case, recommends a severity level to the user. This decision can be broken down into multiple layers of inferable sub-goals or subfactors before arriving at a list of “observables”. These “observables” are gathered from user settings and ticket contents of the proposed system.

The business rules are represented as below and the syntax of the rules can be found in Appendix E:

LHS	RHS
IF HighMarker-LowMarker1 < 2	THEN InvalidMarkers
IF any words extracted matched High Priority words	THEN HighPriorityWordsMatched
IF any words extracted matched Medium Priority words	THEN MediumPriorityWordsMatched
IF any words extracted matched Low Priority words	THEN LowPriorityWordsMatched
IF predicted priority is 0	THEN LowPriorityPredicted
IF predicted priority is 1	THEN MediumPriorityPredicted
IF predicted priority is 2	THEN HighPriorityPredicted
IF sentiments >= HighMarker) AND not InvalidMarkers	THEN GoodSentiments
IF sentiments > LowMark and sentiments < HighMarker) AND not InvalidMarkers	THEN NeutralSentiments
IF sentiments <= LowMarker) AND NOT InvalidMarkers	THEN BadSentiments
IF (GoodSentiments OR LowPriorityPredicted OR LowPriorityWordsMatched) AND NOT HighPriority AND NOT MediumPriority	THEN LowPriority
IF (NeutralSentiments OR MediumPriorityPredicted OR MediumPriorityWordsMatched) AND NOT HighPriority	THEN MediumPriority
IF BadSentiments OR HighPriorityPredicted OR HighPriorityWordsMatched	THEN HighPriority

4. SOLUTION

4.1 System Architecture



The architecture comprises of a user interface (frontend) and webservices (backend) and database. The webservices are supported by multiple language understanding capabilities, machine learning techniques and a rule engine. The domain expert's knowledge are captures as settings and rules in the knowledge base.

4.2 System Scope

The solution aims to provide an API interface to customer's existing ticketing system. Customer may configure their preferred settings in the system and use it to automate the prioritization of their customer service work.

4.3 System Scope

There are 4 key features powered by 3 rulesets. If any of the 3 rulesets are activated, the highest priority applies.

a) Sentiment Analysis (Rule Set)

This feature models the way a human understands and interprets the emotions or mood of the customer. More attention should be given to customer who is perceived to be unhappy. Natural language understanding was applied to categorize the sentiments into 5 classes. The sentiment analysis engine adopted Stanford CoreNLP[1] software and uses 5 classes of sentiments as follows:

Sentiment	Numeric Representation
Very Positive	4
Positive	3
Neutral	2
Negative	1
Very Negative	0

To re-map the sentiments to 3 classes, users can configure a Low Marker and High Marker based on their preference. For example, assume Low Marker=1 and High Marker=3. If the sentiments analysis outputs a 0 or 1, the ticket will be considered of bad sentiments. If the sentiments analysis outputs a 3 or 4, the ticket will be considered of good sentiments. To illustrate, a statement "I am very unhappy with your service level." yield a sentiment of 1. It will activate the sub goal of bad sentiments. This feature contributes to the sub goal of the entire inference system.

b) Keyword Matching (Rule Set)

This feature models the way a human understands and interprets the concepts by picking up entities and events. Part-of-Speech (POS) Extraction and Named Entity Recognition (NER)[1] are applied to extract relevant words from the ticket content. The human mind tends to

[1] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60. [pdf] [bib]

correlate certain words with specific level of priority, hence here, we automate the process by allowing the IT helpdesk staff to predefine the words associated with each level of priority to form part of the knowledge base.

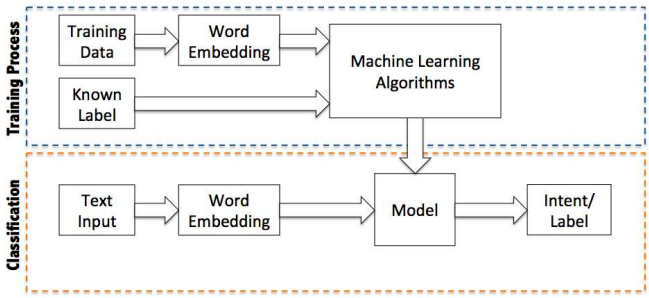
For example:

