Learning Relational Patterns Mined from Temporally Rich Textual Data

Aryana Tavanai
MSc Artificial Intelligence
2010/2011

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)_____

Summary

The goal of this project was to investigate the capability of using patterns extracted from graph mining, in order to learn different classes of textual recipes in a novel approach. To accomplish this, by applying various machine learning and natural language processing methods and techniques, different representations of the patterns were experimented on and evaluated using a supervised and an unsupervised approach.

Before designing and implementing the project, a literature review was undertaken to analyse previous work in the domains of text mining, graph mining and machine learning which provided sufficient information and ideas to design the project. Moreover, various tools and libraries were employed to aid the implementation phase saving valuable time. Using the knowledge obtained in the literature review, a system was designed and developed followed by an evaluation which was performed on different components of the system.

This project aimed at creating a novel method of approaching textual data in natural language processing.

Acknowledgements

Most of all I would like to thank Dr. Muralikrishna Sridhar for his enthusiasm and unending help in this project. Without his expertise and advice this project would have not been where it is today. It has been a great experience working with him.

I would also like to thank my supervisor, Prof. Tony Cohn and assessor, Dr. Vania Dimitrova for their valuable feedback which gave direction to the project.

Finally I would like to thank my father for his support, my mother for her patience and last but not least, my beloved sister for her encouragement throughout my academic life.

Contents

Chapter 1

Introduction

1.1 Overview

One of the main challenges in cognitive science and artificial intelligence, is how to describe and model the way that a cognitive agent builds and evolves representations by interacting with the environment. It is the task of knowledge representation to understand the representation-building mechanisms in human cognition and to model them in an artificially intelligent system. The modelled representation-building mechanism may also be suitable for application in various other domains [31].

In cooking, when a person reads a recipe, the person may associate it with a certain class of recipes. This association is made from the text within the recipe. However, only certain parts of the recipe provide information on the class of the recipe which are given as representations. The representations, also referred to in this report as patterns, may be built from the ingredients in the recipe (nouns), the way the ingredients are used (verbs) or the sequence of actions that are performed in the recipe.

In this project, by applying a supervised and an unsupervised approach, representations originally developed for video activity analysis by Sridhar [53] will be used to find patterns that are suitable for predicting the class of a recipe. These patterns represent relations between verbs and nouns, obtained from a set of temporally sequenced instructions in recipes. The built model mimics the human cognition of recipes with basic representations. It is worth mentioning that modelling of the human cognition is not the aim of this project.

1.2 Aim & Objectives

The overall aim of this project is to create a framework for finding common patterns that occur in text that describe human activities, specially when they are described as a temporal sequence of instructions. Suitable examples of such text are cooking recipes. This framework consists of applying natural language processing, knowledge representation and machine learning techniques on a textual recipe corpus for recipe categorization and clustering. To create the framework and accomplish this aim, the following objectives must be reached:

1. Creating a number of multi-class corpora of recipes, where each recipe consists of several temporally ordered set of instructions.
2. Applying pre-processing on the corpus to extract the nouns and verbs of the recipes
3. Create temporally structured graph representations of the pre-processed recipes.
4. Mining the set of graphs corresponding to the recipes to extract subgraph patterns for creating a feature set.
5. Applying supervised and unsupervised learning on the obtained feature sets to learn models for predicting the recipes.

1.3 Minimum Requirements & Deliverables

The following are the minimum requirements and deliverables set for this project:

1. Designing a representation for the temporal relations.
2. Producing an event learning system.
3. Developing a prototype system to classify corpora recipes based on their graph structures.

1.4 Relevance to Degree

This project builds on knowledge and skills obtained from modules taken during the authors MSc and BSc courses in the School of Computing at the University of Leeds. The main three modules include Language (COMP5410M), which provided knowledge of processing and extracting information from text, Knowledge Representation (COMP5450), which discussed several representational formalisms and Machine Learning (COMP5425M), where various learning techniques and evaluation methods were introduced. This project provided the challenge of combining the three fields of artificial intelligence and applying them in a novel way to the cooking domain.

