AMERICAN UNIVERSITY OF BEIRUT

AUTOMATIC AND ADAPTIVE EXTRACTION OF ACTION KNOWLEDGE FROM PRODUCT REVIEWS

by
BOSAINAH MOHAMMAD AMRO

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Science in Business Analytics
to the Suliman S. Olayan School of Business
at the American University of Beirut

Beirut, Lebanon
April 2023

AMERICAN UNIVERSITY OF BEIRUT

AUTOMATIC AND ADAPTIVE EXTRACTION OF ACTION KNOWLEDGE FROM PRODUCT REVIEWS

by
BOSAINAH MOHAMMAD AMRO

Approved by:

Dr. Fouad Zablith, Associate Professor - Director Suliman S. Olayan School of Business	Advisor [illegible]
Dr. Wael Khreich, Assistant Professor Suliman S. Olayan School of Business	Advisor [illegible]
Dr. Lama Moussawi, Associate Professor Suliman S. Olayan School of Business	Member of Committee [illegible]
Dr. Sirine Taleb, Lecturer Suliman S. Olayan School of Business	Member of Committee Sirine Taleb

Date of thesis defense: April 28, 2023

AMERICAN UNIVERSITY OF BEIRUT

THESIS RELEASE FORM

Student Name: Amro Bosainah Mohammad

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of my thesis; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes

- [✓] As of the date of submission of my thesis
- ☐ After 1 year from the date of submission of my thesis .
- ☐ After 2 years from the date of submission of my thesis .
- ☐ After 3 years from the date of submission of my thesis .

Bosainah Amro [illegible] 08/05/2023

Signature Date

Acknowledgements

First, I would like to express my deepest gratitude to the blessings God have given us to complete this journey.

Second, I am indebted to my Professors, Dr Fouad Zablith and Dr Wael Khreich who have been supportive of my degree goals and who worked actively to provide me with the protected academic time to pursue those goals. I would like to thank also the members of the Committee.

Lastly, this endeavor would not have been possible without the support of all the precious beloved ones in my life.

Abstract of the Thesis of

Bosainah Mohammad Amro for Master of Science in Business Analytics

Major: Business Analytics (MSBA)

Title: Automatic and Adaptive Extraction of Action Knowledge from Product Reviews

Abstract:

Recommending products based on user experience and feedback was proved to be an effective marketing strategy. Product reviews are a promising source of knowledge in the product recommendation process. Much of the research done was mining textual data from reviews to extract sentiments, satisfaction level, and ratings. Little work was invested for extracting action knowledge and other semantics that can help with the recommendation process. Annotating action knowledge from customer reviews was done manually using the contribution of human annotators to read, annotate, and extract the tags and labels stated or deduced from content. How can we automate the extraction of action knowledge entities from unstructured product reviews, whether stated or predicted from context, in order to replace manual annotation tools? Moreover, How to automatically integrate product reviews in the recommendation process through representative knowledge graphs constructed from the predicted entities? How can human intervention help in the adaptation of ML models and to what incremental level of updating can the framework reach from the human-in-the-loop mechanism? This work proposes a framework to automate action knowledge extraction from customer reviews through machine learning models that will be incrementally improved through 'Human-in-the-loop' technique. This framework semantically annotates the actions expressed in product reviews, captures other related entities, links these entities to form a knowledge graph that serves the development of action-aware recommendation apps. The system is adapted and incrementally updated throughout continually retraining the models with approved correct data. To validate the solution proposed, an experimental evaluation protocol is applied to train the models with updated sets of approved annotations. The experiment revealed an improvement in the performance of the predictive models. In addition, the datasets collected by human annotators and used for model retraining were improved in terms of reducing the gap between classes of the models. The contribution of our work is to introduce a full consolidated automatic framework, joining multiple components, to construct an end-to-end prototype that generates the input of action-aware recommendation app. This automatic framework is proposed to automate the action knowledge extraction from product reviews and the construction of knowledge graphs used in recommendation systems, in an adaptive manner.

Table of Contents

ILLUSTRATIONS

Tables

Abbreviations

ML	Machine Learning
DBMS	Data Base Management System
KG	Knowledge Graph
RDF	Resource Description Framework
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
SVM	Support Vector Machine
RF	Random Forest
XGB	XG Boost
CV	Cross Validation
NLP	Natural Language Processing

Chapter 1

Introduction

Product reviews can tell you a lot, they reflect customer experiences, needs, feelings, and ratings which are the most important factors for product development. Review analysis is the act of going through product reviews and uncovering insights [1]. Multiple annotation and analysis techniques exist to process the large number of reviews and extract what is needed, but often these techniques are limited to ratings, customer satisfaction levels, and sentiments, in addition to being unable to integrate properly reviews in the recommendation process of products [2]. Our focus is on recommending products to customers based on reviews left by users. Most recommendation systems face challenges from products that change with time if they do rely on products specs only [2].

Knowledge graphs are mainly used to model knowledge, however manual KG can be time consuming and extensive due to several factors e.g the rapid increase of shared domain data. Knowledge graphs are characterized by their memory, dynamism, polysemy, and automation [3]. They represent main concepts and their associations necessary for reasoning purposes. They are used to enable sharing and reuse of knowledge and facilitate communication and reasoning among systems. One of the main benefits of KGs is to provide a standardised way of representing and sharing knowledge, which improves the collaboration of systems [4]. Most Knowledge graphs are constructed manually by ontology engineers who design entities and their relationships as per the context of domain. One of our objectives in this research is the automatic KG construction from unstructured data (review text) that will reduce the time required to build ontology graphs [5].

Based on the above, this thesis will invest its work answering the following research questions:

1. How can we automate the extraction of action knowledge semantics from unstructured product reviews, whether stated or predicted from context, in order to replace manual annotation tools?

2. How to automatically integrate product reviews in the recommendation process through representative knowledge graphs constructed from the predicted entities?

3. How can human intervention help in adapting ML models and to what incremental level of updating can the framework of automation reach from a human-in-the-loop mechanism?

This thesis is applied on a specific domain which is the unstructured customer reviews extracted from e-commerce websites to discuss recommendations from another perspective, recommendations that are based on action knowledge extracted from customer reviews. Actions or what a buyer can do with a product can influence his/her purchase decisions [6]. Action knowledge are the tags that reflect an action that was made by the user like saying *I use this laptop for watching movies.* In the preceding example, the action that can be extracted from this annotation is ‘Watch‘. Action knowledge can be extended to embrace not only action words, but also other related event semantics that can be included within the text or deduced from the context. These action knowledge semantics will be predicted using machine learning models. Since little work is given to address the actions that products enable their buyers to do and employ such actions in the recommendation process, our project will enhance this integration of actions in recommendations to better match products with customer needs. The integration of product reviews in the recommendation process proved its effectiveness, usefulness, and ease of use [7]. A framework is proposed to automatically annotate the actions expressed in product reviews using ML models, captures other related entities, links these entities to form a knowledge graph that will serve the development of action-aware recommendation apps by integrating product reviews [6]. This automatic framework will be incrementally updated by adapting ML models through human intervention.

The framework has an automatic tool that replaces the manual annotation tool by directly processing review text from the web, extracting all action knowledge entities mentioned in the reviews, enabling human annotators to check and edit machine predictions, and finally approving the correct group of entities for each annotation, to construct automatically its knowledge graph. This knowledge graph makes use of our existing ontology structure that was designed by ontology engineers for this specific domain data. The entities of the existing ontology are the actions that was done using the product, the agent that is doing the action, the environment where the action is taking place, the features used, as well as the sentiment of the context [6]. These entities are the predicted results of the framework's ML models. Moreover, one of the main goals is to assist human with machine learning to optimize performance, and combine human and machine intelligence to maximize accuracy. Frameworks having good data with simple algorithms is much more likely to generate better outcome than advanced algorithms with bad data [8]. A human-in-the-loop technique manages the adaptation of ML models in the framework to improve the automation process.

This integration of human force in the adaptation of ML models is proved throughout an experimental evaluation protocol, where checked annotation data is collected by human annotators using the automatic tool and fed to our existing models as new training data. Like most proofs, logic proofs usually begin with an initial state and then the results are observed within time. In our case, we started our experimental protocol and proof of human adaptation of machine learning models using the initial manually annotated training set of reviews. Then, we upgraded our data sets with newly checked data from our automatic framework, that involves human practices in checking machine results. The experiment starts to modernize the imbalanced manual annotated dataset that resulted initially with ambiguous accuracy scores, then it shows an increase in model's accuracy results. This increase builds for better performance in the future with more collected data through time.

The contribution of the work, is the development of an end-to-end prototype that processes raw review text to generate different knowledge graph entities and relations using multiple machine learning algorithms. Then, reaching a trusted automatic framework that can annotate and build knowledge graphs with minimal level of human intervention. Finally, implementing a credible feedback mechanism that increases the efficiency of our work and incrementally update the prototype.

1.1 Aims and Objectives

This thesis project focuses on automating the extraction of action knowledge from product reviews, as well as automating the construction of knowledge graphs. Our aims and objectives in this thesis are the following:

1. Assist humans in detecting and representing action knowledge from product reviews.
2. Build an end-to-end prototype starting from product URLs having review text, and ending with the final knowledge graph construction.
3. Implement an easy online accessible prototype used by human annotators to check, edit, and approve machine predictions on annotations. Besides, the prototype will enable the user to retrain models, and construct the final knowledge graph for approved annotation.
4. Combine human and machine intelligence to maximize accuracy by using human-in-the-loop technique to update the prototype incrementally.
5. Evaluate the automatic tool through hyper-parameter optimization technique as well as an adaptive retraining of the models.

1.2 Thesis Outline

The remaining of the thesis is structured as follows:

Chapter 2 highlights the main problem of the thesis. We will propose the literature review in this chapter to combine some related research work to our domain topic. The chapter analyzes the gap in previous work done and proposes a solution to our main problem.

Chapter 3 explains the methodology of this thesis that compares two main approaches: the past manual annotation process and the current automatic annotation framework.

Chapter 4 discusses the design of the prototype. A section will be set for testing the prototype for evaluation purposes. The purpose of this chapter is to evaluate the performance of the proposed automatic framework, and at which level it replaces the manual review annotation tools. Besides, another validation will take place in this chapter for retraining the models with collected correct data by a human annotator, to support the incremental update of the framework in this research.

Chapter 5 explains the implementation section of our work verifying all the results generated from our models and analysis. The chapter also proposes the hyperparameter optimization phase to fine-tune the framework's models. Best models will be retrained multiple times on updated sets of data collected by human annotators from the automatic tool, and results will be explained.

Finally, Chapter 6 concludes the thesis and summarizes discussions, future work and some personal reflection.

Chapter 2
Literature Review

2.1 Introduction

Marketing practice requires a deep understanding of customer needs especially when recommending products to customers. User generated content is a promising source for understanding their needs [9]. It can tell a lot more about the behaviour of customers when using a particular product. Much of the existing research on textual information processing has been focused on mining and retrieval of sentiment and satisfaction information, giving other semantics a little attention while being highly important. Text mining is the process of identifying facts, labels, tags, relationships, and assertions that are hidden within the mass of textual data generated every minute. Mining this textual data requires a variety of methodologies to process and extract information from unstructured forms of textual data. Natural language Processing is the main methodology in the field of text mining, with its both branches, Natural Language Understanding and Natural Language Generation.

One of the most potential aspects in the product recommendation process is integrating customer reviews about the products within the recommendation process. These reviews reflect customer experiences, needs, feelings, and ratings which are the most important factors for product development. Reviews can tell a lot; they can provide insights on high granular levels, thus employing these insights properly within the product recommendation system. Knowledge graphs and ontology have been known for their powerful semantic representation of knowledge [3]. Actions that products enable their buyers to perform are extremely significant in purchase decisions, as well as in the integration of action-based recommendations using knowledge graphs [5] — [6]. Action knowledge extraction from customer reviews is becoming an interesting area of research and it is needed to develop an automatic action knowledge application, to ease the process of annotation and knowledge graph construction.

2.2 Manual annotation and knowledge graph construction

Knowledge graphs can be designed manually, for example, existing efforts have annotated customer reviews manually using a proposed 'Review Annotator' extension on Chrome [6]. This annotator is a tool that helps classify annotations, but with human intervention for extracting, highlighting, and classifying the action, in order to be annotated in the review. For example, here is a review text annotated with a 'Study' action, a 'Student' agent, a 'University' environment, a 'Positive' valence, and a 'Laptop' object. A form is filled with these action knowledge tags using the chrome extension manual annotation tool.

I purchased this laptop for studying when i started university.

The manually annotated reviews is used to build the first knowledge graph to act later as a reference graph for correctness, completeness, and consistency of the automatically adapted ontology graphs. The graph is made of multiple classified nodes representing the semantics of the annotation, the attributes of each node, as well as the relationship between the nodes. Designing a knowledge graph should follow a standard declarative mapping rules that guarantees the systematic and sustainable workflow for constructing and maintaining a KG. Manual KG can be time consuming and extensive due to the increase in domain data that should be modeled in entities and relationships to be shared between systems. For example, the tool formulates the rules for the above entities as follows:

(StudyAction, isPartOf, annotation-md5)

(StudyAction, agent, Student)

(StudyAction, location, University)

(annotation-md5, hasValence, Positive)

2.3 Semantic Predictions

The categorization of textual data is perhaps the most dominant multi classification application [10]. Machine learning classifier for multi-label and multi-class classification models will manage the different semantics to be predicted from the textual data of reviews.

2.3.1 *Multi-Class Classification Models*

Prediction and correct voting are critical tasks in multi-class classification of textual data especially when having imbalanced data [11]. Besides, multi-class imbalanced learning is much harder than binary classification of one or 2 classes. It is a typical problem, because data is hard and expensive to collect, and we often collect and work with less data than we need to experiment even after dealing with big data in current times. Many real-world classification problems are usually imbalanced like fraud detection, medical diagnosis, and text classification [12]. Imbalance data affect the performance of multi-class classification models by giving very high accuracy scores that are ambiguous results. Our domain data is the unstructured textual data which is the product reviews. The contribution at this stage is to normalize our data sets by collecting as much data as possible to improve our data sets and have representative samples for every class in the model to resolve the imbalance of data issue. [13]

2.3.2 *Multi-Label Classification Model*

In many important data mining applications, such as text categorization, instances are associated with more than one class label. Many classification strategies have been introduced to deal with multi-label data [14]. The first strategy is converting the multi-labeled data into single labeled set of data and then solving it using single-label classifiers such as binary relevance and label power set transformations. The second approach is adapting and extending the single classifiers to cope with multi-labeled data. Some common adaptations are multi-label k-nearest neighbors, multi-label Naïve Bayes, multi label Ada boost, and others [14]. This intuitive approach to solving multi-label problem tends to decompose it into multiple independent binary classification problems (one per category), so that the existing single-label algorithms can be used.

2.4 Cosine Similarity Feature Matching

Cosine similarity is one of the popular distance measures used for text classification problems [15]. This technique measures the similarity between two vectors of words of an inner product space. Cosine similarity was used widely for different applications, like face verification where Nguyen and Bai [16] used it, linguistic applications like text summarization where Silber and McCoy [17] applied it and detecting emotions in the Arabic language text where Takçi and Güngör [18] benefited from.

2.5 Knowledge Graph construction

Semantic knowledge in web searches is usually referred to the term knowledge graph as Google proposed in 2012. KGs represents semantic networks that evolve with time and integrate with multiple types of applications like recommendation systems [19]. The KG captures entities and their relationships, following rules and patterns of the first manual KG done in the project. The semantics predicted using the multiple models explained above will be represented as entities in the KG, as well as the review text or annotations. Links and connections between entities will be then created based on the rules of the manual KG as per the below example.

(Annotation, hasTarget, review Body)

(Action, isPartOf, annotation)

(Feature, isPartOf, annotation)

(Action, agent, agent)

(Action, location, environment)

(Annotation, hasValence, valence)

2.6 The proposed solution

This thesis focuses on the automatic extraction of action knowledge derived from customer reviews as well as the automatic knowledge graph construction. This work investigates the potential of using appropriate machine learning techniques from customer text reviews. These semantics will cover the action, the agent, the environment in which the action is taking place, the features of the products that are used for each specific action annotated, as well as the sentiment and polarity that promotes us to cluster data based on cons and pros [20]. The main goal behind this work is to design an automatic end-to-end framework starting from unstructured text sources providing an openly accessible knowledge graph that explicitly links user actions, products, and reviews to continuously embed such data into the recommendation process [3] — [6]. Both the machine learning strategy and the data annotation strategy are closely intertwined, and better accuracy for the models can be obtained when combining the two approaches efficiently [8]. The framework embraces the annotation of data strategy through human-in-the-loop technique, that will take place in parallel to our developed simple machine learning models, that works on extracting action knowledge from customer reviews.

This thesis examines the performance of information drawn from the different levels of linguistic text features [21]. The first major label to be predicted will be the action done using the product. Users will mention that they often use the respective product for drawing, streaming movies, studying, gaming and so on. The model focuses on the acquisition of the action label in the text, and exploit the dependency of some others with the action. Independent labels will employ independent multi-class classifiers like other common approaches [22].

We can have multiple actions in one review that probably needs multiple annotations. But a multi-label classification model won't perform well using our initial manual annotated data set that is an imbalanced one. The multi-label classification approach for predicting multiple actions in one annotation will be part of our future work and developments of the framework when a vast amount of data will be collected for a good fit in a multi-label classification model. A well performing predicting classifier that fits our initial set of data is the multi-class classifiers. The models are required to predict 5 different labels for every annotation: action, agent, environment, valence, and object.

Feature label is predicted using a cosine similarity measure. This similarity is measured between the vector of words of a text review and the vector of words of a lexicon of product features. The lexicon will be developed with time as new features will be captured during the human-in-the-loop process. Multiple features can be matched with the context of a single sentence. For example, the below sentence matches with 'Memory' feature label as its context reveal that it talks about how much storage this product can handle. *I can run all my favorite games and more on this!*

Knowledge graph construction is based on a declared mapped set of rules that link predicted entities with their respective relationships based on the manual KG designed.

In the following chapter, the thesis introduces the methodology approaches that are tackled to demonstrate our solution proposed. The methodology emphasizes the automatic framework that serves our purpose in automating action knowledge extraction and knowledge graph construction in an adaptive way.

Chapter 3
Methodology

3.1 Introduction

This work proposes an automatic framework that is based on three different phases. A Generation phase where machine learning models stand, a Refinement phase where human intervention takes place to check for anomalies and errors in machine predictions, and finally a Mapping phase where a person makes the final decision and votes for the best outcome of the prior two stages to complete and construct the knowledge graph we need. This chapter emphasizes the automatic framework and its adaptive incremental learning process that stands as a solution for the manual review annotation and action knowledge extraction from customer reviews. The proposed automatic framework is monitored and controlled by human at a certain level reaching a future optimal stage, in which all machine learning models reach high trusted performance level with cumulative learning curve. This performance will be reached out by developing the training set of data continually under human supervision. This frequent improvement of data set and update in models are tested and evaluated through an experimental protocol, where models are retrained with new data collected by human annotators. The framework takes raw review text data from an e-commerce website as an input, generating a final knowledge graph for every annotation of the review as an output.