Category	Associated Words
High Priority	OperationalSystemName, ignore, disappointment
Medium Priority	ABC System, delay
Low Priority	assist, SimpleSystem,

Any mention of “OperationalSystemName” will activate the ruleset and infer the high-priority sub goal.

c) Supervised Machine Learning (Rule Set)

This is the 3rd ruleset to enable customers to try out supervised machine learning without the need to know technically how it works. The customer can make use of their historical data to generate model that is relevant to his organization. Essentially, this is an implementation of intent detection through supervised machine learning models



The customer shall perform the following to begin training:

I) Prepare data by extracting the textual tickets contents and place it under 'sentence' column.

II) For each sentence, label the data based on 3 levels of severity (0,1,2) under 'priority' column, with 2 being the most severe.

III) Ensure data is in comma-delimited format (shown below) and in a CSV file and upload into the system.

```
sentence,priority
"With or without the sex , a wonderful tale of love and destiny , told well by a master storyteller",0
"Cattaneo should have followed the runaway success of his first film , The Full Monty , with something different .",0
"It 's best to avoid imprisonment with the dull , nerdy folks that inhabit Cherish .",0
An ambitious and moving but bleak film .,0
A model of what films like this should be like .,0
And I expect much more from a talent as outstanding as director Bruce McCulloch .,0
"Although Jackson is doubtless reserving the darkest hours for The Return of the King , we long for a greater sense of urgency in the here and now of The
Two Towers .",0
It offers little beyond the momentary joys of pretty and weightless intellectual entertainment .,0
"As a movie , it never seems fresh and vital .",0
"It 's so crammed with scenes and vistas and pretty moments that it 's left a few crucial things out , like character development and coherence .",0
Vividly conveys both the pitfalls and the pleasures of over-the-top love .,0
"Occasionally funny and consistently odd , and it works reasonably well as a star vehicle for Zhao .",0
Consider the film a celluloid litmus test for the intellectual and emotional pedigree of your date and a giant step backward for a director I admire .,0
"Yes , 4Ever is harmless in the extreme and it 'll mute your kids for nearly 80 minutes , but why not just treat the little yard apes to the real deal and
take them to Spirited Away ?",0
```

The priority column will be the target variable while the sentence will be the dependent variables. The sentence is first transformed into word vectors(embedding) representation so that the machine can understand and train the models. In other words, features were extracted from ticket textual content using the bag of words model. Specifically, for each term in our dataset, we will calculate a measure called Term Frequency, Inverse Document Frequency (tf-idf).

The tf-idf vector for each of sentence will be calculated using the following settings:

- sublinear_df is set to True to use a logarithmic form for frequency.
- min_df is the minimum numbers of documents a word must be present in to be kept.
- norm is set to l2, to ensure all our feature vectors have a euclidian norm of 1.
- ngram_range is set to (1, 2) to indicate that we want to consider both unigrams and bigrams.stop_words is set to "english" to remove all common pronouns ("a", "the", …) to reduce the number of noisy features.

The feature vector is used to generate the prediction model using Linear SVC. The objective of a Linear SVC (Support Vector Classifier) is to fit to the given data, returning a "best fit" hyperplane that divides or categorizes the given data. Both the features vectors and predictive

model are saved in the knowledge base. The confusion matrix for the generated model will be displayed to the user to decide if he wants to use it. This feature contributes to one of the sub goals.

d) Rules Engine

The rule engine is an embedded Drools rule engine as part of the webservices and is capable of forward chaining inference. Rules and facts are drawn out from the knowledge base and loaded into the working memory to help the human determine the severity classification of the ticket based on the dependency graph.

5. FUTURE IMPROVEMENTS

If given more time, the system could be improved to make it more intelligent, flexible and robust. The following outlines what could be done to further improve the cognitive abilities of the system:

Context Understanding

Although the system has already implemented some forms of natural language processing and understanding, more could be done to improve the context understanding detection. It is possible to construct new knowledge using frames, slot and fillers enhance our understanding of the context of the ticket contents.
For example, we require external system to also supply some identifier about the person who posted the content. Eg. Jenny (Staff ID 12345, Female, Manager) makes a complain about Tommy, who is the customer service officer supporting the System A

Frame1: Complain About Staff

Slot	Value

ID	12345
GENDER	Female
HASROLE	MANAGER
COMPLAIN	Tommy
SENTIMENT	Positive

Frame1: Complain About System

Slot	Value
ID	12345
GENDER	Female
HASROLE	MANAGER
COMPLAIN	System A
SENTIMENT	Negative

From the example above, it it possible to infer that Jenny is more unhappy about the system than Tommy. With better understanding of context, reasoning can be improved.