1.5 Project Management

The initial and revised project schedules are provided in Figures D.3 and D.4 in Appendix D. The initial project schedule illustrates the tasks and subtasks set in order to complete this project. This

schedule consists of two milestones, namely interim report and final report. Between the two milestones, an initial prototype *P1* and a final prototype *P2* of the system was developed developed.

In the revised schedule, the length of the implementation and experimentation tasks were extended as many more challenges were experienced during the implementation of *P2* where obtaining higher accuracies compared to *P1* and the bag of words method proved to be more complex than initially thought. At the time of submitting this project all the tasks set in the project schedule were completed.

1.6 Research Methodology

A suitable methodology for this project is the *evolutionary prototyping* approach [30]. This approach builds an initial prototype which is evaluated. Based on its performance in the evaluation, the prototype is modified to overcome faults or shortcomings it has and may be extended to add new features. The modified prototype is evaluated again and this cycle is repeated until the final prototype satisfies the aims and objectives of the required framework. This approach is mainly used when the aims and objectives of a project are known, however, uncertainty exists in the way they will be achieved. In this project, uncertainty exist in the processing of natural language, the representation of the verb relations and the optimal learning methods to be implemented in the final framework. A drawback of the evolutionary prototyping approach however, is the complexity of controlling the changes that are made to the system [30, 56].

The initial prototype in this project was the creation of a framework which extracts graph structures from an artificial text corpus with the aid of a dictionary. This framework uses programs obtained from Sridhar [53], which were modified to work on the artificial corpus. The prototype was then extended to create representations of these structures and using them for an unsupervised learning method such as k-nearest neighbour. Once this prototype was built, it was then modified to applied on a real recipe corpus resulting in the prototype *P1* in the project schedule. This prototype was a baseline working system which provides results. However, these results may not be optimal.

Prototype *P2* is a modified prototype of *P1* which was extended to use a variety of machine learning and graph mining techniques. This prototype was used to experiment and evaluate the implemented methods and parameters with the aim of finding a suitable framework to answer the questions outlined in section 1.8. The prototype development plan is provided in Appendix C.

To find a suitable framework, methods implemented in the framework must be evaluated. For the supervised aspect of the framework, the bag of words method is used as baseline for evaluation. The unsupervised aspect of the framework is evaluated against clustering made by participants in a cognitive clustering experiment. The components of the framework are presented in the next section.

1.7 Framework Overview

To investigate the benefits of using relational patterns, the following components were used in the framework:

1. **Corpus Creation:** A set of recipe corpora were created so that the framework could be applied on. Each corpus consisted of multiple classes and various number of recipes.
2. **Pre-processing:** This stage was required to process raw text to a form that facilitates the extraction of relations between verbs and nouns. This component uses four methods, namely dictionary lookup, resolving of missing nouns, substituting verbs and encoding to achieve this.
3. **Graph Based Relational Representations:** Relations between verbs and nouns are represented as a three layered graph. The three layers of the graph are namely the entity, relation and temporal layers. These intervals between the layers correspond to the relations between nouns and verbs, and verbs and temporal relations respectively.
4. **Feature Representation:** For categorisation and clustering, the three layered graphs are mined to discover subgraphs to be used in a feature set.
5. **Feature selection:** Feature selection was applied to obtain the most suitable features, in order to obtain subgraphs that ensure maximum discrimination between classes in the context of the supervised approach.
6. **Learning:** Both supervised and unsupervised learning was applied on the selected feature set for classification and clustering respectively.

1.8 Research Questions

By completing the project, the following questions were set to be answered.

1. Can nouns and verbs be accurately mined from a recipe using Natural Language Processing techniques?
2. Is it possible to find the most representative pattern in a class?
3. What is the most suitable structure for a pattern representation?
4. Can the class of a recipe be predicted from its instructions?
5. Will an unsupervised approach be more suitable than a supervised approach to be applied by the framework?
6. Will the framework be able to automatically find the number of clusters which represent the classes of recipes accurately.
7. Is it possible to classify cooking recipes with graph representations using unsupervised methods?