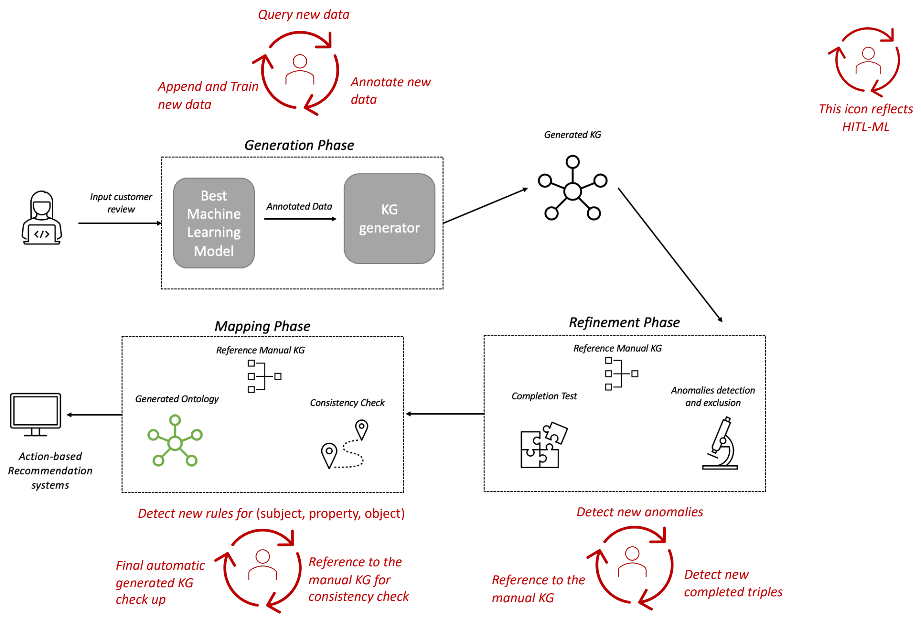

Query new data
Append and Train new data
Annotate new data
This icon reflects HITL-ML
Generation Phase
Input customer review
Best Machine Learning Model
Annotated Data
KG generator
Generated KG
Mapping Phase
Reference Manual KG
Generated Ontology
Consistency Check
Refinement Phase
Reference Manual KG
Completion Test
Anomalies detection and exclusion
Action-based Recommendation systems
Detect new rules for (subject, property, object)
Final automatic generated KG check up
Reference to the manual KG for consistency check
Detect new anomalies
Reference to the manual KG
Detect new completed triples

Figure 3.1: The Automatic Ontology Generation Framework

3.2 Automatic ontology generation framework

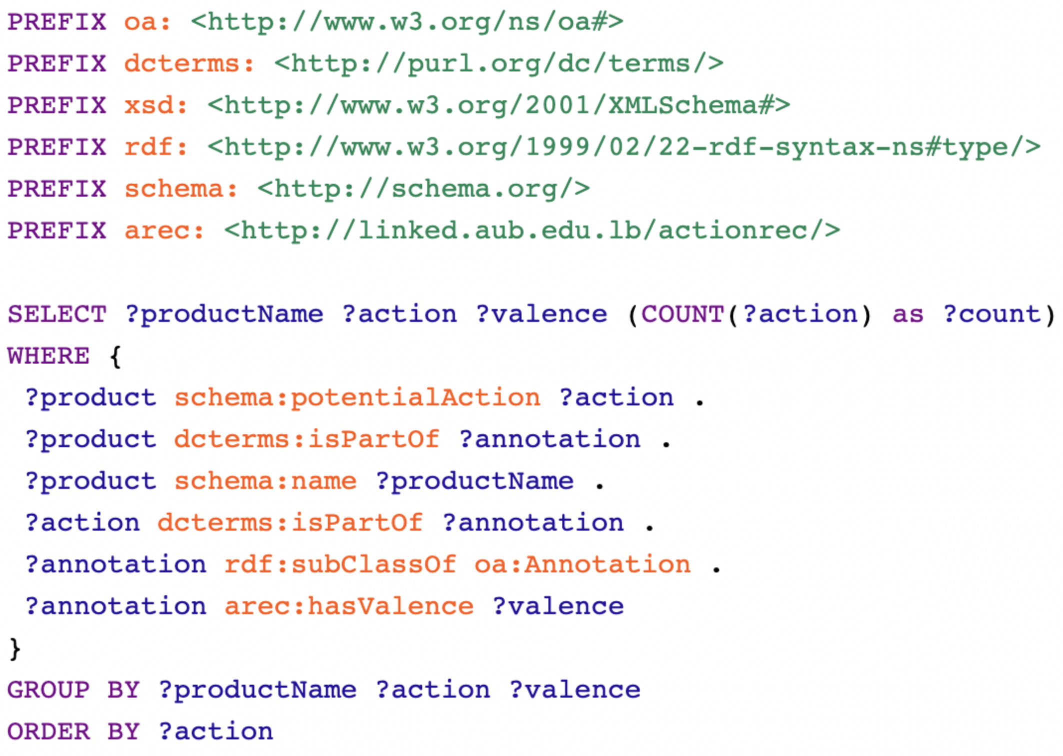
At this stage of the research, a deep understanding of three phases takes place: Generation Phase, Refinement Phase, and Mapping Phase. The three phases are sponsored by HITL ML (Human-in-the-loop machine learning) system for cumulative learning for the machines hosted in the framework.

3.2.1 *Generation phase*

The generation phase in Figure 3.1, embraces all machine learning models which were trained on set of reviews to predict action knowledge semantics from text. This step carries out this repetitive and mundane task that should be done continuously in this framework, so that we free up time and resources like human intervention. This will be the first stage where we level down the human intervention with an accurate trained machine learning model for annotating text reviews and preparing structured data for knowledge graph construction step.

Data Extraction

The manual annotated set of data explained in Chapter 2 is used to recommend products for a recommendation website for electronic products. This thesis started its work with a specific domain, which is reviews from products belonging to the ‘Computers and Tablets‘ category in Best Buy. Initially, our work will be limited to this specific domain in order to open the research gate to other domains in the future. Annotations done manually were based on the existence of an action tag in the sentences. If an action was found, then the sentence can be annotated and the prediction can be expanded to retrieve other events like agent, environment, valence, and object. Every annotation is represented by a knowledge graph stored in KG Triple store. To extract these KGs, a sparql query is modeled containing a set of constraints needed to get all reviews with their annotations and other sentences that weren't set as annotations, see example of query in Figure 3.2. Both types of sentences, informative (refer to an action) and non-informative (doesn't refer to an action) are used in our models, since having some negative occurrences in the data set improves the classification of the model. The data extracted for training the models is cleaned as we retrieved it from a structured source of data. Cleansing is not required at the generation phase, but will be required at later stages when the automatic tool extracts unstructured form of data like customer reviews directly from the web.



```
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#type/>
PREFIX schema: <http://schema.org/>
PREFIX arec: <http://linked.aub.edu.lb/actionrec/>

SELECT ?productName ?action ?valence (COUNT(?action) as ?count)
WHERE {
 ?product schema:potentialAction ?action .
 ?product dcterms:isPartOf ?annotation .
 ?product schema:name ?productName .
 ?action dcterms:isPartOf ?annotation .
 ?annotation rdf:subClassOf oa:Annotation .
 ?annotation arec:hasValence ?valence
}
GROUP BY ?productName ?action ?valence
ORDER BY ?action
```

Figure 3.2: Sparql Query to Fetch KG Data from Triple Store

Machine Learning Models for Action Knowledge Predictions

Machine learning models that proved to be good for natural language processing are plenty, but this project works on traditional models using TF-IDF features. TF-IDF weights words based on relevance, this technique is used to determine that the words with the highest relevance are the most important for featuring our model. This can be used to help summarize reviews more efficiently, or to simply determine keywords or tags for a review text. Traditional classifiers that will be used for our text data are mainly logistic regression, SVMs, Random Forest, and Naïve Bayes. The main advantage of using traditional machine learning classifiers is its speed and relative simplicity. In addition, these algorithms are human interpretable, that are essential for failure analysis, model improvement and the discovery of insights and statistical regularities. The model will be trained jointly across multiple tasks and action knowledge extraction. [23]. The classifiers will act as multi-class ones since our project requires predicting multiple classes for each semantic. A future window will be opened to multi-label classification model for the action entity as we encountered a little number of examples having more than one action in the same sentence like saying

I mainly use this to draw, read, or stream movies/TV and I'm very impressed.

Data is split on 80 percent for training and 20 percent for testing and validation. The best performing model on the testing set is chosen to be part of our prototype system and deployed to be live in order to process new reviews in real-time.

The Reference Ontology for formulating the triplets of the automatic knowledge graphs

The generated output of the best machine learning model is the input to a knowledge graph generator that is used to get the preliminary KG in the form of triples in RDF format (subject, predicate, object) with their corresponding confidence score. The RDF is a directed labeled graph data format for representing information in the web. The generated KG is reinforced with the manual ones generated above for reference. Triplets that should be constructed for every annotation are a total of 37 different (subject, predicate, object) RDF, see example in Figure 3.3. These triplets are based on the preliminary knowledge graph that is used for a website recommendation system based on action knowledge manually annotated [6]

```
# Rule 1: <schema:Action><dct:isPartOf><oa:Annotation>

# Rule 2: <arec:Feature><dct:isPartOf><oa:Annotation>

# Rule 3: <arec:Ability><dct:isPartOf><oa:Annotation>

# Rule 4:<schema:Location><dct:isPartOf><oa:Annotation>

# Rule 5: <schema:Object><dct:isPartOf><oa:Annotation>

# Rule 6: <arec:Agent><arec:hasAbility><arec:Ability>

# Rule 7: <arec:Ability><arec:supports><schema:Action>

# Rule 8: <schema:Action><schema:agent><arec:Agent>

# Rule 9: <schema:Action><schema:location><schema:Location>

# Rule 10: <schema:Action><schema:object><schema:Object>

# Rule 11: <oa:Annotation><os:hasTarget><schema:Review>

# Rule 12: <arec:Feature><arec:supports><schema:Action>

# Rule 13: <oa:Annotation><arec:hasValence><valence>

# Rule 14: <schema:Review><schema:reviewBody><review text>
```

Figure 3.3: Example of Triplets in RDF Format

3.2.2 ***Refinement Phase***

The reference ontology that was constructed manually in the first part of the project is used to evaluate and verify the generated KGs. At this stage, we will be excluding anomalies from our generated KGs. This step ensures the elimination of irrelevant, illogical, and unrelated nodes (objects) and relations as per the rules and patterns of the reference KG. This is done using embedding for all actions, environments, agents, and features that might be involved in our domain.

Finally, to complete the refinement phase referred in Figure 3.1, the generated KGs should pass through a completion test to ensure that none of the generated KGs are incomplete. Here we used the knowledge graph embedding model that is based on scoring technique for each triple, so that we exclude incomplete triples (The completion scoring technique was done by Dat Quoc Nguyen) [24].

This phase of our automatic framework that translates the processes of refinement described in Figure 3.1 is part of annotators job, where annotations done by machine learning models are double checked and corrected in a simple easy way by users. By checking and editing machine results by human annotators, new entities are assigned for some annotations. These newly checked annotations are stored separately as new sets of annotations to be used in retraining our ML models.

Refining our data and updating the models recurrently fine tunes our framework and filters the triplets from any falsely predicted action knowledge and semantics. Refinement stage is prior to the mapping phase that pushes final knowledge graph rules to the triple store.

3.2.3 *Mapping Phase*

In the last phase structured in Figure 3.1 of the automation framework, the KG should go through a consistency check, which aims to solve the issue of interoperability to ensure that all objects and relations are consistent. For example, if (subject, property, object) with Microsoft surface pro as the subject and a relation of 'hasFeature' is the property, linked to Keyboard feature as the object, then 'hasFeature' relation makes Microsoft surface pro belongs to the domain class (Product). Inconsistency occurs when a subject isn't placed in its correct domain (Method developed by Péron et al) [25].

This consistency is also based on the reference Knowledge Graph, and ends up with a final generated knowledge graph that includes the new annotated reviews. To maintain clean pushed RDF rules to the triplestore, the automatic tool manages a checked annotation portal that gives an authorized user a privilege to vote for the best edited annotation. Checked annotation should pass through this phase for final processing, the edited labels generated by each annotator are displayed in this portal for the user to decide on the best action knowledge semantics assigned for every annotation. Mapping these action knowledge with their RDF rules is managed by one person who is the only one authorized to push correct annotations to KG triplestore.

3.2.4 *Human-in-the-loop machine learning (HITL-ML)*

Google's search engine has always been a form of HITL machine learning by providing users with content they wish to find based on words from their query. They created a system to display options, make predictions, listen to user feedback, validate accuracy of predictions and modify their predictions [26].

In HITL systems, the same input given for the machine models are handled manually by a human. Data annotation, Knowledge graph construction, fine-tuning of data and KG objects are done manually and then the results of the manual KG and the automatic generated KG will be weighed. This makes sure that weak points and errors are detected from the initial stage of the framework which increases the framework efficiency and saves time. This might be repeated multiple times initially until we reach out for similar (approximately equal) scores for the manual KG and the machine generated one [26].

The HITL system integrates human knowledge and user experience in our machine learning framework as feedback loops, which increases reliability and machine robustness. Human intervenes in this framework at each stage to ensure that the adaptation of the automatic knowledge graphs constructed is aligned with the refined knowledge graph manually done [27]. As discussed previously, the first manual KG references to the basic rules and patterns of RDF triplets that are used to construct new KGs. Removing anomalies and the completion test at this stage should be based on these rules and patterns, training our machine on newly generated ones. The constant human feedback loop in the framework ensures the constant retraining and fine-tuning of the model. This feedback loop is needed till we acquire a streamlined data annotation pipeline to make sure the model runs smoothly with a minimum level of human intervention [28].

The continuous optimization and improvement in production of the machine learning model using human power is necessary to adapt the model to the ever-changing data. The loop of querying, annotating, appending, and training will continuously work around until we reach the high accurate real-time data labeling of customer reviews needed to construct a good knowledge graph [28]. This stage of human intervention is reflected in a specific page of the prototype system we developed. This page manages training the models continuously with the newly checked annotations by the users. This periodic training of models adapts the existing models to the updated training set involving human force in the loop by correcting all falsely predicted outcomes from the ML models.

3.3 Testing and Evaluating the Automatic Framework

Evaluating the methodology described above is done through demonstrating a full automatic prototype. It is the proof of concept for the automatic action knowledge extraction and knowledge graph construction processes. It is a streamlit online web app consisted on four different pages, each reflecting a part of the automatic framework structure in Figure 3.1. Two types of users are authorized to work on this app, the admin who is responsible of multiple tasks and the annotator who accesses only one page of the app. Privileges for each user are explained in Chapter 4.

Testing the prototype enhances the development approach through a model hyper-parameter optimization phase. Multiple parameters are adjusted for machine learning classifiers to get the best set of parameters that fits each model. Another testing takes place to evaluate model performance and accuracy growth when checked data is collected by annotators. This strategy is highly important to validate the efficiency of human intervention in the automatic framework this research proposes. The aspect of this approach is collecting continually new training data under human supervision and approval to retrain models and increase their accuracy, since it is very common in the industry of machine learning to improve models performance by annotating more training data [8]. Collecting training data from human annotators should be as simple and friendly as possible to avoid the annoyance of repetitive tasks. The framework simply requires the annotators to edit machine learning results, thus checking on annotations predicted before entering the training phase again. These testing sets help leverage the performance of the models with time. Training the models won't be once, it is performed periodically after collecting new batch of reviews, predicting semantics, confirming or editing them by annotators.

Chapter 4

Evaluation

4.1 Introduction

This chapter discusses the evaluation of our approach using the automatic framework we proposed in this project. The chapter explains the prototype of the framework on both levels; design and implementation.

4.2 Prototype

Our automatic framework connected multiple endpoints together to structure an end-to-end prototype for a guided flow of data. The prototype consists of a data engineer offline portal, an annotator front-end page, an admin front-end page, MySQL DBMS, KG Triplestore, as well as a streamlit app that hosts ML models and multiple functions.

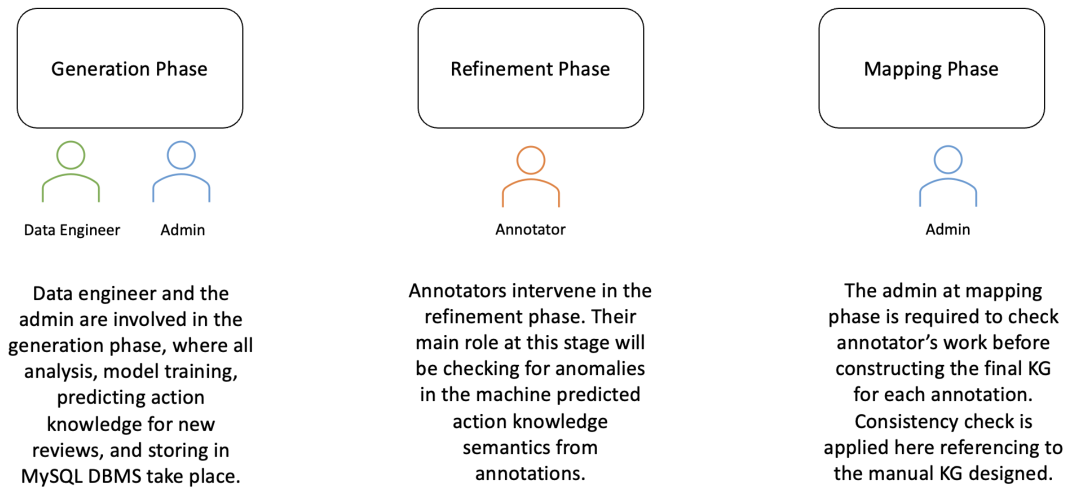

Generation Phase
Refinement Phase
Mapping Phase
Data Engineer
Admin
Annotator
Admin
Data engineer and the admin are involved in the generation phase, where all analysis, model training, predicting action knowledge for new reviews, and storing in MySQL DBMS take place.
Annotators intervene in the refinement phase. Their main role at this stage will be checking for anomalies in the machine predicted action knowledge semantics from annotations.
The admin at mapping phase is required to check annotator's work before constructing the final KG for each annotation. Consistency check is applied here referencing to the manual KG designed.