Improved Machine Learning Algorithms.

Only one algorithm LinearSVC was deployed in the supervised machine learning portion to demonstrate the capability. The system could be made smarter to autonomously try out a few algorithms, compare the accuracies and use the more suitable one for the inference.

Improved Rule Engine and Management.

System is configured to trigger high level inference if any of the sub goals is activated. An external rule engine could be explored to allow users more flexibility in changing the business process and rules.

6. CONCLUSION

Building the system itself reinforced some of the learning points taught in the courses. Overall, it was truly a multi-dimensional problem due to large number of cognitive system capabilities are put together into a hybrid rule-based reasoning system.

7. APPENDICES

Appendix A

See attached for Project Proposal.

Appendix B

Related Courses	Mapped System Functionalities
Machine Reasoning	1. The system is a reactive (or data driven) rule-based reasoning system that performs synthetic tasks in classification and assessment. 2. The system has an inference engine Drools that goes through a simple Recognise-Act Control cycle. Though Drools capable of both forward chaining and backward chaining inference, only forward chaining was implemented in the project. Rules and facts are loaded into the working memory to determine the severity classification of the ticket. 3. Rules are a knowledge representation of what the Helpdesk staff knows. These rules are stored in .drl files and that represents the knowledge base. Rules are derived from domain experts who use experiential knowledge and “rules-of-thumb” . If a customer sounds unhappy or negative, the staff is more likely to assign more attention to the person.

Cognitive Systems	1. The inference system (Drools) deployed an architecture similar to Soar Cognitive Architecture, where Knowledge (Rules) kept in Long-Term Memory, while States and transitions constructed in working memory to derive a solution. 2. The system also understands textual content written in English and is able to infer the sentiments of what was written and extracts interesting words to offload the helpdesk personnel's task of reading so that he can focus on resolving issues. 3. The supervised machine learning modules implemented a 3-class intent detection using word embeddings while the sentiment analysis implemented 5-class intent detection algorithms.
Reasoning Systems	1. The system deployed a uninformed search technique such as forward chaining to infer the severity of the helpdesk ticket.

Appendix C

See attached for User Guide.

Appendix D

Tng Sian Soo (Shawn)

1) Contributions to Project

Based on the business idea, I conceptualized the solution and designed the architecture. I designed, developed, and integrated the system using Java, Python and JavaScript programming languages. During design, I evaluated various technologies for their suitability and relevance of the overall system objectives. Significant effort was done in mimic how a data scientist or engineer will mine the data for machine learning. For example, to mitigate imbalance of the dataset uploaded by the customer, stratified sampling was done as part of the training process.

2) What learnt is most useful for you

I learnt how to apply some of the techniques taught from the courses, such as data-mining intent detection, rule-based reasoning systems and cognitive architecture. If given data, more could be done using associative analysis and profiling to determine the rules.

3) How you can apply the knowledge and skills in other situations or your workplaces

The knowledge and skills acquired could be applied at my workplace to build reasoning systems that can solve cognitive overload issues faced by my co-workers. The informed search algorithms learnt could also be applied to solve different kind of optimization problems, such as optimizing the utilization time of company vehicles and drivers.

Kumaravelu Varadharajan (Varad)

1) Contributions to Project

My contribution to this project is to provide the initial business idea since I have been in front office support in international banks in the past 18 years. I have personally experienced the backlog in the ticketing system and the negative effect of a critical ticket getting stuck in the pile of other issues. I currently work in an incident manager capacity and we have a manual triaging process which needs automation. I was able to provide the subject matter expertise and the business need for the same. Lastly, I also directed and edited the marketing video for the final system.