1.9 Thesis Overview

In Chapter 2 of this report, a literature review of the methods suitable for the implementation of the framework is presented. This review is done for various relevant areas to the framework, namely natural language processing, data mining, knowledge representation, event learning and machine learning.

In Chapter 3, the framework of components outlined in section 1.7 are described. These are namely, corpus creation, pre-processing, graph based relational representations, feature representation, feature selection and learning. The theoretical basis of each of the mentioned components are described.

In Chapter 4, various experiments are provided to evaluate the methods and representations used in the framework. The experiments in chapter 4 are divided into four main sections, namely natural language processing, knowledge representation, machine learning and evaluation of the final framework as a whole.

In Chapter 5 the overall evaluation of the project is given by describing its achievements, answers to research questions and the future work set due to the limitations of the project.

Chapter 2

Literature Review

Before building a framework, it is important to investigate available methods that have been previously employed. Therefore in this chapter, various fields and methods related to the building process of the framework components described in section 1.7 will be reviewed.

As the project will be applied on recipes in a textual from the first step is to extract the data needed from the recipes such as nouns and verbs from the recipes as well as removing any additional information. By using pre-processing on the textual recipes, nouns and verbs may be extracted while removing any additional information. Pre-processing can be achieved by applying natural language processing techniques on the recipes which will be described in the next section.

2.1 Natural Language Processing

Natural Language Processing (NLP) is a field in Artificial intelligence which is concerned with the interaction between computers and human language. Using NLP, human language may be converted into information for computers or vice versa. One of the difficulties in the conversion of written text to information is the ambiguity in written text, namely *lexical* and *syntactic* ambiguity [37].

Lexical ambiguity is when a word has multiple meanings and the computer does not know which meaning is correct in the sentence. This ambiguity can be resolved by considering other words in the sentence or the grammatical position of the word. The method of finding the correct meaning of a word is referred to as *word sense disambiguation* [39].

Syntactic ambiguity occurs if there are more than one grammatical structures, or parses, for a sentence [37]. One of the methods to overcome this ambiguity is to use *Probabilistic Context Free Grammars* (PCFG) to create all parses and find the one with the highest probability to be suitable [46]. In this project, it is important to find the correct parse of a sentence since correct entities and events in recipes must be found. Parsing may be used to overcome this problem.

Parsing is the process of grammatically structuring a sentence based on the set of rules given as grammar [25]. This is done by producing a parse tree as illustrated in Figure D.2 in Appendix D, which represents the syntactic structure of a sentence. The interior nodes are grammar nodes and the leaf nodes are words in the sentence [66]. parsing can be applied through two methods, namely *top-down* and *bottom-up* parsing. In top-down parsing, the original production process of the sentence is imitated by starting from the root node and re-deriving the sentence from top to bottom. The reverse process, bottom-up parsing, is to start from the sentence and reduce it to the root node. Although top-down parsers are simpler to code, they are computationally inefficient for backtracking and recursion. Bottom-up parsers are more efficient with recursion, however, they are more complex to code [25].

Parsers provide the correct parse of a sentence along with the grammatical position of all words. An alternative to parsers are part-of-speech (POS) taggers. POS taggers provide the grammatical position of a word based on its lexical and contextual information. Two accurate POS taggers are the Brill tagger and Stanford tagger. Unlike parsers, POS taggers do not provide a parse or a structure for the sentence, however, they require less time to run, are easier to implement and obtain an accuracy near the level of human performance [19].

In this project, by using a parser or a POS tagger on a recipe, the nouns are extracted as entities and the verbs as relations. After the nouns and verbs have been extracted, they may be used for text classification. Two main areas of using temporal structures based on text were found during the literature review of natural language processing techniques. The first is an *N-grams* based text categorization and the second is where temporal structures of text are used for learning events and patterns. These two areas are described below.