Figure 4.1: Users Responsible for each Phase in our Approach

Distribution of roles and privileges explained in Figure 4.1 is modeled in a full consolidated prototype. Refer to Figure 4.2 for the prototype's flow of work. All links for streamlit web app are in the Appendix B of this thesis.

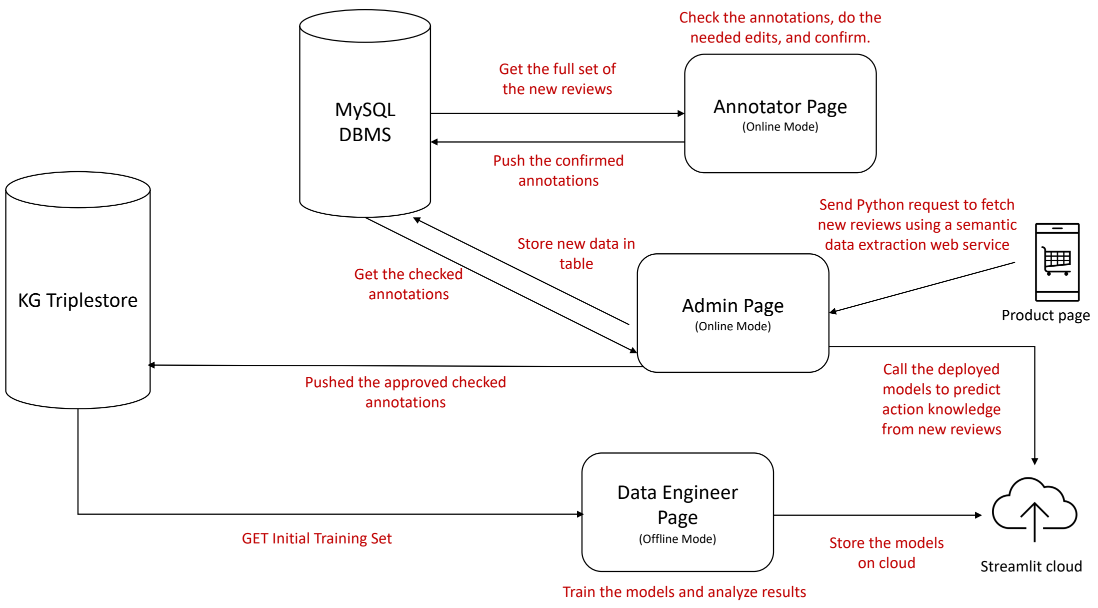
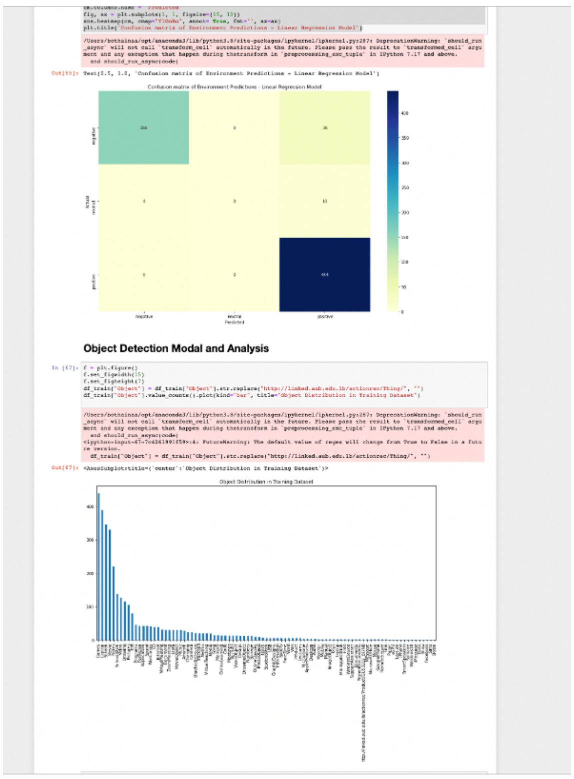

MySQL DBMS
Check the annotations, do the needed edits, and confirm.
Get the full set of the new reviews
Annotator Page (Online Mode)
Push the confirmed annotations
KG Triplestore
Send Python request to fetch new reviews using a semantic data extraction web service
Store new data in table
Get the checked annotations
Admin Page (Online Mode)
Product page
Pushed the approved checked annotations
Call the deployed models to predict action knowledge from new reviews
Data Engineer Page (Offline Mode)
GET Initial Training Set
Store the models on cloud
Streamlit cloud
Train the models and analyze results

Figure 4.2: The Automatic Framework Prototype

4.2.1 *Prototype's Design*

Data Engineer Offline Portal

The data engineer portal is an offline page, where all the initial training, testing and analysis took place. Refer to Figure 4.3. Firstly, the data engineer gets the training data from KG triplestore using SparqlWrapper package in python [29]. A query was posted to fetch out the needed data having all required labels for training. Secondly, the best models chosen after a deep analysis and study, are trained. Errors for each model are analyzed through an error heat map showing the areas of error condensation. This analysis can help the data engineer manipulate data if needed in case of high error rate at a specific area. Finally, the data engineer deploys the approved models on the cloud, to ensure an open source framework that can be accessed easily.



```
cm.columns.name = 'Predicted'
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
sns.heatmap(cm, cmap="YlGnBu", annot= True, fmt='', ax=ax)
plt.title('Confusion matrix of Environment Predictions - Linear Regression Model')
```

```
/Users/bothainaa/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

```
Out[65]: Text(0.5, 1.0, 'Confusion matrix of Environment Predictions - Linear Regression Model')
```

Confusion matrix of Environment Predictions - Linear Regression Model
negative
actual
positive
Predicted
neutral

Object Detection Modal and Analysis

```
In [67]: f = plt.figure()
f.set_figwidth(15)
f.set_figheight(7)
df_train["Object"] = df_train["Object"].str.replace("http://linked.aub.edu.lb/actionrec/Thing/", "")
df_train["Object"].value_counts().plot(kind="bar", title="Object Distribution in Training Dataset")
```

```
/Users/bothainaa/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
<ipython-input-67-7c4261991f59>:4: FutureWarning: The default value of regex will change from True to False in a futu
re version.
  df_train["Object"] = df_train["Object"].str.replace("http://linked.aub.edu.lb/actionrec/Thing/", "")
```

```
Out[67]: <AxesSubplot:title={'center':'Object Distribution in Training Dataset'}>
```

Figure 4.3: Data Engineer Offline Portal

The Admin Page

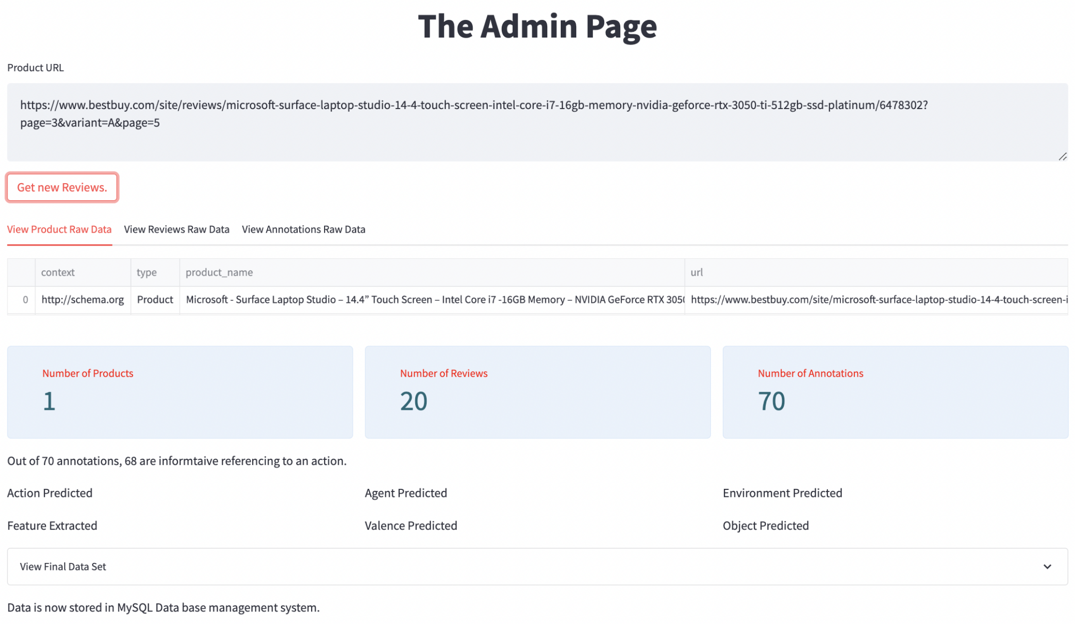
The admin has the privilege to access three pages of the app. The main page for getting new reviews from e-commerce website. A second page to check the work done by annotators and vote for the annotation to be pushed into triplestore with its labels. A third page for retraining the machine learning models and reinforce them with the checked annotations done by annotators. All three pages will be explicitly explained throughout this section. Links for pages are available in Appendix B.

Get New Reviews Page

The admin page is a streamlit app that enables an authenticated user to fetch new batch of reviews from product URL. The page accepts a product URL as an input, retrieve the reviews in this respective page using an http request from the requests package in python. The app calls then the models hosted in the streamlit cloud and test them on the new review set to predict semantics. The steps for prediction are tracked and displayed in a simple way to make sure the models are running in real-time and accessible any time. Data is stored in MySQL DBMS, through the python connection using MySQL connector and SQL alchemy packages. There are many features in the admin main page as per Figure 4.4 that is responsible for retrieving

[0]Link to ‘Get New Reviews‘ page: https://bma52-reviews-actions-getnewreviews-lojlss.streamlit.app/

new data from best buy product URL.



The Admin Page

Product URL

https://www.bestbuy.com/site/reviews/microsoft-surface-laptop-studio-14-4-touch-screen-intel-core-i7-16gb-memory-nvidia-geforce-rtx-3050-ti-512gb-ssd-platinum/6478302?page=3&variant=A&page=5

	context	type	product_name	url
0	http://schema.org	Product	Microsoft - Surface Laptop Studio – 14.4” Touch Screen – Intel Core i7 -16GB Memory – NVIDIA GeForce RTX 305	https://www.bestbuy.com/site/microsoft-surface-laptop-studio-14-4-touch-screen-i

Number of Products
1

Number of Reviews
20

Number of Annotations
70

Out of 70 annotations, 68 are informtaive referencing to an action.

Action Predicted

Agent Predicted

Environment Predicted

Feature Extracted

Valence Predicted

Object Predicted

Data is now stored in MySQL Data base management system.

Figure 4.4: Retrieve new reviews admin page

1. The user can view the product information retrieved by clicking on the three horizontal tabs.
   (a) **View Product Raw Data** shows you the main fields for the product selected from the web page.
   (b) **View Reviews Raw Data** shows you the reviews that were posted in the respective product page as well as their fields of information.
   (c) **View Annotations Raw Data** shows you the aggregation of the reviews by sentences. Each review text is split by sentences to be able to detect the labels for each sentence separately.
2. Three metrics display the numbers of total products, reviews, and sentences retrieved from this product URL.
3. An initial layer classifies the sentences into informative and non-informative depending on a machine learning classifier, that predicts whether the sentence might have an action or not.
4. Six live models run in the back end to predict the different semantics for each sentence. After the models finish their predictions, a consolidated set of data having all the reviews with their sentences and predicted labels, can be viewed through a tab that can be expanded. The page displays the predicting semantic load in a nice way.

5. The user can inspect the sentences and their predicted labels by clicking on the tab ‘View Final Data Set‘.

6. The page stores this final data set and the raw data retrieved in MySQL. A note is displayed at the bottom, once all data is stored in MySQL.

Checked Annotations Page

Considering the case when annotators may be unreliable, but also when their expertise vary depending on the data they observe, the admin have the privilege of checking on what annotators edit in machine results, and then decides on pushing to the triplestore the correct entities or labels for KG construction. The admin assesses the need to aggregate and make inferences about the collection data from multiple annotators. This task is done by navigating the ‘Checked Annotation‘ page. The page loads initially with a filter to choose the number of annotators, in which each annotation should be checked by at least 2 annotators, or at least 3 annotators, refer to Figure 4.5.

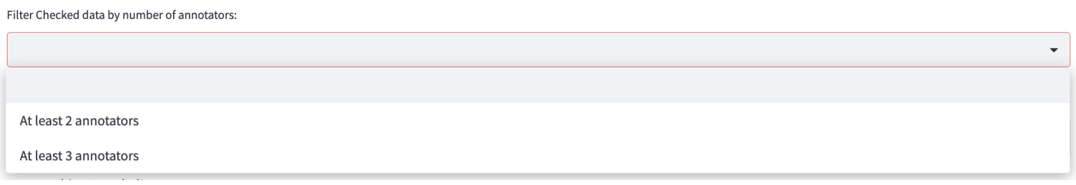
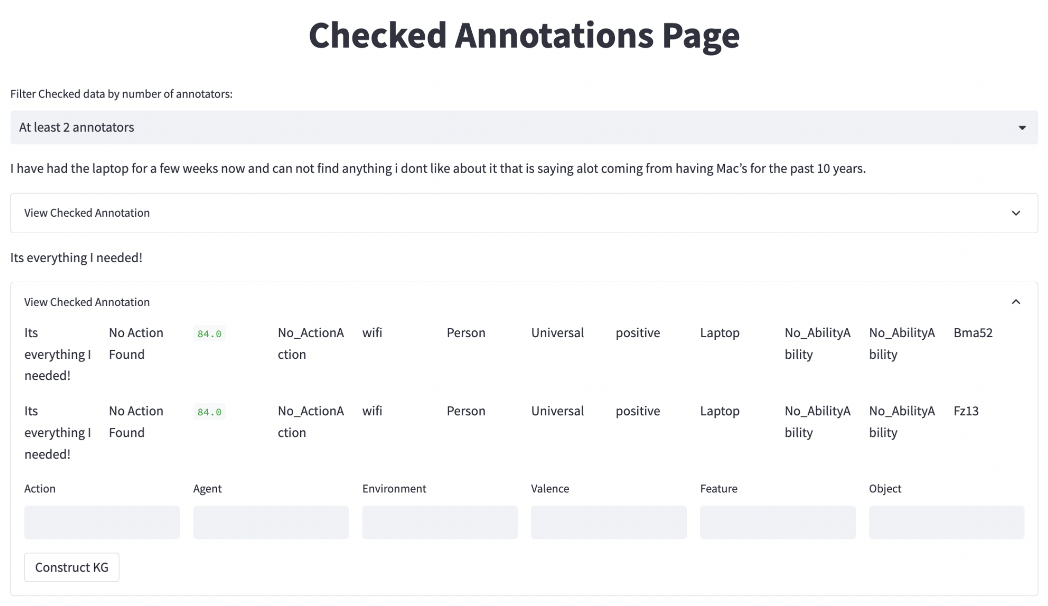


Figure 4.5: Checked by number of annotators filter

This page loads a table including all the checked annotations by different annotators. A filter by annotator's username is applied at the top of the page to help the admin easily add constraints on the checked annotations they want to view. For example, if they want to view all annotations that were checked by three annotators, they simply select all annotators that they want in the multi select box. When reviewing the annotation and its labels in the table loaded, the admin has the privilege to vote for which annotation to be pushed into triplestore for constructing the knowledge graph of this particular annotation. A ‘Construct KG‘ is assigned for each row of the table, thus for each annotation reviewed by an annotator. Refer to Figure 4.6.

[0]Link to ‘Checked Annotations‘ page: https://bma52-reviews-actions-getnewreviews-lojlss.streamlit.app/CheckedAnnotations



Checked Annotations Page

Filter Checked data by number of annotators:

At least 2 annotators

I have had the laptop for a few weeks now and can not find anything i dont like about it that is saying alot coming from having Mac's for the past 10 years.

View Checked Annotation

Its everything I needed!

View Checked Annotation

Its everything I needed!	No Action Found	84.0	No_ActionAction	wifi	Person	Universal	positive	Laptop	No_AbilityAbility	No_AbilityAbility	Bma52
Its everything I needed!	No Action Found	84.0	No_ActionAction	wifi	Person	Universal	positive	Laptop	No_AbilityAbility	No_AbilityAbility	Fz13

Action Agent Environment Valence Feature Object

Construct KG

Figure 4.6: Checked annotations table with KG buttons

Retrain Models Page

Retraining models is also one of the privileges given to the admin. A table in MYSQL is initially created to store the performance metrics of our ML models. The admin through his/her port can access this table and view these metrics explicitly, to decide on which model to be retrained and updated. The retraining models page displays a report for each trained model, for the admin to inspect performance and metrics before deciding whether this updated version is better than the previous one or not. The models are automatically retrained when loading the page, but models won't be updated and saved unless the admin decides. Refer to Figure 4.7.

[0]Link to 'Retrain Models' page: https://bma52-reviews-actions-getnewreviews-lojlss.streamlit.app/RetrainModels

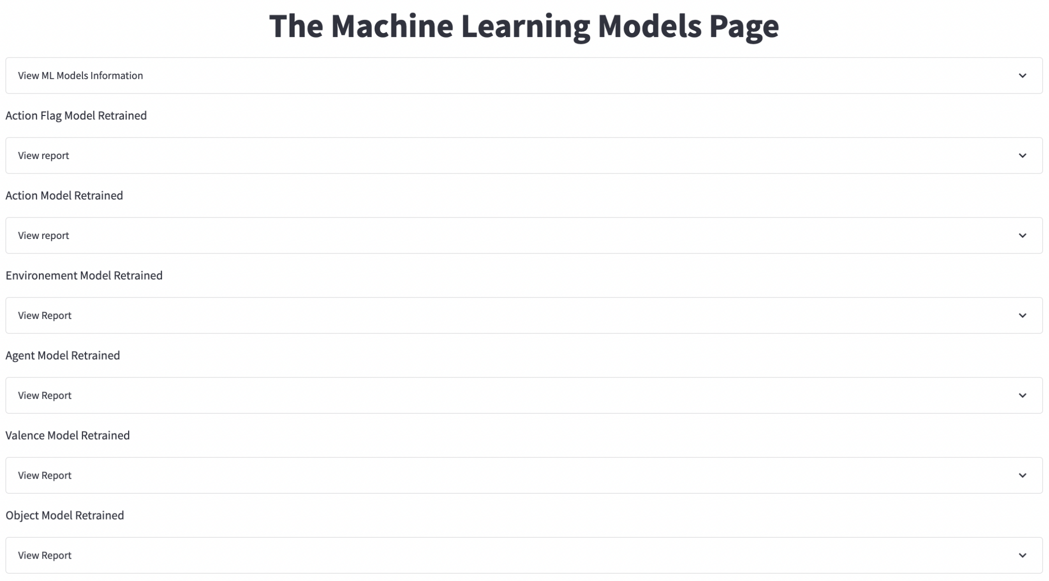
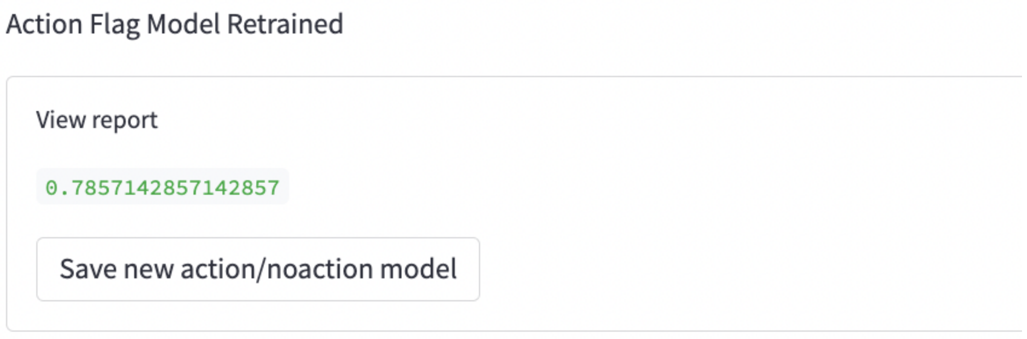


Figure 4.7: Retrain Models Page

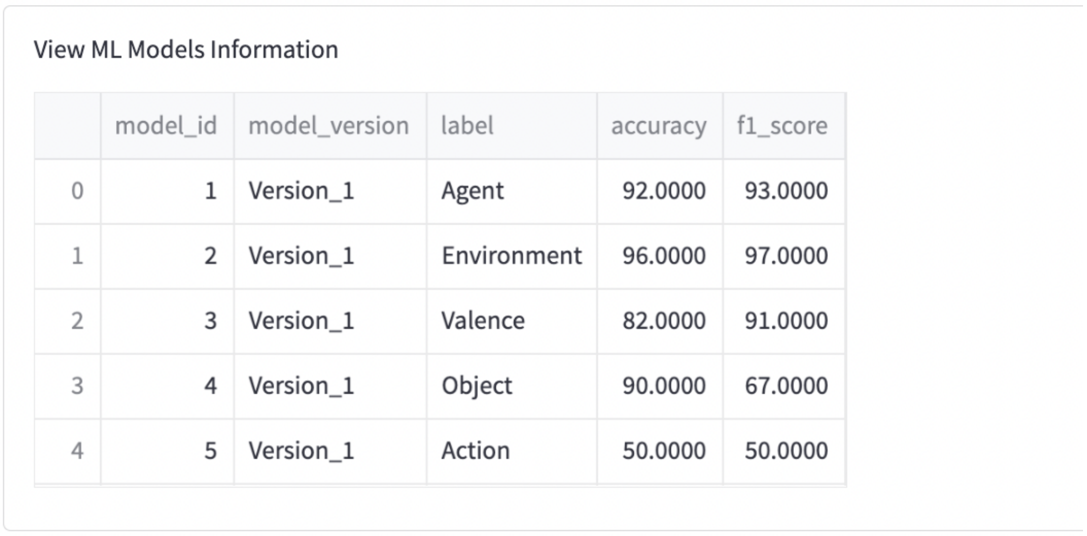
1. If the models performed better, the admin can save it by clicking on 'Save this model' button, refer to figure 4.8.
2. If the model didn't perform better, the admin can ignore saving it and monitor its performance later on with more training data.