2) What learnt is most useful for you

Based on technical knowledge acquired about intelligent reasoning systems from the courses, I conceptualize the business model and how to implement them using Java.

3) How you can apply the knowledge and skills in other situations or your workplaces

Some of the learnings from this project can be directly implemented in my workplace conceptually and help the team to build an intelligent system that can automate some of the manual tasks.

Appendix E

The following was the Drool rules used for inference.

```
rule "Invalid Settings"
	dialect "mvel"
	salience 50
	when
		$setting: (
		SentimentSettings( sentimentMarker2-sentimentMarker1 < 2)
		)
	then
		insertLogical( new InvalidSentimentSettings($setting))
	end
rule "High Priority"
	dialect "mvel"
	salience 10
	when
		$tickets : Ticket()
		exists(SentimentBad()) or exists(MatchedP3WordSettings()) or exists(AIBad())
	then
		$tickets.setSeverity(2)
		insertLogical( new PriorityHighEvent($tickets))
	end
rule "Low Priority"
	dialect "mvel"
	salience 8
	when
		$tickets : Ticket()
		exists(SentimentGood()) or exists(MatchedP1WordSettings()) or exists(AIGood())
		exists(not PriorityHighEvent())
		exists(not PriorityMediumEvent())
	then
		$tickets.setSeverity(0)
		insertLogical( new PriorityLowEvent($tickets))
	end
rule "Medium Priority"
	dialect "mvel"
	salience 9
```

```
	when
		$tickets : Ticket()
		exists(SentimentNeutral()) or exists(MatchedP2WordSettings()) or exists(AINeutral() )
		exists(not PriorityHighEvent())
	then
		$tickets.setSeverity(1)
		insertLogical(new PriorityMediumEvent($tickets))
	end
rule "SentimentBad"
	dialect "mvel"
	salience 20
	when
		$settings: SentimentSettings()
		$tickets: Ticket(sentiments <= $settings.sentimentMarker1)
		exists(not InvalidSentimentSettings())
	then
		insertLogical( new SentimentBad($tickets))
	end
rule "SentimentGood"
	dialect "mvel"
	salience 18
	when
		$settings: SentimentSettings()
		$tickets: Ticket(sentiments >= $settings.sentimentMarker2)
		exists(not InvalidSentimentSettings())
	then
		insertLogical( new SentimentGood($tickets))
	end
rule "SentimentNeutral"
	dialect "mvel"
	salience 19
	when
		$settings: SentimentSettings()
		$tickets: Ticket(sentiments > $settings.sentimentMarker1)  && Ticket(sentiments < $settings.sentimentMarker2)
		exists(not InvalidSentimentSettings())
	then
		insertLogical(new SentimentNeutral($tickets))
	end
rule "AIBad"
	dialect "mvel"
	salience 20
	when
		$ai: AIEvent(priority==2)
	then
		insertLogical( new AIBad($ai))
	end
rule "AIGood"
	dialect "mvel"
	salience 18
	when
		$ai: AIEvent(priority==0)
	then
		insertLogical( new AIGood($ai))
	end
rule "AINeutral"
	dialect "mvel"
	salience 19
	when
		$ai: AIEvent(priority==1)
	then
		insertLogical(new AINeutral($ai))
	end
rule "Matched Priority 1 Words"
	dialect "mvel"
	salience 40
	activation-group "word match 1"
	when
		$setting: SentimentSettings()
		$tickets: Ticket()
```

```
                $word: P1WordSettings($tickets.bagofwords.contains(this.word))
        then
                insertLogical( new MatchedP1WordSettings($tickets))
        end
rule "Matched Priority 2 Words"
        dialect "mvel"
        salience 40
        activation-group "word match 2"
        when
                $setting: SentimentSettings()
                $tickets: Ticket()
                $word: P2WordSettings($tickets.bagofwords.contains(this.word))
        then
                insertLogical( new MatchedP2WordSettings($tickets))
        end
rule "Matched Priority 3 Words (High)"
        dialect "mvel"
        salience 40
        activation-group "word match 3"
        when
                $setting: SentimentSettings()
                $tickets: Ticket()
                $word: P3WordSettings($tickets.bagofwords.contains(this.word))
        then
                insertLogical( new MatchedP3WordSettings($tickets))
        end
```