Used by Khreisat [35] and Suzuki [55], n-grams are common ways of categorizing text. The proposed methods in the mentioned papers, use word n-grams and character n-grams to extract index terms consisting of various combinations of sequences in words and word characters respectively. The character n-gram is more often used compared to the word n-gram method, as it has proven to be more accurate [63]. To categorise text, a bag of words method is used where the features are represented as the tokens extracted. By obtaining a feature vector for each text, using the feature representation of the bag of words, a feature set is created. By applying learning methods such as an SVM, texts are categorised. The sequence of words or characters may be considered as a temporal structure between

the words or characters. However, the tokens extracted by using these type of relations are limited to only a certain range of sequence of words or characters where in each case no gaps exist in the sequence. The framework proposed in section 3 aims this shortcoming by obtaining relational patterns based on Allen relations, described in section 2.2. This enables finding patterns between words which are separated by other words.

The second area where temporal structures are considered in natural language processing, is where they are extracted from text and used for learning events and common patterns [61, 17] in a collection of texts. The different methods used in the mentioned papers for this task are the following. By using a parser, data is extracted and by using Allen relations or hand-built rules, temporal relation are obtained which are represented as graphs. By using a Markov Logic Network (MLN) classifier or a greedy inference method, the main temporal structures of the texts are extracted. It was observed that other methods used in this area do not directly use the temporal relations as features, unlike the first area described, where a bag of words method is created using the extracted tokens as features.

The framework provided in section 3 combines both areas where initially, temporal relations are extracted between verbs in recipes and directly used as features in a method similar to the bag of words method.

Before applying data mining to extract the features in data, they require to be presented by a suitable temporal representation. In the next section a suitable representation is described.

2.2 Temporal Knowledge Representation

According to Galton [22], the conception of time is determined by the way time is specified and measured. The measurement of time may be defined as the temporal duration of an event or interval. The specification of time is locating an event in a time series relative to other events which may be done *qualitatively* and *quantitatively*. An example of qualitative specification is to say event E_1 occurred after event E_0 and before event E_2. To make this also quantitative, the duration of the intervals between each of the mentioned events should be specified. An important relation that is found in the quantitative aspect of temporal ordering is the *precedence* relation. This relation is the primitive basis for creating models with a qualitative nature and has the following properties [22].

1. Irreflexibility: No instant precedes itself.
2. Transitivity: If t precedes t and t' precedes t'', then t also precedes t''.
3. Linearity: Of any two distinct instants, one precedes the other.
4. Past unboundedness: For every instant there is an instant which precedes it.
5. Future unboundedness: For every instant there is an instant which it precedes.
6. Density: Between any two distinct instants there is a third instant which precedes one and is preceded by the other.

7. Continuity: If a set S of instants is such that (a) no instant in S is preceded by an instant not in S and (b) there is at least one instant not in S, then either (c) there is a last instant in S or (d) there is a first instant not in S.

The mentioned model is based on instances and ordering relations, however in knowledge representation within artificial intelligence, using intervals is more beneficial than using instants [22]. Allen [2] proposed an interval based model and showed that by using a single primitive *meets*, all qualitative relationships between intervals may be expressed. For example i_1 meets i_2 means that interval i_1 ends exactly where interval i_2 starts. Intervals and the 13 distinct qualitative relations between them are known as Interval Calculus [22]. The following are possible relations intervals may have.

<:	is before	>:	is after
m:	meets	mi:	is met by
o:	overlaps	oi:	is overlapped by
s:	starts	si:	is started by
f:	finishes	fi:	is finished by
d:	is during	di:	contains

=: equals

Using the above relations the instructions in a recipe may be temporally represented. Information from this representation may be extracted by applying data mining which is described in the next section.

2.3 Data Mining

Applying Machine Learning techniques to a large databases is called Data Mining [4]. It is the process of finding unknown patterns or relationships in data. This is achieved by selecting, exploring and modelling large data [24]. According to Bellazzi [7], data mining tasks can be classified as two main tasks of description and prediction. Description tasks aim at finding patterns and associations which are human-interpretable. Prediction tasks however, predict a response of interest after constructing a model on all the data. Although the aims of the description and prediction tasks may sometimes overlap, for example the model obtained by the prediction tasks provides useful human-interpretable patterns, however, their main difference is that the prediction tasks require a response variable which will be used for prediction, similar to supervised learning. Therefore prediction data mining may be considered as a classification tool. Both description and prediction tasks may be used in this project to obtain results.

Data mining may be applied to different types of data in order to create a model. The following are common data mining methods applied to certain data types.