```
0.7857142857142857
```

Figure 4.8: View Model Report tab

All model versions are stored in a table in MySQL for reference, so that the user can monitor the historical information of every model, and check how the incremental update is taking place and our models are being adaptive with the new trained and checked data. Refer to Figure 4.9 for model information table about trained versions and their respective accuracy scores.



View ML Models Information

	model_id	model_version	label	accuracy	f1_score
0	1	Version_1	Agent	92.0000	93.0000
1	2	Version_1	Environment	96.0000	97.0000
2	3	Version_1	Valence	82.0000	91.0000
3	4	Version_1	Object	90.0000	67.0000
4	5	Version_1	Action	50.0000	50.0000

Figure 4.9: Table of Models Versions from MySQL

Human in the loop mechanism is the crux of our project. How retraining models from human feedback continuously strengthen the models performance. Trusting our models rely on the proper check a human do on machine predictions. The checked and edited entities that a human annotator apply on the annotation replaces the predicted machine entities. All checked annotations are stored in a new table in MySQL to use them later in retraining our models. This continuous human check and model learning optimizes our framework. Optimization is reached after multiple full cycles in this prototype. The purpose of the Retraining models page is to monitor and record these updates in model's performance. Each cycle reproduces a new version of the model, and the admin has the privilege to replace the previous saved model, or waits for a better performing one to rise.

The Annotators Page

Learning from multiple annotators has become an important critical task in machine learning problems. Varying expertise among annotators is one of the main ramifications for better data collection, since some action knowledge tags are predicted from context rather than stated. The prototype introduces a simple app that enables annotators to explicitly view the review text with its annotations and predicted labels, sentence by sentence in an organized way. The full review text is splited into sentences, these sentences are called annotations. The page is divided into containers, each consisting of one sentence of the review and its predicted labels. The annotators role is to double check the labels for each annotation and edit it whenever it is wrong. A drop down selection box appears if the annotator confirmed that this label is ‘Not correct‘. The annotator should check the ‘Confirm annotation‘ Checkbox placed at the bottom right of every annotation when finishing his/her edits.

Use cases that can be applied in the annotator’s page:

1. Reviews are loaded after an annotator enters his/her username by selecting one of the authorized users from the select box that will appear at the top of the page, refer to Part 1 in Figure 4.10.

2. Review text is displayed with the product name, each annotation with its predicted labels in a container to easily separate between annotations, refer to Figure 4.11.

3. Annotation for every review text can be sorted by their action probability, this promotes an easy check up process from the high accurate predictions to the low accurate ones, refer to Part 2 in Figure 4.10.

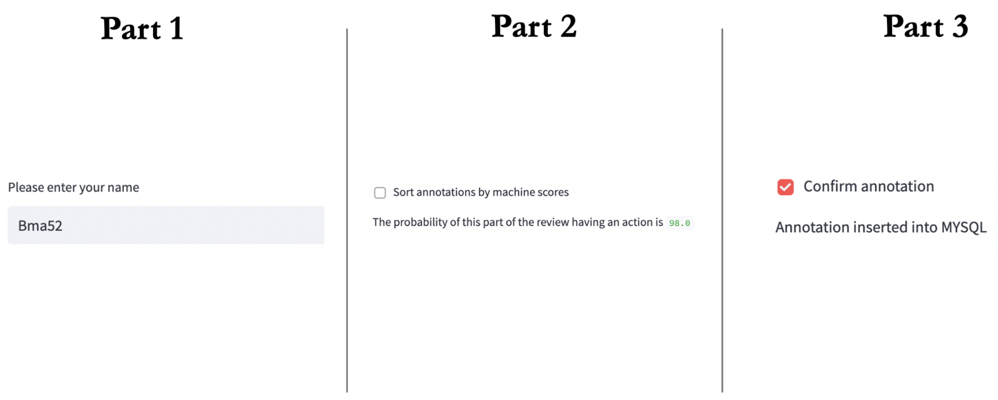
4. The Annotator can take breaks while annotating without any confusion between the reviews checked and those that are still without check up. This use case will be managed by loading only the annotations that weren’t checked by this particular annotator at every new session opened.

5. In case an annotation was predicted by the machine as informative (Have an action), and the review actually doesn’t reflect any action, the user has the option to select ‘No Action‘ from the new action select box. This ‘no Action‘ classified annotation is eliminated from the triplestore and only used for re-training the action/no-action model, refer to Part 1 in Figure 4.12.

6. In case the sentence has an action, but none of the actions from the select drop down menu fits the sentence. The user has the option to input a new action word by clicking on ‘Add a new action‘ option in the drop-down menu, that displays an input text field to be filled with a new action, refer to Part 2 in Figure 4.12.

[0]Link to ‘Annotators Page‘ page: https://bma52-reviews-actions-getnewreviews-lojlss.streamlit.app/AnnotatorPage

7. A note will appear as the annotator clicks on the ‘Confirm Annotation‘ checkbox, informing the user that this annotation is now stored in MySQL, refer to Part 3 in Figure 4.10.



Please enter your name

Bma52

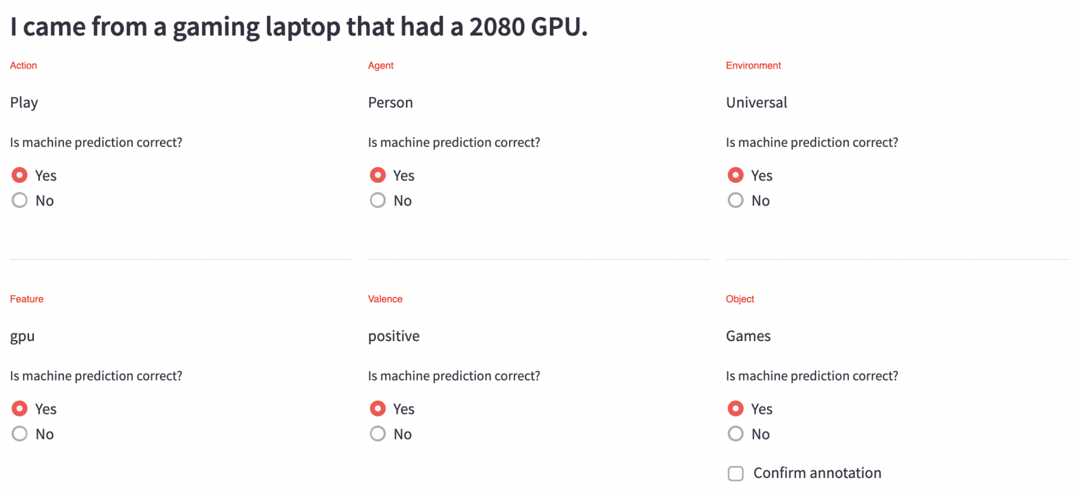
☐ Sort annotations by machine scores

The probability of this part of the review having an action is 98.0

☑ Confirm annotation

Annotation inserted into MYSQL

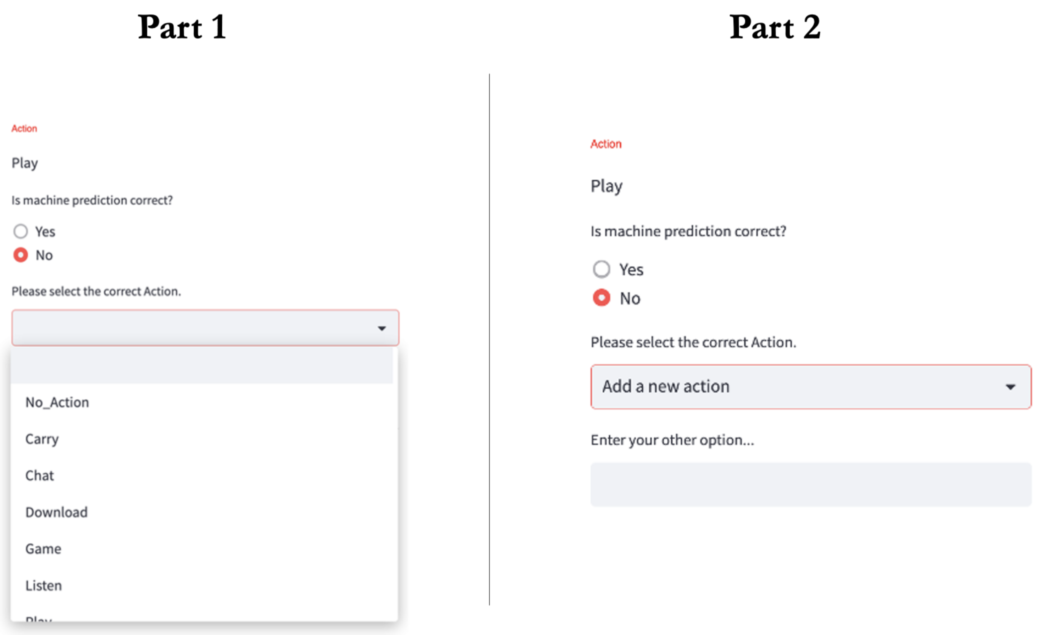
Figure 4.10: Annotator’s Page Features



I came from a gaming laptop that had a 2080 GPU.

Action	Agent	Environment
Play	Person	Universal
Is machine prediction correct? [✓ Yes] No	Is machine prediction correct? [✓ Yes] No	Is machine prediction correct? [✓ Yes] No
Feature	Valence	Object
gpu	positive	Games
Is machine prediction correct? [✓ Yes] No	Is machine prediction correct? [✓ Yes] No	Is machine prediction correct? [✓ Yes] No
		☐ Confirm annotation

Figure 4.11: The Annotator’s Page Displaying Each Review and its Annotations



Action

Play

Is machine prediction correct?

Yes [✓ No]

Please select the correct Action.

No_Action
Carry
Chat
Download
Game
Listen
Play

Action

Play

Is machine prediction correct?

Yes [✓ No]

Please select the correct Action.

Add a new action

Enter your other option...

Figure 4.12: No Action and New Action Features in Annotators Page

KG Triplestore

SPARQL is the standard query language and protocol for RDF databases [30]. It enables users to query information from triplestores or other data sources that can be mapped to RDF. Our project uses this query language to get and store data from and to an existing triplestore (KG Triplestore). Queries are wrapped up using the Sparql Wrapper python package. This query is initiated when the data engineer uses the data for training the ML models, noting that the initial data stored in this triplestore is the manual annotated data that was collected using the 'Reviews Annotator' chrome extension. An INSERT query is wrapped up also using the same package, and used when the annotator pushes the checked triplets to the triplestore again.

Sparql Wrapper python package

SPARQLWrapper 2.0.0 was used in implementing the connection between our python script and the sparql triplestore [29]. This is a wrapper around a SPARQL service that helps in creating the query URI and converts the result into manageable RDF format. This SPARQL query is placed in the admin page as he/she are responsible for pushing the approved checked annotations into sparql triplestore.

MySQL DBMS

MYSQL is an open-source relational database management system [31]. MYSQL in our project acts as an intermediate DBMS to temporarily store our data and retrieve it when needed. The database is formed of five tables; Product, Review, Annotation, Checked Annotation, Annotator, and ML models, refer to Figure 4.13. This storing stage will relief the load of noise inserted to the KG triplestore, as only checked and approved data from MYSQL will enter the triplestore.

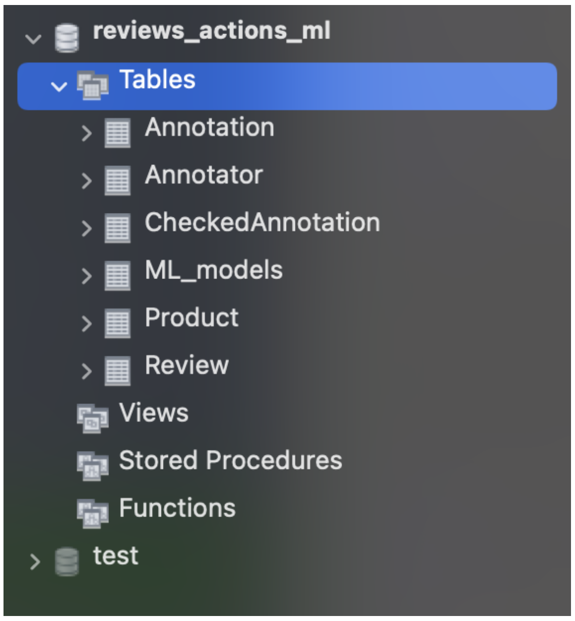


Figure 4.13: MySQL Database Tables

MySQL python connector

MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the python database [32]. Queries are executed to insert and to retrieve data to and from MYSQL. This connection is placed on both the admin and the annotator side, as a direct communication with the database has to be. Credentials were hidden and stored in a properties file in the GitHub repository for security reasons. A Properties() function from the JProps package in python was called to retrieve the configurations of the server [33].

Streamlit Deployment

Deploying platforms in the cloud is a must-go-through phase in every project. Our app was made using streamlit python package which is an open-source free package that helps turning python scripts into shareable web app. Streamlit's Community Cloud is an open and free platform for the community to deploy, discover, and share Streamlit apps and code with each other. The streamlit cloud is connected to a public GitHub repository where all models, scripts, requirements, and other related endpoints in our project are hosted. Deployment on a cloud is characterized by its Scalability, Accessibility, Mobility, and Easy real-time access of the app on the web.

4.3 Retraining Models Usability Study

Success of machine learning depends on obtaining the right quality data. The production, selection, and annotation of data is a human endeavor [8]. This is the reason why this study is based on the adaptation of machine learning models involving human-in-the-loop. This human intervention will be going throughout the annotators, who will be checking and editing machine results, using the automatic tool.

Model retraining adapts the model to the updated data to make more accurate predictions. This recurrent training is fundamental to ensure that the models are consistently predicting the most correct output, by providing an automated pipeline for collecting new data, predicting labels and checking those predicted output by an outside factor. This factor is the human intervention. The current and updated data represent the checked annotations by human annotators who edit any falsely predicted labels by the models. Models are retrained frequently on these human checked annotations appended to the initial training set. This incremental update of the models responds to changes that degrade the level of human intervention.

Model retraining experiment of the automatic framework was done over a period of one month by collecting new data and retraining the models. Human annotators used the tool to annotate new batch of reviews. Checked annotations are stored in MySQL to be able to retrain the models using the new updated data reviewed by the annotator. New versions of the models are also stored in MySQL to monitor their performance at every state.

4.3.1 *Experimental Protocol for Retraining Models*

This section goes through the retraining models experiment explaining how data was collected, split, and fed to models. Moreover, it goes through the process of how the models were evaluated to choose the best ones that will handle the predicting task, and how they were trained and tested on multiple data sets.

This experiment discusses multiple states of data collection and model retraining on distinct sets of data. A section in Chapter 5 emphasizes the training, validation, and testing data used at every state, the best performing models, and the results of the experiment explicitly.

Data Split

Initially models were trained on a set of data D(0) split into train, validation, and test. The models were optimized using hyper-parameter Grid Search technique to get the best parameters that should be used in each model [34]. Retraining the best model is based on the new checked annotations set appended to the initial training set. Every set of newly checked annotations are divided into train set of size 0.8 and a test set of size 0.2. The model is trained with bigger set of data at every state, while having multiple testing sets to score. For example, taking the D1 set of data in Figure 4.14, Test 0 is a fixed set that will always be scored to measure the improvement due to adding more data for training. Test 1, is from the newly added data, and scoring it indicates whether the new data is similar or not to the old one. Test 01 is a cumulatively growing set that gives indications about generalization of the system. Figure 4.14 displays the splitting of data process and how the training and testing sets are growing with time.

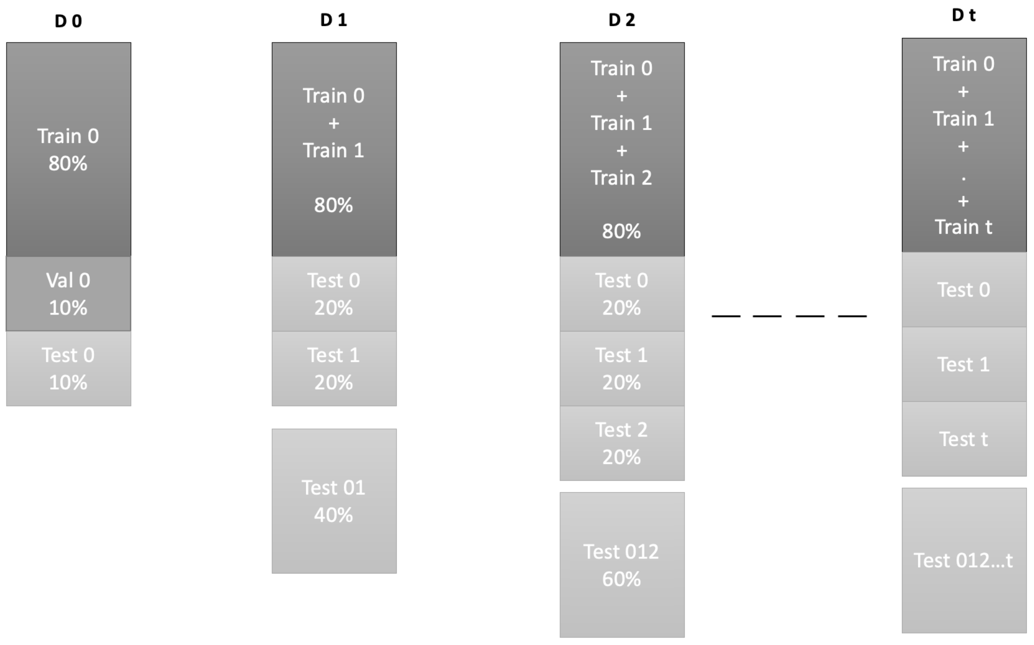


Figure 4.14: States of Splitting Data for Retraining Models

Retraining models on multiple sets

Choosing models based on a hyper-parameter optimization technique resulted with the best models in terms of performance and accuracy score. At the initial state, the hyper-parameter optimization experiment is done using data set D0, which is split into training, validation, and testing. The validation set is used to validate our model performance during training. This validation process helps us tune the model's hyper-parameters and configurations accordingly. Data is then collected by human annotators and split as per the explained method in Figure 4.14. Models are retrained using new training set at every state, while having a fixed testing set (ex. Test 0), a new testing set (ex. Test 1) and a growing testing set (ex. Test 01). The scores on multiple test sets is explained and displayed in Chapter 5. The purpose for having fixed testing sets at all states is to provide an unbiased final model performance metric in terms of accuracy, precision, and f1-score.

Chapter 5
Results and Discussion

5.1 Prototype's Implementation and Results

The Machine Learning Models

Informative and Non-Informative classification layer

Informative and Non-Informative classification layer preceded the semantic prediction phase to reduce the number of noise that can disrupt the performance of the models. Since the initial data set for training included only the sentences of the review having an action while those with no action didn't exist. In our opinion, the 'No Action' sentences will add value to the classification model, since a negative class can be a good replacement of the falsely predicted actions in a sentence. To tag each sentence if it contains an action or not, we applied a split by sentence algorithm on each review text. For example, if we have this review text, it is classified as follows:

Review Text:

This laptop is everything you need! The graphics are beautiful and it is very user friendly. I can run all of my favorite games and more on this!

Split by Sentence and Classify

This laptop is everything you need! - No Action Found

The graphics are beautiful and it is very user friendly. - No Action Found

I can run all of my favorite games and more on this! - Action Exist

Our initial training set contained only the annotations or the sentences having an action. To classify all the sentences of the reviews, we applied some preprocessing.

A split by sentence to the review texts was done, then these sentences were matched with the annotated ones in the training set, thus resulting in the following:

1. If the sentence is found within the initial training set, it is informative since it was tagged with an action.
2. If the sentence can't be found in the training set then this was probably a non annotated sentence, thus non-informative annotation.

This layer reproduced a column 'Action Flag' for our data set to distinguish the annotations having 'Action Exist' and others having 'No Action Found' flags. The Splitting and Tagging layer for every sentence in the review text resulted with an imbalanced data set for training the models, as the majority class will be the 'No Action' one. To fix this issue, we under-sample our data following the 'Majority' sampling strategy which trims the class with highest number of occurrences to have the same number of records like the minority class. The distribution of classes before and after sampling can be referred to Figures 5.1 and 5.2.

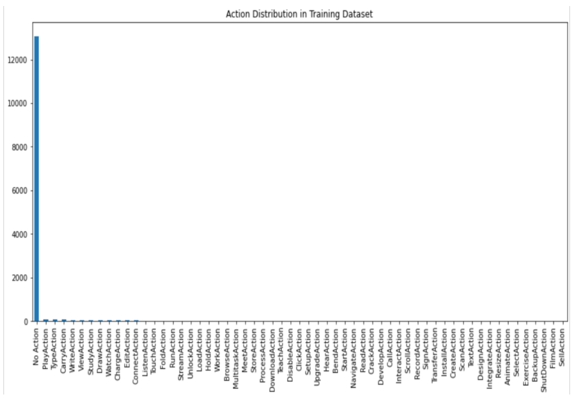


Figure 5.1: Before Under-sampling

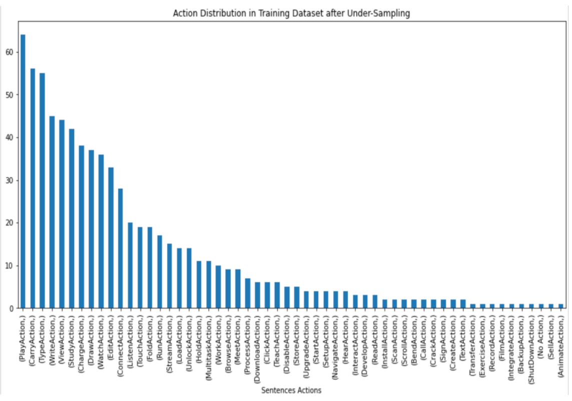


Figure 5.2: After Under-sampling

Four different classification models were tested at this level, to proceed with the best model in terms of accuracy; the Random Forest model having 99 percent on training and 91 percent on validation sets. Figures 5.3 and 5.4 show the ROC curves of the different classifiers trained and tested at this stage. The ROC cure is a sensitivity plot between true positive and false positive rates, while the area below the curve measures the accuracy. Analyzing the below ROC curve comparing four different classifiers, it is seen that the best classifier having the highest Area under the curve AUC, is the Random forest also reflected in Table 5.2.

Accuracy is defined as the percentage of correct predictions for the test data. It is calculated based on the below formula.

$accuracy = correct\ predictions/all\ predictions$

Entity	Number of Classes	Total Set size	Train Size	Valid Size	Test Size
Action Flag	2	3389	2711	339	339

Table 5.1: Action Flag Entity Model Information

Action/No-Action Entity

Model	Training Set	Validation Set
Support Vector Machine	97	91.6
Linear Regression	87	90
XG Boost	86	89
Random Forest	99	91.4

Table 5.2: Accuracy Results for Action/No-Action Entity

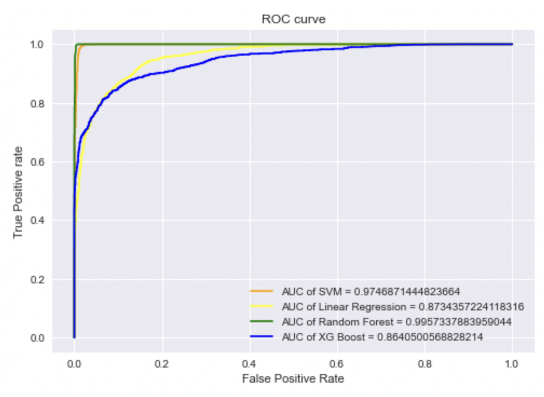


Figure 5.3: ROC curve on Training Set

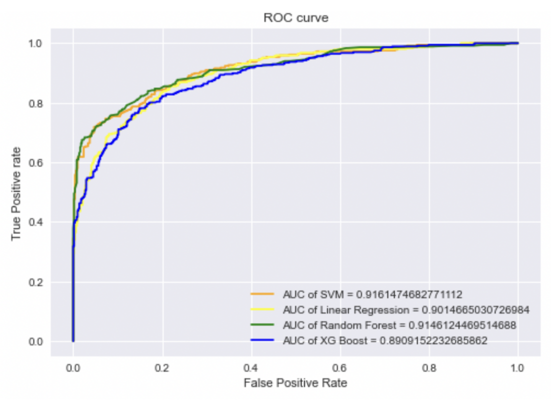


Figure 5.4: ROC curve on Testing Set

Multi-label Classification Experiment

Action prediction is the main focus in this thesis, in which it is the task of assigning predefined classes of actions that may be found in product reviews. After inspecting a wide range of reviews, we observed having multiple actions in the same sentence as represented in Figure 5.5

An actual example from our training set.

The studio mode is great for drawing and writing things.

A well performing predicting classifier should be able to assign this sentence two instances of actions, Draw and Write. This is an example of multi-label classification

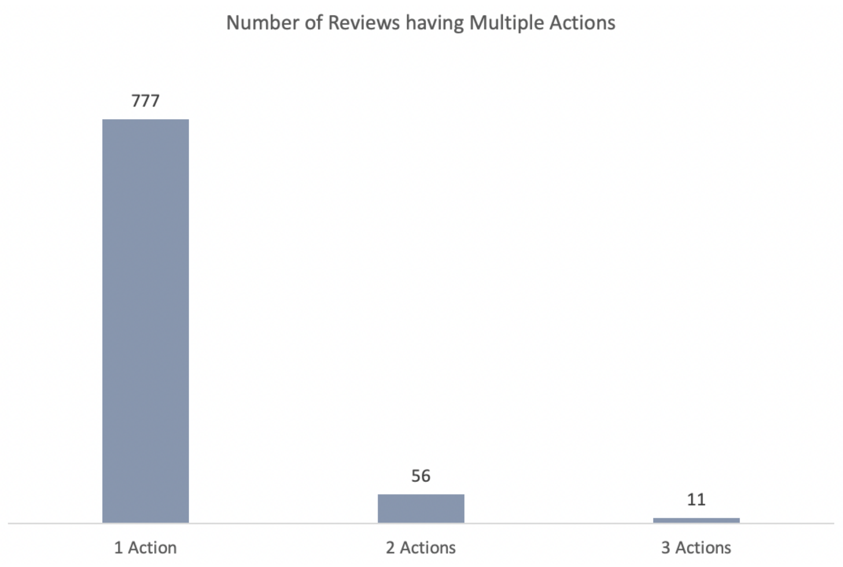


Figure 5.5: Action Distribution in Reviews

that handles this multi-labeling of textual input. We tested multiple algorithms that are used usually for multi-label classification problems such as binary relevance, label power set, and the classifiers chain.

The evaluation measures for single-label are usually different than for multi-label. In multi-label classification, a miss-classified result is no longer a hard wrong or right. The prediction in multi-label is usually an array of binary results that reflects the existence of multiple labels for the same text input. A prediction containing a subset of the actual classes should be considered a better choice than a prediction that contains none of them. For example, predicting Work and Stream actions for the below annotation is better than predicting no labels at all or only one out of 3, even when an action is missing from the prediction array resulted. (The output should be work, read, and stream actions)

The studio mode is great for working, reading, and streaming movies.

To calculate the metrics of a multi-label classifier we usually consider two methods, Micro-averaging and Macro-averaging (Label based measures). In micro-averaging, you sum up the individual true positives, false positives, and false negatives of the system for different sets of the predictions and apply the average of the equations to get the f1-score. In macro-averaging, we just take the average of the precision and recall of the system on different sets. Hamming-Loss is the fraction of labels that are incorrectly predicted. For example, the fraction of the wrong labels to the total number of labels in the prediction array. Finally, accuracy in this classification problem is called subset accuracy or the exact match ratio. Subset accuracy is the most strict metric, indicating the percentage of samples that have all its labels classified correctly. The disadvantage of this metric, is that it ignores those partially correct matches.

The results of multi-label experiment were not promising for a machine learning model to predict action entities in the framework, the reason why we will consider predicting the action entity in the framework using a multi-class classification model instead of multi-label model for better results. We cannot directly convert multi-label classification to multi-class classification problem. A simplistic approach to handle this situation is called binary relevance method, where you train one binary classifier for each label. The group of annotations having multiple actions within the same sentence will be handled through this approach. Multi-label aspect will be a window for future work where more training data is available, thus having a better strong learning base for the multi-label model. Based on the above, action is predicted like other labels with a multi-class classification model, the results of the experiments and classifiers trained on action label are displayed in Table 5.4.

Multi-Class Prediction Models

Traditional classifiers handle the prediction of a group of semantics in the project. These classifiers work in a multi-class classification mode as the models have multiple classes as predicted labels. These semantics are action, agent, environment, valence, and object, in addition to the first layer of classifying the data as informative or non-informative sentences. An aggregation level was done to downgrade the number of classes in each model for better model performance. Classifiers upon having more classes, will be expected to have lower accuracy that should be boosted with more training data in the future. Results for classifiers performance are reflected in multiple tables below for each entity. The best performing model was chosen based on the accuracy score. The initial set of data that we worked with is the manual annotated batch of reviews using the manual annotation tool. The total size of annotation of the initial data set is 720 annotations or sentences referencing to an action. The nature of our initial data is imbalanced, thus always having one or two major classes that suppress the other ones. A distribution of the classes for the action entity is displayed in Figure 5.5 along with it's models performance.

The ultimate goal of any machine learning model is to learn from examples, and generalize some degree of knowledge regarding the task that is being performed. These machines learn from the initial manual annotated data set and continue to learn successively from the checked annotations reviewed by human annotators. This human involvement helps build a consolidated training set, that is learned by the model to reach a trustful algorithm for predicting semantics.

The following are the experiments done on each entity or action knowledge, and the results from the different classifiers highlighting the best in performance. Also, an error analysis is done on the best classifier chosen for each label to inspect the wrong occurrences of predictions that are taking place, and to examine if any error can be fixed through data manipulation.

Action Entity

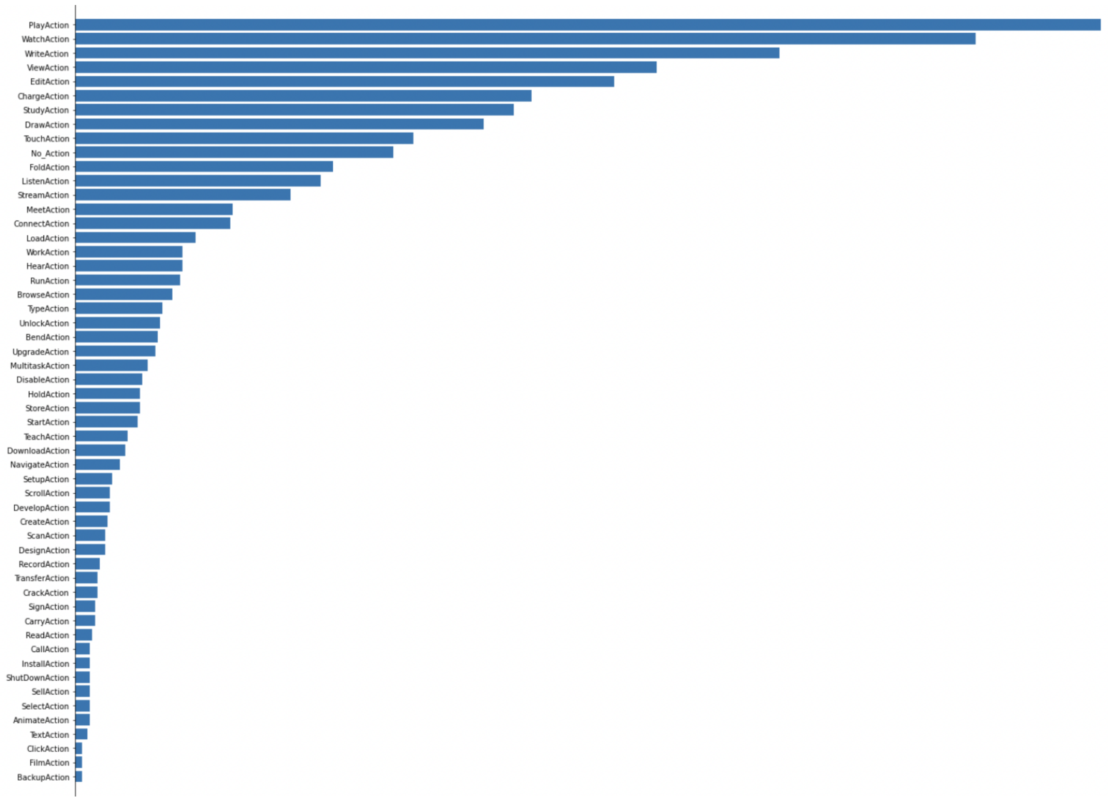


Figure 5.6: The Imbalance Distribution of Action Entity

Entity	Number of Classes	Total Set size	Train Size	Valid Size	Test Size
Action	59	720	576	72	72

Table 5.3: Action Entity Model Information

Model	Training Set	Validation Set
Support Vector Machine	87	48
Linear Regression	71	48
XG Boost	92	50
Random Forest	93	50

Table 5.4: Accuracy Results for Action Entity

Error Analysis of Action Model

Analyzing the errors reproduced in the action model, we were able to detect the highest wrong occurrences of some actions like having Watch action predicted as Draw 4 times, Store 3 times, Edit 1 time, Touch 1 time, and play 1 time. 3 wrong occurrences of Store action predicted as Carry. The error margin can be reduced by training the models with more data. Refer to Figure 5.7

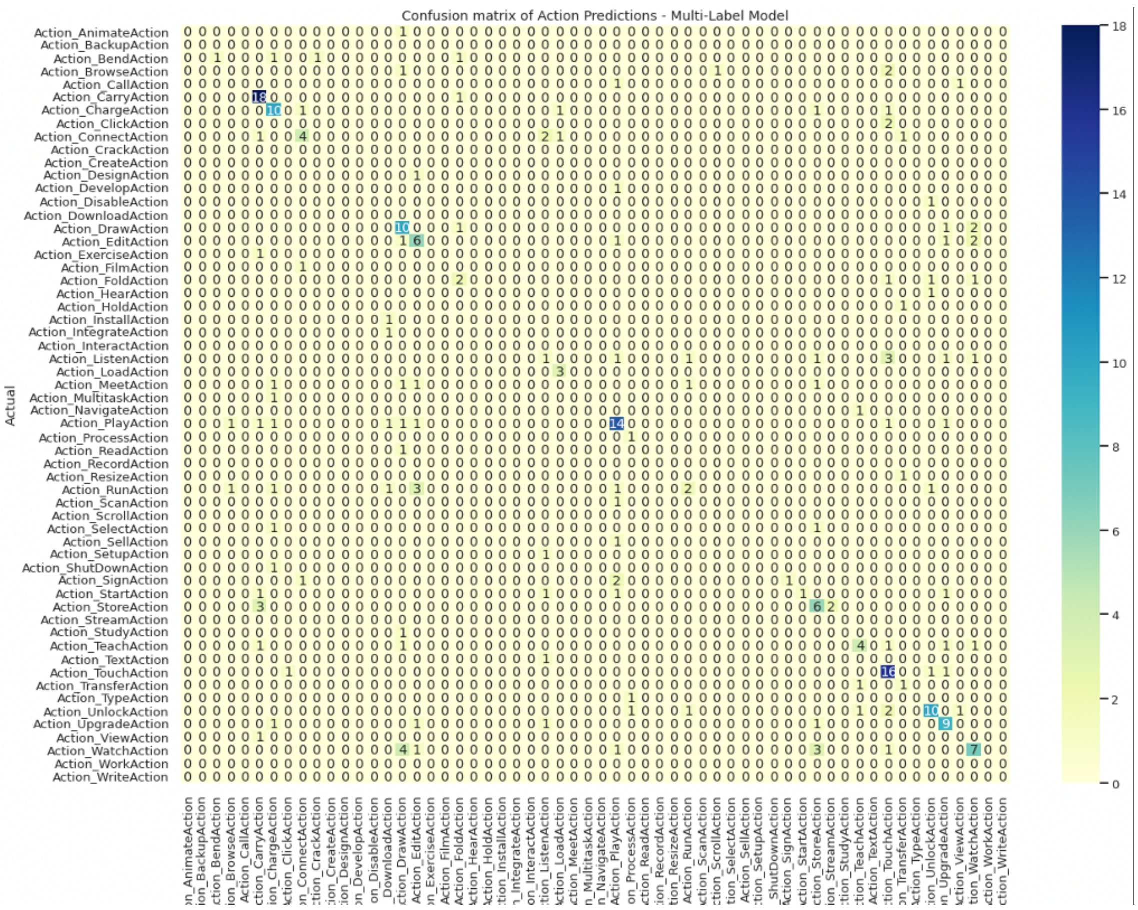


Figure 5.7: Error Analysis for Action Classification Model

The high accuracy scores in the training set is due to the imbalance data that is being worked on. The more data is collected, the more instances for the minor classes can be found, thus re-sampling the dataset to perform in a better way in future retrains. The top two major classes are Play and Watch actions. Instead of dropping major classes from the dataset, which is one of the solutions to resolve imbalance dataset problem, we are working on collecting more instances of the minor classes of action entity to reduce the gap between classes. This method preserves instances of the all classes since no class should overcome others in such classification problem.

Agent Entity

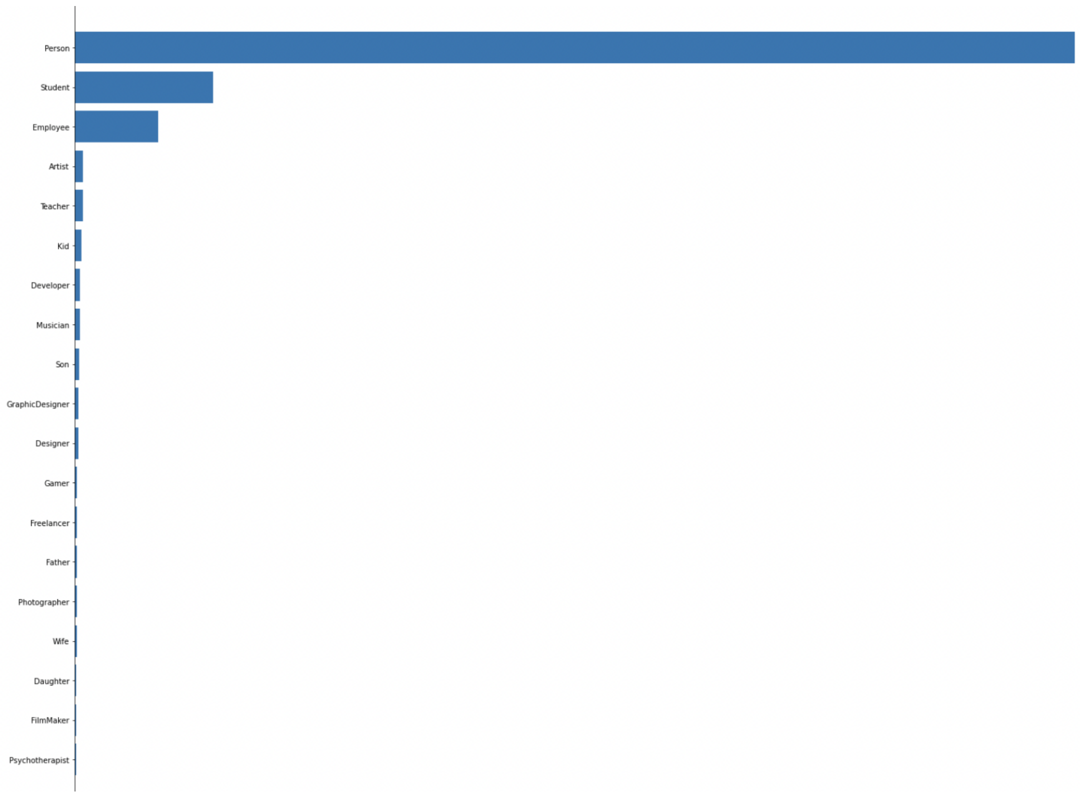


Figure 5.8: The Imbalance Distribution of Agent Entity

Entity	Number of Classes	Total Set size	Train Size	Valid Size	Test Size
Agent	25	720	576	72	72

Table 5.5: Agent Entity Model Information

Two classifiers performed well in agent prediction: the XGBoost and the Random Forest. The XGBoost scored better in validation. The data set of agent label is highly imbalanced having the ‘Person‘ class as a major one, since all the annotations that do not refer to a clear known agent, we refer them to a ‘Person‘ type of agent.

Model	Training Set	Validation Set
Support Vector Machine	89	80
Linear Regression	81	79
XG Boost	95	82
Random Forest	96	80

Table 5.6: Accuracy Results for Agent Entity

Error Analysis of Agent Model

Agent Error condensation was 17 occurrences of Employee agent falsely predicted as Person. Other errors appeared normal as 1 or 2 scattered occurrences or miss-classified agents. Collecting more data to increase the sample size of every category in the agent class enhances the performance of the model, and reduces the margin error and the confusion between classes upon predicting. This error analysis is monitored when training the models recurrently with new reviews and checked annotations. Refer to Figure 5.9.

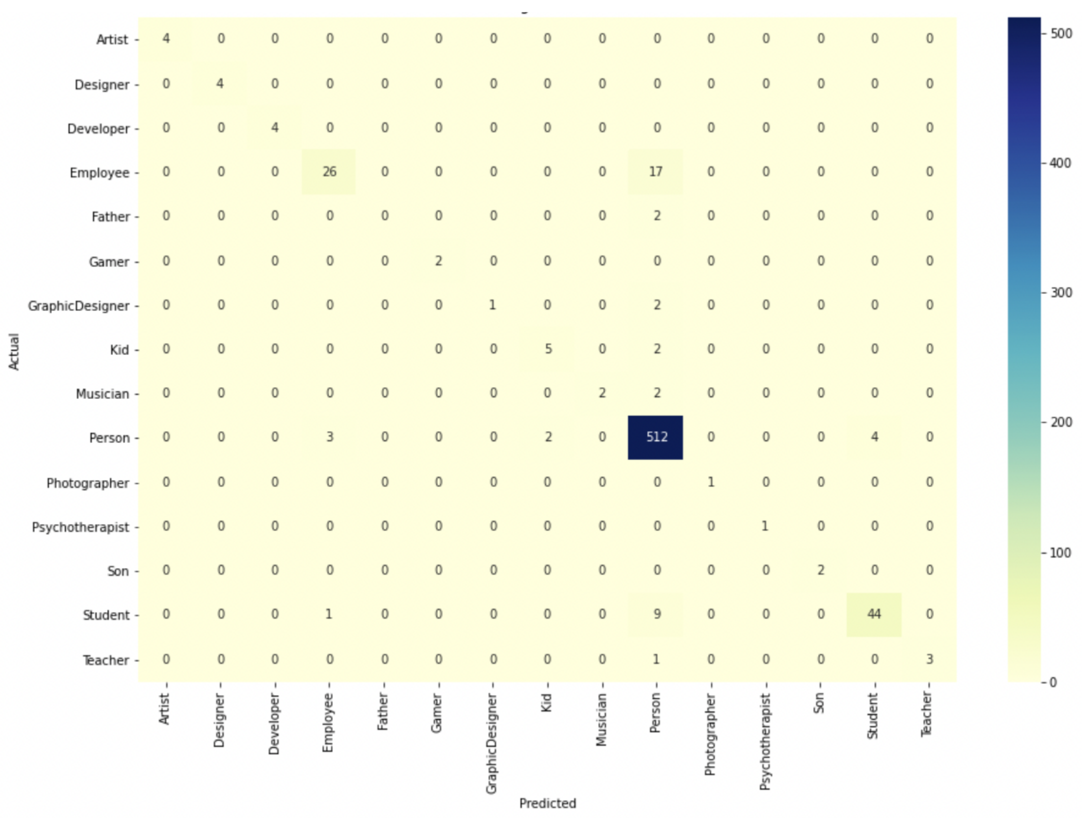


Figure 5.9: Error Analysis for Agent Classification Model

Environment Entity

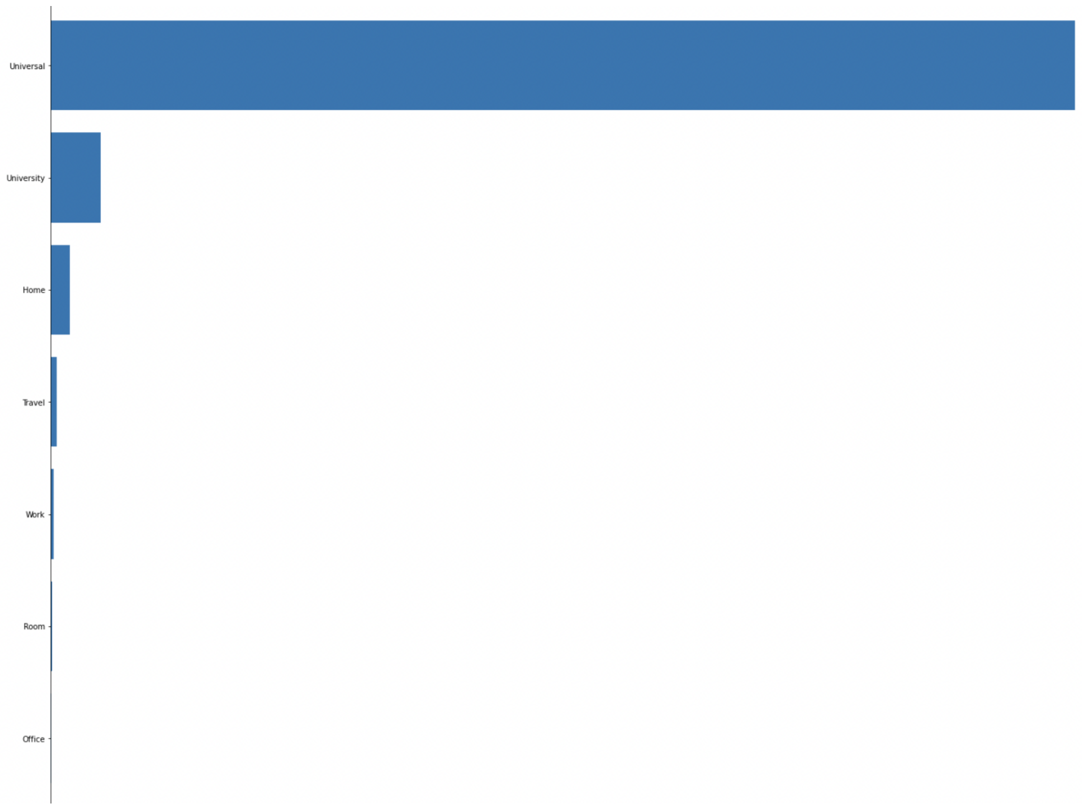


Figure 5.10: The Imbalance Distribution of Environment Entity

Entity	Number of Classes	Total Set size	Train Size	Valid Size	Test Size
Environment	9	720	576	72	72

Table 5.7: Environment Entity Model Information

Environment entity major class is 'Universal', refering to an ambiguous and unclear environment. Accuracy in the environment model is also very high as a result of the imbalanced dataset we have.

Model	Training Set	Validation Set
Support Vector Machine	92	95
Linear Regression	91	95
XG Boost	97	96
Random Forest	97	96

Table 5.8: Accuracy Results for Environment Entity

Error Analysis of Environment Model

Miss-classification of environment mainly occurred between the actual class ‘Home‘ and the predicted ‘Universal‘, where 11 occurrences took place. The others are normal false predicted classes. Refer to Figure 5.11.

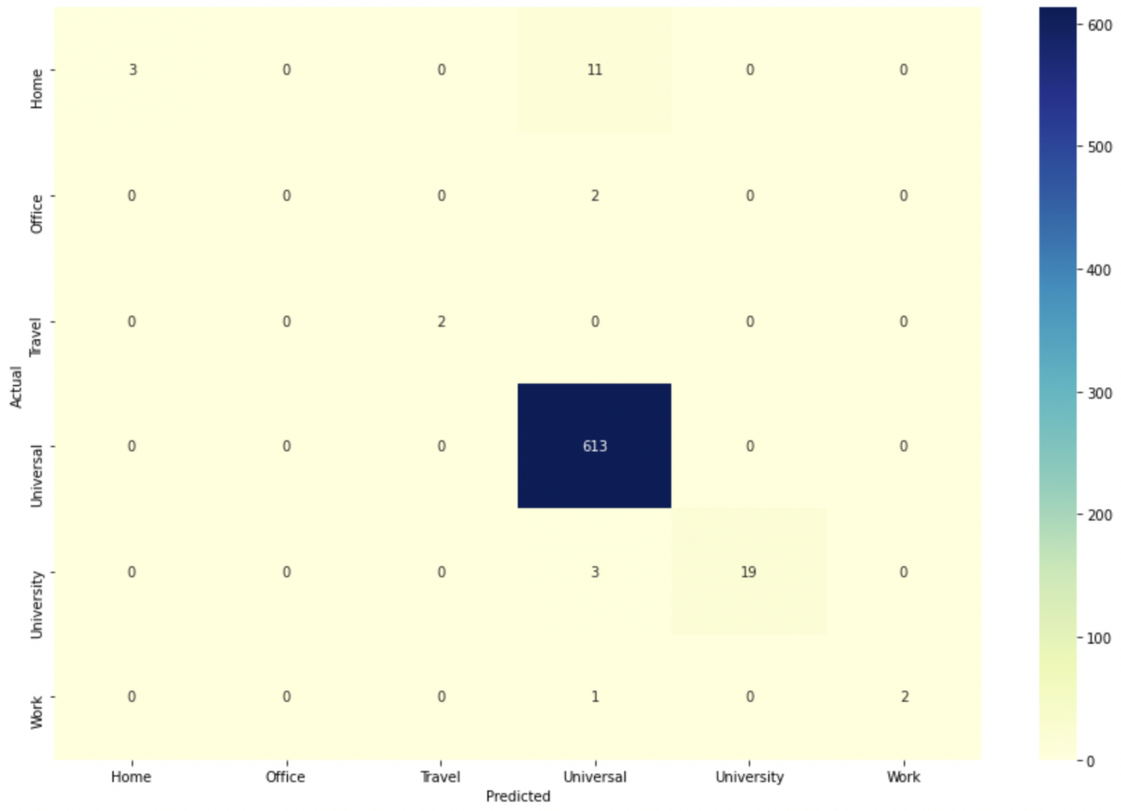


Figure 5.11: Error Analysis for Environment Classification Model

Valence Entity

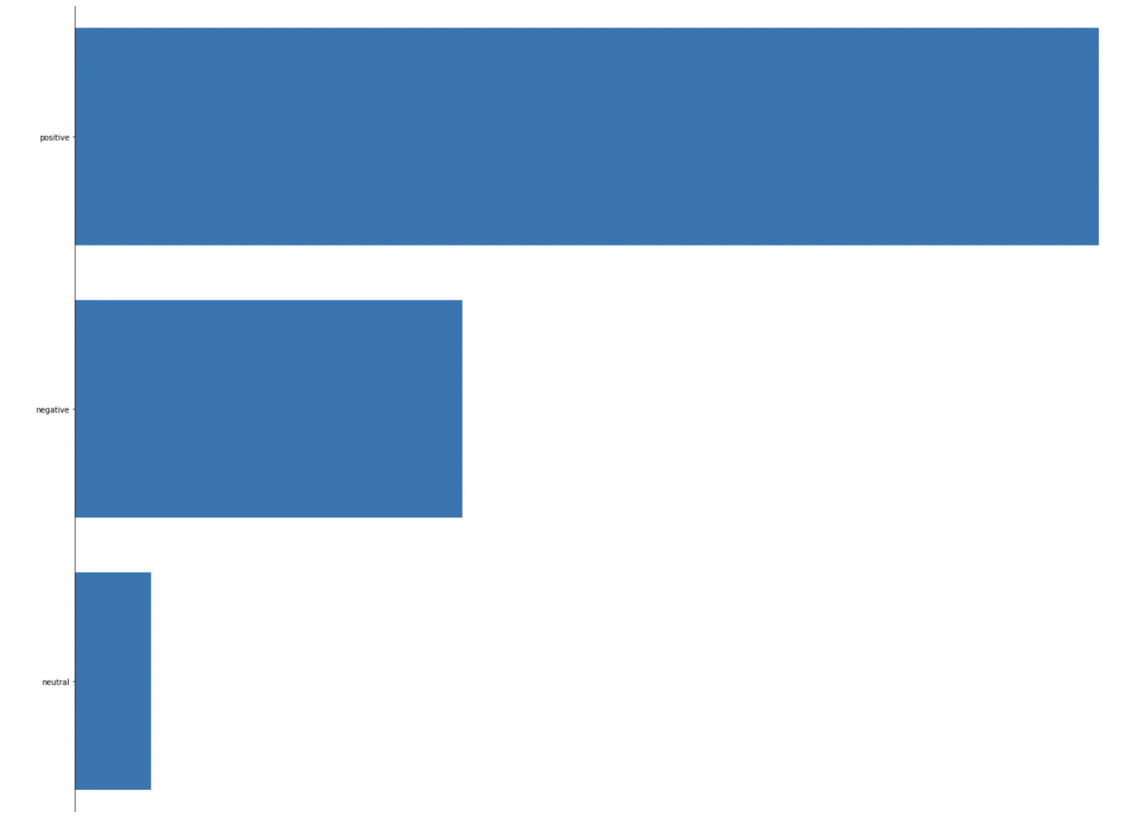


Figure 5.12: The Imbalance Distribution of Valence Entity

Entity	Number of Classes	Total Set size	Train Size	Valid Size	Test Size
Valence	3	720	576	72	72

Table 5.9: Valence Entity Model Information

Model	Training Set	Validation Set
Support Vector Machine	92	80
Linear Regression	90	82
XG Boost	92	79
Random Forest	94	77

Table 5.10: Accuracy Results for Valence Entity

Error Analysis of Valence Model

Our valence classification model had 36 false ‘Positive‘ predicted occurrences of the actual ‘Negative‘ class and 10 occurrences for the actual ‘Neutral‘ class. Training the model with more data will leverage its performance and reduce the error margin. Refer to Figure 5.13.

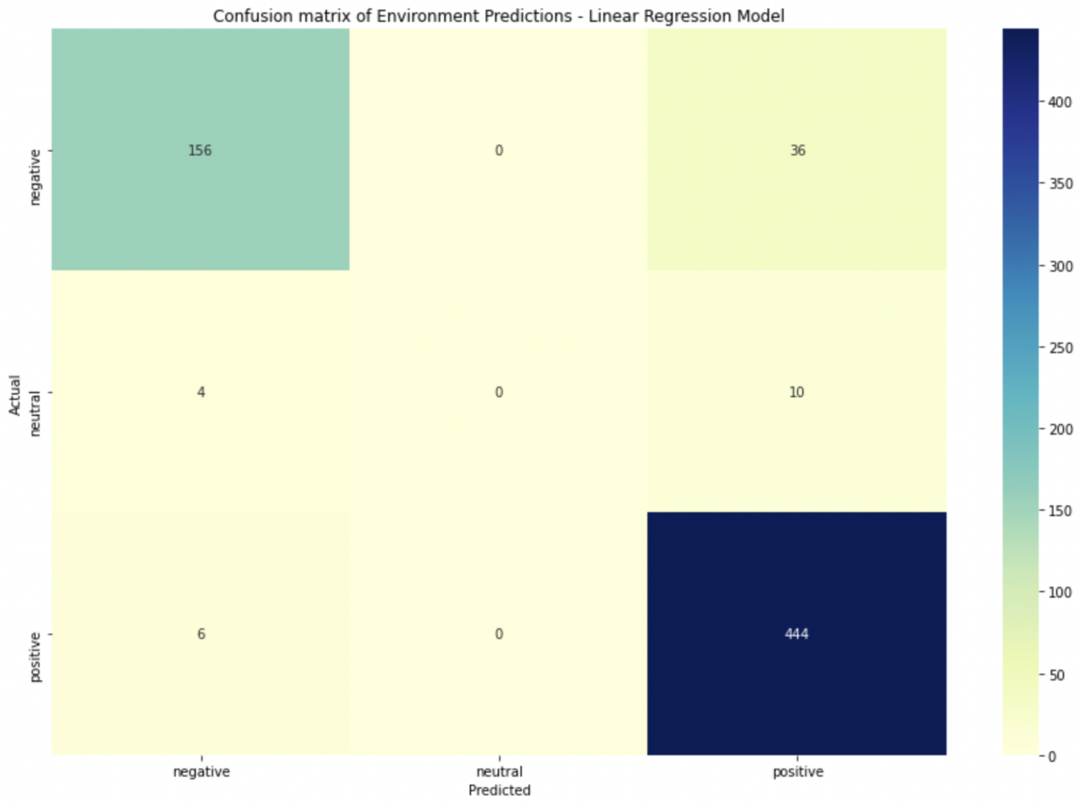


Figure 5.13: Error analysis for valence classification model

Object Entity

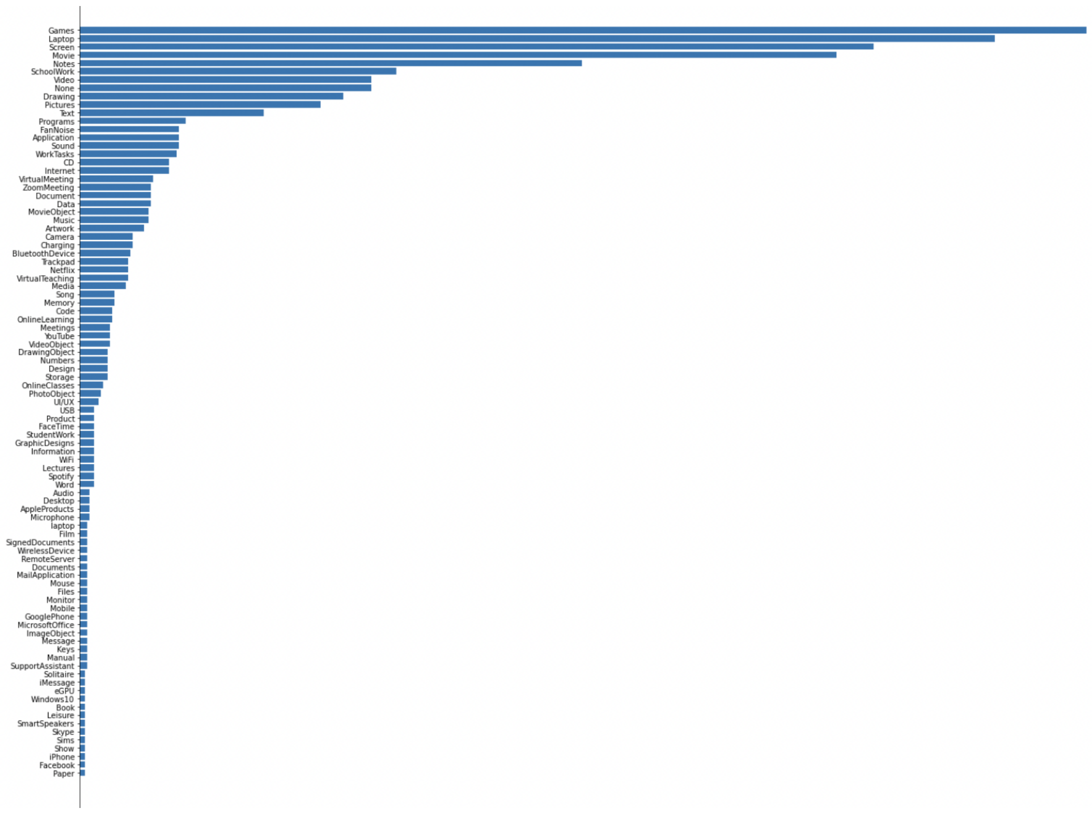


Figure 5.14: The Imbalance Distribution of Object Entity

Entity	Number of Classes	Total Set size	Train Size	Valid Size	Test Size
Object	70	720	576	72	72

Table 5.11: Object Entity Model Information

Model	Training Set	Validation Set
Support Vector Machine	75	72
Linear Regression	67	48
XG Boost	67	50
Random Forest	71	48

Table 5.12: Accuracy Results for Object Entity

Error Analysis of Object Model

Object has 70 classes to predict in the multi-class model, which will lead to many error occurrences in data. Training the model with more reviews in the future will increase performance and reduce the error margin in the object model. Figure 5.15 shows the error analysis done for inspecting object entity miss-classified occurrences. 10 actual instances of 'Laptop' were falsely predicted as 'School Work'.

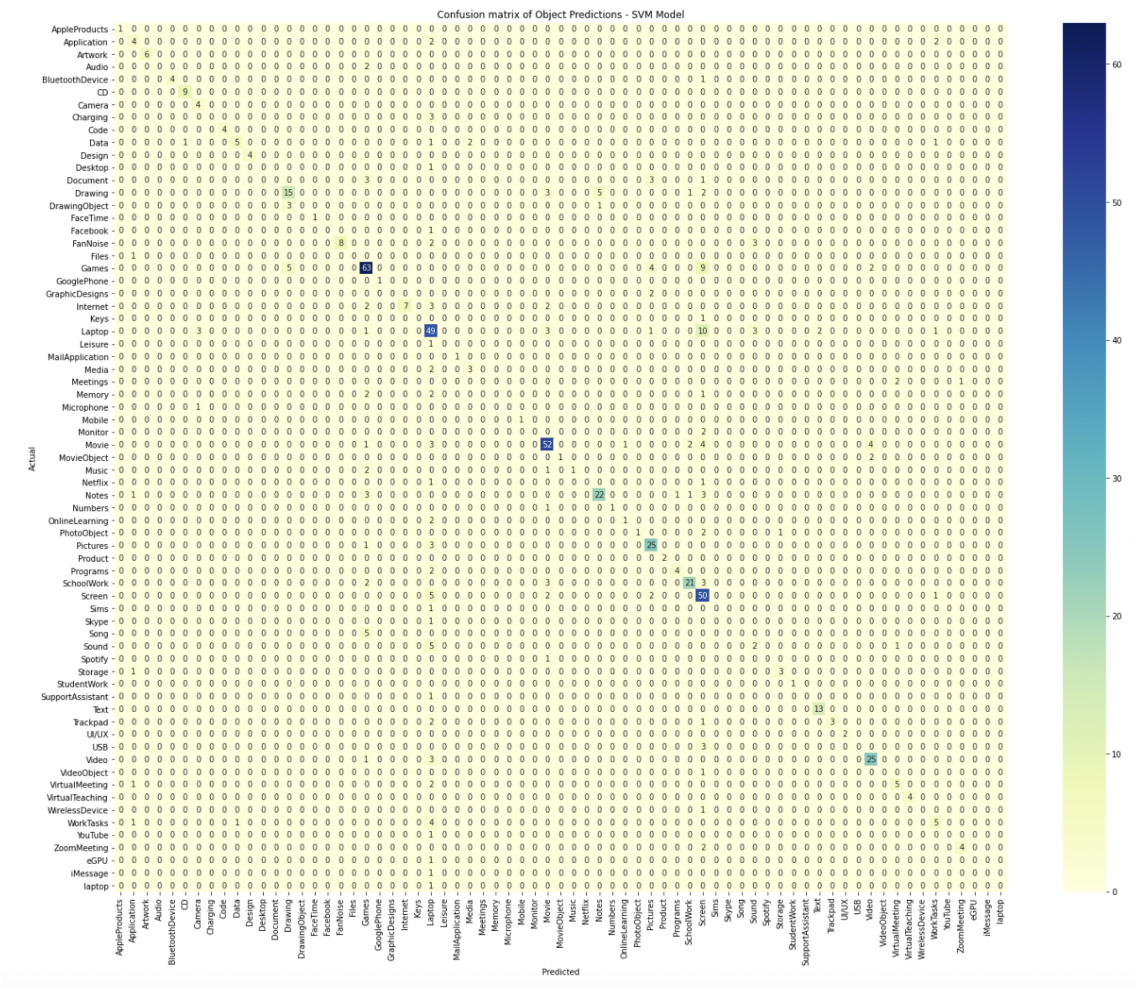


Figure 5.15: Error Analysis for Object Classification Model

5.2 KG Construction from Predicted Entities

The five predicted entities explained before are used to construct the knowledge graph for each annotation of our datasets. The automatic construction of KG make use of existing knowledge and enables the verification of new entities to be formed. Model performance is highly correlated with KG rules, as having accurate predictions reduces the probability of having badly designed knowledge graphs. Thus, the consistency of KGs rely on model performance. Enriching features (in our case, features are product reviews) of our models to optimize its performance, expands the knowledge data that constructs an ontology. New entities and new relationships will be introduced to our existing ones.

5.3 Models Parameter Optimization Phase

The models in this section are parameterized so that their behavior can be tuned and optimized. A model parameter is a configuration variable that is internal to the model, and whose value is estimated from the given data. A combination of parameters for each model is chosen and applied within a grid search cross validation in order to choose the best combination for best performance. Three models will be tested for all 6 classifications; action/no-action, action label, agent label, environment label, valence label, object label. The models have a set of parameters with multiple values validated in a grid search. The Grid search has 5 cross validations to select the best parameters estimated for each model.

Classifier	Parameter 1	Parameter 2	Parameter 3
Support Vector Machine	C	gamma:	kernel
XG Boost	n-estimators	max-depth	learning-rate
Random Forest	n-estimators	max-depth	-

Table 5.13: Parameters Set for each Classifier

Support Vector Machine Definitions of Parameters

The C parameter in SVM is Penalty parameter of the error term. It is the degree of correct classification or optimization that the classifier has to meet. The gamma parameter defines how far the influence of a single training example reaches, with high values meaning ‘close‘ and low values meaning ‘far‘. The kernel parameters are used as a tuning parameter to improve the classification accuracy. There are mainly four different types of kernels (Linear, Polynomial, RBF, and Sigmoid) used in SVM classifier.

Random Forest Definitions of Parameters

The n-estimators parameter in the Random Forest specifies the number of trees that is constructed in the forest of the model. The max-depth parameter in the Random Forest defines the longest path between the root node and the leaf node.

XG Boost Definitions of Parameters

The n-estimators parameter in the XGBoost classifier is the number of runs XGBoost will try to learn. The max-depth parameter in the XGBoost defines the longest path between the root node and the leaf node. The learning rate makes the boosting process more or less conservative. It is the shrinkage that is made in every step.

5.3.1 *Results of fine-tuning the models through hyper-parameter optimization*

Choosing the optimal hyper-parameters for every model boost its performance and correctly map the input feature (review text in our case) to the labels or targets (action knowledge entities in our case), for the purpose of achieving some form of intelligence in classification. It is listed above the group of parameters that are used in Grid Search CV technique for every model, thus giving results for the best parameters, displayed in Table 5.14. Hyper-parameter optimization phase changed the selection of some models as they performed better than default models when setting parameters. For example, The best action/no-action model before parameter optimization was the Random Forest having a testing accuracy of 91, while after setting best parameters, the Support vector machine performed better having a testing accuracy 98.

Model	Best Model	Score
Action Flag	SVM(C = 1, gamma = 1, kernel = ‘poly‘)	accuracy=98
Action	SVM(C = 100, gamma = 0.1, kernel = ‘rbf‘)	accuracy=97
Agent	SVM(C = 100, gamma = 0.1, kernel = ‘rbf‘)	accuracy=96
Environment	SVM(C = 10, gamma = 1, kernel = ‘rbf‘)	accuracy=99
Valence	SVM(C = 10, gamma = 1, kernel = ‘sigmoid‘)	accuracy=94
Object	SVM(C = 100, gamma = 0.01, kernel = ‘rbf‘)	accuracy=72

Table 5.14: Best Parameters for each Model

5.4 Retraining Models Stages

Evaluation study takes place in this research for validating the models of the automatic tool. This study is done over one month period of time. A human annotator works on correcting the predicted labels of a batch of new reviews using the automatic tool. These collected and checked data will be added to the initial training set to retrain the models and monitor their accuracy. Within this one month evaluation study, the models are retrained two times with updated sets of data. First round of model retraining was done after collecting 200 new annotations by human annotators. The accuracy scores resulted after this round were not satisfying, and didn't reveal any increase in performance. So, we decided to continue the collection process to reach 400 new annotations and apply another round of model retraining. Round two of retraining unfolded an increase in accuracy scores of ML models, thus a leverage in model performance. Tables 5.15 and 5.16 reflect this increase in model accuracy scores. The reinforcement of data using human intervention optimizes the performance of the models to reach the maximum accuracy needed for trusting the framework in the future. This evaluation study proved the process of adapting ML models with new data through time. Some real examples of truly predicted and falsely predicted output from the machine models are mentioned. These examples give a realistic overview of the performance of the models.

Initial State of the models

Number of annotations D0: 720

The Accuracy of the models initially are displayed in Table 5.15

Some examples of errors falsely predicted by machine

Great Choice! : predicted with a Carry action

It is everything i needed : predicted with a Carry action

Works like a charm : predicted with a Play action

State 1: Experiment done after collecting 400 new annotations

Retraining the models with 200 new annotations didn't show an increase in performance, so annotators kept on collecting more annotations to reach a batch of 400 new checked records. This group of new data is appended to the initial training set to retrain the models with bigger set of data. We tested the updated version of model on the old testing set to see how accuracy scores improved when adding more training data.

Number of annotations D1: 720 + 400 = 1120

The Accuracy of models retrained in this state are displayed in Table 5.16

Real examples of correctly predicted annotations by machine after human checking route.

I came from a gaming laptop that had a 2080 GPU : predicted with a Play action

Love this laptop : predicted with a No action

Great Choice! : predicted with No action

It is everything i needed : predicted with No action

Works like a charm : predicted with a Work action

5.4.1 ***Results of retraining the best models on multiple data sets***

Experiment using data set D0

Table 5.15 records the initial accuracy results of our models for all labels we need to predict. The 100 percent accuracy doesn't mean that the model is a good one with no false predictions resulting from it, instead, it means over-fitting. Over-fitting occurs when the machine tries to cover all the data points available in the data set when having unbalanced data. The initial set of data manually annotated has only the sentences having 'Action Exist' flag, thus having one class that results with 100 percent accuracy of the model. Starting with over-fitted data, the model can be improved with new data having negative classes (sentences classified having no action) that re-samples our dataset in order to reach optimal level of accuracy.

Nine-tees of accuracy in some models might also mean over-fitted data, or unbalanced set of data. The initial manual annotated set of data had a huge gap in the classes of every model. A major class in each model has invaded the data-set and resulted with this highly skewed models. To make the data set balanced there are two ways to do so :

1. Under-sampling: Remove samples from over-represented classes. Usually this is used when having huge dataset, unlike our case as we started with a small data set to be developed with time through human participation.
2. Over-sampling: Add more samples from under-represented classes. This is used when having small training set. Human intervention in the framework is able to accumulate training data for various classes with time.

Model	Training 0	Validation 0	Testing 0
Action Flag model	100	98	97
Action model	99	98.2	98.5
Agent model	96.2	93.8	95.6
Environment model	98.9	97.3	98.8
Valence model	95.5	94.4	95.3
Object model	79	67.1	71.9

Table 5.15: Model's Results on D0 Dataset

Experiment using data set D1

In this stage, results begin to unfold. The models are now trained on new training set having a total of 1120 annotations. Training scores are still high, but looking at the results of 'Testing 0' of data in Table 5.16, we observe an increase in model performance when predicting labels for this testing set, compared to the initial state results. This increase can build premises on the experiment of retraining data, that is being done, and give evidence to our main contribution of adapting the models through human integration within an automatic framework. This increase happened after adding 400 new annotations to the training set.

Collecting data recurrently and in large amounts is very important to create the balanced multi-class set of data that can result with an optimal learning curve for our machines. Our training set should include multiple records on almost every class to predict, thus giving the model multiple instances to learn for each class. The more data human annotators can collect, the more instances will be given to the models, thus enforcing better learning.

Model	Training 1	Testing 0	Testing 1	Testing 01
Action Flag model	89.3	96.1	39.6	89.7
Action model	99	99.2	94.3	90.6
Agent model	99.4	99.2	98.1	98.1
Environment model	99.6	99.2	98.1	99
Valence model	97.3	96.7	90.5	94.9
Object model	96.8	95.1	96.2	90

Table 5.16: Model's Results on D1 Dataset

5.5 Impact of human-in-the-loop technique on the framework

The main aspect of our thesis project is to validate the proposed solution of human integration in an automatic framework for annotating customer reviews. The project involved human action for collecting data using the automatic tool of the framework, then retraining the existing models with the human checked data frequently. The evaluation experiment done gave evidence on ML model adaptation process through human intervention in the automatic framework. Results in Tables 5.15 and 5.16 reflected this adaptation of ML models.

As per the results, it is observed that the datasets, when growing and increasing in number of training records, starts to give more accurate and logical results than the initial state. Collecting data through human integration improved the training sets that were weak and small, and caused multiple errors like over-fitting in one of our initial models. This increase started to appear in stage 1 of retraining the models with additional 400 newly checked annotations. Focusing on 'Testing 0', it is observed that there is a shift in accuracy between stage 0 using 'Training 0' and stage 1 trained with a bigger set of data 'Training 1'. The improvement happens when the gap between classes of each entity starts to minimize. When a major class invades the dataset, the model will consider it as a positive occurrence versus all other classes as negative occurrences. Collecting vast amounts of reviews and annotating them will increase the number of occurrences of all classes thus retraining the model on with a stronger base of data.

This automatic framework works on machine adaptive learning, where human domain experts have control over the learning process. Then upon trusting our models after multiple retraining rounds, the type of learning will be reduced to interactive machine learning where a close interaction will occur between users and the machine. This is one of our main objectives of this thesis, to reduce the human intervention level with time when models are trusted enough to predict and give accurate action knowledge tags to build correct knowledge graphs for recommendation systems.

5.6 Discussion of the Results

Evaluating and testing the framework proposed was on multiple stages. The first stage of evaluation was through optimizing our models using a hyper-parameter Grid Search CV technique. This optimization fine-tuned our models with the best parameters to get the best outcome. Three different classifiers were tested on every entity to be predicted and multiple parameters were set to choose the best one to use. The hyper-parameter values, when set right, can build highly accurate models, this is the reason why we tried multiple combinations of parameters to finally choose the best set.

The second stage of evaluation was retraining the best models obtained from stage one (optimization phase). Through experimenting our automatic tool, we collected and checked newly machine predicted annotations and then retrained the models with new sets of data. Training and testing the models followed a specific pattern that pertains a fixed testing set in order to provide an unbiased final model performance metric in terms of accuracy, precision, etc. Training sets were growing with every newly collected batch of checked annotations, while having a fixed testing set, in addition to another growing testing set for further validation. Accuracy and model performance were recorded on every set trained and tested to see how the models are being improved using the human intervention technique proposed.

Upon adding 400 new annotations to the training set of the models using human feedback mechanism, the adaptation of ML models was achieved. It is good to mention that trying a smaller batch of annotations didn't show significant results in terms of better model performance. Completing this cycle of collecting data, predicting action knowledge entities by machine, checking and approving predicted results by annotators, and finally retraining models with newly updated sets of data, will optimize our algorithm's outcomes and the automatic framework will be incrementally updated. This continuous cycle of retraining the models with bigger datasets will rest out to a level where human integration can be reduced and limited.

Chapter 6

Conclusion

Behind a recommender system, appears a combination of knowledge, processes, techniques, and models that calibrate and improve users experience with the system. The best source of knowledge is customer reviews that hold a wealth of captured user experiences and emotions regarding a specific product. The contribution of this thesis is to automate the action knowledge extraction from customer reviews and the knowledge graph construction from machine predicted entities. We were interested in showing to what level can we integrate these automatically predicted action knowledge and constructed KGs in the recommendation system of products. Moreover, the thesis focuses on the adaptation of this automatic framework using human-in-the-loop machine leaning technique, and how this intervention can be reduced with time when frequent retrain of models occur.

Addressing our research questions, the approach proposed by this thesis for the purpose of automating action knowledge extraction was having adaptive machine learning models that accepts new customer reviews from the web and predict multiple entities. The models were able to split the review text by sentences, annotate the ones that refer to an existing action and expand the prediction process to extract other related semantics. The automatic framework embraced multiple ML algorithms like multi-class classifications, similarity techniques, and knowledge graph-based methods. Multiple phases of the framework were presented in the methodology: Generation phase, Refinement phase, and Mapping phase.

The framework also maintained the process of integrating product reviews in the recommendation systems automatically, through their representative knowledge graphs. The framework will automatically construct the KG of each annotation from the machine predicted entities and push it to KG triplestore to enter the recommendation process of products. This automatic KG construction helps in reducing the time consumed to design a graph for every annotation, especially when dealing with the rapid increase in domain data.

The human-in-the-loop mechanism was an advanced feature proposed in this thesis. The purpose of integrating human in the automatic framework was improving the initial data of the ML models. In addition to retraining these models multiple times to reach optimal accuracy scores. These goals were achieved and validated through an experimental protocol that involved human participation in collecting new data using the automatic annotation tool. The stages of the retraining experiment proved the adaptation of ML models upon the frequent collection of large amounts of new data. This recurrent retraining of ML models incrementally update the performance of our automatic framework in terms of better predictions of entities, thus better knowledge graph construction from those predicted entities.

The web app prototype developed was a proof of concept to ultimately replace the manual annotation tools that exist. A clear scenario was written to demonstrate the steps for using the app as a user, annotator or admin. The app consisted of 4 pages, each of multiple tasks and jobs to perform. A full consolidated cycle of tasks was described in the detailed scenario, showing how the automatic annotation process happens. The app was used to do the evaluation experiment that validated model's performance while updating the training data with newly checked annotations.

Evaluating the automatic framework of this thesis paper was done on multiple stages, first a hyper-parameter optimization technique that resulted with the best parameters for each model to leverage model performance. Another experiment was based on collecting human checked annotations using the automatic tool we developed. The evaluation was done over one month period of time. Checked annotations were appended to the initial training set to retrain the models on new dataset and monitor performance. To avoid any bias in our experiment, the same testing set was scored using the updated versions of the models. Results of this experimental protocol applied were promising as we started to observe better performance in models when more training data was collected. This continuous collection of data improved our dataset and re-sampled it properly to give accurate results. One of the tasks performed by this automatic framework is to increase the training set with time through human participation. We recommend collecting batches of at least 400 new annotations before a new retraining round takes place, as our experiment recorded a change in accuracy after adding 400 new review annotations to the training set. A smaller batch of annotations may not give significant results.

The final stage of the framework builds automatically an openly accessible knowledge graph for every checked annotation. This automatically constructed knowledge graphs take place in the recommendation process of products in an action-aware recommendation app. With vast amount of automatically annotated data over the web, we will be able to integrate more products in the recommendation app saving time with better results.

6.1 Meeting Aims and Objectives

This thesis project tried to meet all aims and objectives set in the Introduction. Our goals were achieved in this automatic framework and the results were appealing. The goals accomplished are:

1. The automatic framework proposed in this thesis assisted human in detecting and representing action knowledge from product reviews.
2. A well constructed flow of work is implemented in the prototype built connecting multiple endpoints of the framework.
3. The web app prototype developed, is an online accessible tool that helped users accomplish their roles in an easy way.
4. HITL is used to optimize model performance by collecting more training data to the models, thus incrementally updating the framework with new versions of the ML models.
5. Evaluation stage supported our thesis and showed the impact of model optimization and human intervention on the adaptation of the framework.

The contributions set at the beginning of the thesis were achieved by constructing the automatic framework that is meant to replace the existing manual annotation tools. Our proposed framework succeeded in automatically extracting action knowledge entities from product reviews and constructing a well designed knowledge graph connecting these entities. The framework was guided with HITL-ML technique that proved its effectiveness and impact on the adaptation of ML models of the framework.

6.2 Research Limitations

Customers reviews are one of the free-text examples in NLP that require a set of annotated data to teach the models how to predict the classifications of the respective text. This study is limited with its number of review text available for training usage, besides having them all in English language. As part of our future plans, extending the work of these multi-class and multi-label classifiers to shift from monolingual text classification to multilingual text classification maintaining the coherence and well-structure of the framework and resulting with the same set of needed labels no matter what the free-text review input is going to be [35].

6.3 Future Work

This study can benefit from several research approaches. First, the data used was limited with its number of reviews text available for training usage, with having them all in English language. In the future, extending the work of these multi-class and multi-label classifiers to shift from monolingual text classification to multilingual text classification.

Another direction that can be tackled in the future is generalizing the domain of our research because of the dramatic increase in the use of knowledge graphs in almost all domains. The purpose is to generalize our automatic framework job to extract action knowledge from multiple types of unstructured textual data, instead of limiting the models to product reviews from e-commerce websites. This thesis was limited to the initial training set that we worked with, which was product reviews from 'Computers and Tablets' category. Domain generalization will start by adapting other products from other categories in Best Buy in this automatic framework. Then successively expand our domain environment to reach out multiple categories from multiple e-commerce websites. This domain adaptation will be based on the similarity between the domains, thus having their training data closer in the embedding space of action knowledge entities. In contrast, having different domains from different environments will be more challenging to integrate in the same embedding space.

One of the framework's characteristics discussed in this project is the incremental update that is due to the human intervention mechanism that aims to upgrade our ML models by retraining with new sets of checked data. This human intervention will be reduced with time as models score better prediction results.

Some future work will be applied to our framework to make sure it serves our purpose in automating the extraction of action knowledge and the construction of knowledge graphs perfectly well. We will continuously retrain our ML models with newly checked data for better performance. The design of the prototype's interface will be improved for a more user-friendly app. Improving some functionalities of the app like looping over all product pages automatically, to capture all reviews at once. Another example is to give access to add entities or labels not found in the select boxes, these new entities will automatically expand our knowledge graph design. Finally, we aim to minimize human intervention level with time upon models reaching a trusted level of performance.

Appendix A

Appendix

A.1 Personal Reflection

This research brought together different aspects of machine learning classification models and knowledge extraction. The thesis made use of the fundamental machine learning aspects we have learnt during our studies. This knowledge was reinforced with experience in multiple projects previously done during our year bringing good practices not just from a structural perspective, but also a maintainability and efficiency standpoints.

The project also made good use of the concepts learnt during the courses addressing how a good storyboard should be built to make a perfect yet simple flow of ideas for our audience. This knowledge was very important in ensuring data was being represented appropriately and therefore analysis could be as efficient as possible.

A.2 Code Links

The GitHub repository having all scripts and the data engineer offline page is found in the link below:

GitHub Public Repository

The Automatic Annotation Web App

The admin page to get new reviews:

The Admin page

The Annotator Page were a group of authorized people can annotate review texts.

The Annotator Page

The Checked Annotation page where admin can view all checked annotations and decide on the best one to be pushed to triple store.

The Checked Annotation Page

Retraining the models will be in a seperate page where the admin can view previous models performance and decide which model should be retrained and updated.

The Retraining Models Page

Bibliography

[1] I. Roldós, "How to effectively analyze customer and product reviews," *MonkeyLearn Blog*, Mar. 2022. [Online]. Available: https://monkeylearn.com/blog/reviews-analysis/.

[2] S.-S. Weng and M.-J. Liu, "Feature-based recommendations for one-to-one marketing," *Expert Systems with Applications*, vol. 26, no. 4, pp. 493–508, 2004, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2003.10.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741740300188X.

[3] E. Elnagar, A. Zeoli, R. Rahif, S. Attia, and V. Lemort, "A qualitative assessment of integrated active cooling systems: A review with a focus on system flexibility and climate resilience," *Renewable and Sustainable Energy Reviews*, vol. 175, p. 113 179, 2023, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2023.113179. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032123000357.

[4] T. Rim, S. Dervaux, P. Buche, L. Ibanescu, and J. Dibie, "Ontology evolution for an experimental data integration system," *International Journal of Metadata, Semantics and Ontologies*, vol. 11, p. 231, Jan. 2016. DOI: 10.1504/IJMSO.2016.10004259.

[5] Z. Ma, H. Cheng, and L. Yan, "Automatic construction of owl ontologies from petri nets," *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2019. [Online]. Available: https://ideas.repec.org/a/igg/jswis0/v15y2019i1p21-51.html.

[6] F. Zablith, "Actionrec: Toward action-aware recommender systems on the web," in *Proceedings of the 20th International Semantic Web Conference (ISWC) Demos. CEUR Vol. 2980.*, 2021. [Online]. Available: https://fouad.zablith.org/docs/ISWC2021_Demo_FZablith.pdf.

[7] Z. K. A. Baizal, A. Iskandar, and E. Nasution, "Ontology-based recommendation involving consumer product reviews," in *2016 4th International Conference on Information and Communication Technology (ICoICT)*, 2016, pp. 1–6. DOI: 10.1109/ICoICT.2016.7571890.

[8] R. Monarch and C. D. Manning, *Human-in-the-loop machine learning active learning and annotation for human-centered AI*. Simon and Schuster, Jun. 2021, ISBN: 9781638351030.

[9] A. Timoshenko and J. R. Hauser, "Identifying customer needs from user-generated content," in *Proceedings of Marketing Science 38(1):1-20.*, Jan. 2019. [Online]. Available: https://doi.org/10.1287/mksc.2018.1123.

[10] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Proceedings of Dept. of Informatics, Aristotle University of Thessaloniki, 54124 Greece*, Jan. 2010. [Online]. Available: http://lpis.csd.auth.gr/publications/tsoumakas09-dmkdh.pdf.

[11] M. Sahare and H. Gupta, "A review of multi-class classification for imbalanced data," *International Journal of Advanced Computer Research*, vol. 2, no. 3, p. 160, 2012.

[12] J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour, "Boosting methods for multi-class imbalanced data classification: An experimental review," en, *J. Big Data*, vol. 7, no. 1, Dec. 2020.

[13] M. Lango and J. Stefanowski, "What makes multi-class imbalanced problems difficult? an experimental study," *Expert Systems with Applications*, vol. 199, p. 116 962, 2022, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.116962. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422003888.

[14] R. B. Pereira, A. Plastino, B. Zadrozny, and L. H. Merschmann, "Categorizing feature selection methods for multi-label classification," *Artificial Intelligence Review*, vol. 49, no. 1, pp. 57–78, 2018.

[15] F. S. Al-Anzi and D. AbuZeina, "Toward an enhanced arabic text classification using cosine similarity and latent semantic indexing," *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 2, pp. 189–195, 2017, Arabic Natural Language Processing: Models, Systems and Applications, ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2016.04.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157816300210.

[16] H. V. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," in *Asian Conference on Computer Vision*, 2010.

[17] H. G. Silber and K. F. McCoy, "Efficiently computed lexical chains as an intermediate representation for automatic text summarization," *Computational Linguistics*, vol. 28, no. 4, pp. 487–496, 2002. DOI: 10.1162/089120102762671954. [Online]. Available: https://aclanthology.org/J02-4004.

[18] A. F. El Gohary, T. I. Sultan, M. A. Hana, and M. El Dosoky, "A computational approach for analyzing and detecting emotions in arabic text," *International Journal of Engineering Research and Applications (IJERA)*, vol. 3, no. 3, pp. 100–107, 2013.

[19] J. Chicaiza and P. Valdiviezo-Diaz, "A comprehensive survey of knowledge graph-based recommender systems: Technologies, development, and contributions," *Information*, vol. 12, no. 6, p. 232, 2021.

[20] L. F. B. PhD, “People’s words and actions can actually shape your brain - a neuroscientist explains,” *ideas.ted.com*, Nov. 2020. [Online]. Available: https://ideas.ted.com/peoples-words-and-actions-can-actually-shape-your-brain-a-neuroscientist-explains-how/.

[21] G. Boleda, S. S. im Walde, and T. Badia, “Modelling polysemy in adjective classes by multi-label classification,” in *Conference on Empirical Methods in Natural Language Processing*, 2007.

[22] N. Ghamrawi and A. McCallum, “Collective multi-label classification,” University of Massachusetts - Amherst: ScholarWorks@UMass Amherst, 2005. [Online]. Available: https://scholarworks.umass.edu/cs_faculty_pubs/190.

[23] J. Shin, S. Wu, F. Wang, C. De Sa, C. Zhang, and C. Ré, “Incremental knowledge base construction using DeepDive,” en, *Proceedings VLDB Endowment*, vol. 8, no. 11, pp. 1310–1321, Jul. 2015.

[24] D. Q. Nguyen, “An overview of embedding models of entities and relationships for knowledge base completion,” *ArXiv*, vol. abs/1703.08098, 2017.

[25] Y. Péron, F. Raimbault, G. Menier, and P.-F. Marteau, “On the detection of inconsistencies in rdf data sets and their correction at ontological level,” Jun. 2011.

[26] J. Johnson, “What is human in the loop (hitl) machine learning?” *BMC Blogs*, Aug. 2020. [Online]. Available: https://www.bmc.com/blogs/hitl-human-in-the-loop/.

[27] C. Deng, X. Ji, C. Rainey, J. Zhang, and W. Lu, “Integrating machine learning with human knowledge,” en, *iScience*, vol. 23, no. 11, p. 101 656, Nov. 2020.

[28] S. Kimura, “Why we will always need humans to train ai - sometimes in real-time,” *KDnuggets*, 2021. [Online]. Available: https://www.kdnuggets.com/2021/12/why-we-need-humans-training-ai.html.

[29] “Sparqlwrapper,” *PyPI*, [Online]. Available: https://pypi.org/project/SPARQLWrapper/.

[30] “Sparql query language for rdf,” *SPARQL query language for RDF*, [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/.

[31] “Mysql data base management system,” *MySQL*, [Online]. Available: https://www.mysql.com/.

[32] “Mysql-connector-python,” *PyPI*, [Online]. Available: https://pypi.org/project/mysql-connector-python/.

[33] “Jprops,” *PyPI*, [Online]. Available: https://pypi.org/project/jprops/.

[34] J. Jordan, “Hyperparameter tuning for machine learning models.,” *Jeremy Jordan*, Dec. 2018. [Online]. Available: https://www.jeremyjordan.me/hyperparameter-tuning/.

[35] A. Byström, “Extending a text classifier to multiple languages,” M.S. thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2021, p. 35.