2.3.1 Text Mining

Text mining is a method used to discover patterns from collections of large unstructured text [60]. It is based on methods such as Natural Language Processing, Information Retrieval and data mining. Although different, it may be thought that text mining and Information Retrieval are similar in what they achieve. Hearst [28], one of the founders of text mining, differentiates text mining from Information Retrieval by the way each accomplish its task. The goal of Information Retrieval is to return a document that consists of information a user requires. Text mining does not concern unknown information, but rather it deals with finding known information in a larger collection of information. Therefore, no new discovery is made. In text mining new information may be created and patterns may be found by going through large collections of text.

Text mining is one of the main methods used in the Biomedical and Molecular Biology domains as the size of biological textual data rapidly increases. It is not feasible for humans to process the large collection of text and text mining is therefore used to automate the process of knowledge discovery and finding patterns [1]. Uramoto et al. [60] have developed a system called MedTAKMI which uses text mining to mine the entire MEDLINE database which consists of 12 million citations from various fields of medicine. The main objective of this system is to extract biomedical concepts such as genes and proteins. This is performed by undertaking two primary subtasks:

1. Entity extraction: The recognition and extraction of gene, protein and chemical names from biomedical text.

2. Relation extraction: The extraction of relationships among the mentioned entities.

The MedTAKMI system performs entity and relationship mining by using two steps. The first step extracts entities using a dictionary lookup where the dictionary contains 2 million biomedical entities. This method is conceptually simple and efficient. As the entities are now identified, in the next step the system will extract relations among the entities which may have various patterns. Examples of such relations are ”A inhibits B” and ”A activates B” which have the pattern ”Entity Relation Entity” and a syntactic structure of ”Noun Verb Noun” [60]. The MedTAKMI system may be a suitable approach for this project. This project also aims to extract entities and relations and to store them as patterns. In this project, Similar MedTAKMI, the extraction of entities and their relations in this project may also be achieved by using a dictionary lookup which acts similar to MedTAKMI.

2.3.2 Graph Mining

Graph mining is a method which provides new principles and patterns by mining spatial or temporal properties of substructures embedded in graph data. Washio and Motoda [62] explain the theoretical basis of graph mining, described in section 2.3.2.1, and search methods used to solve graph mining problems, described in section 2.3.2.2.

2.3.2.1 Theoretical Bases of Graph Mining

The following are four theoretical bases of graph mining:

Subgraph Categories In graph mining, different classes of substructures are used for mining as graph data are very generic and include characteristic substructures. In Figure 2.1 an example of a graph is illustrated along with its main substructure classes of its possible subgraphs.

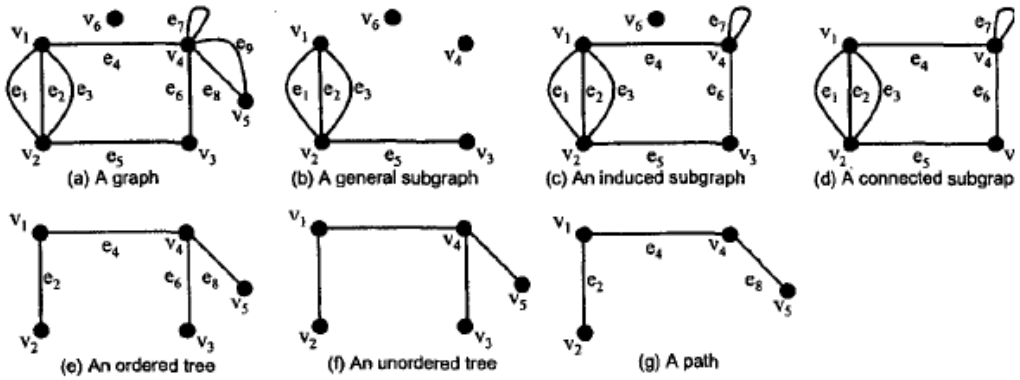


Figure 2.1: Representative subgraphs [62]

A graph $G(V,E,f)$ is represented in Figure 2.1 (a) where $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ is the set of vertices, $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$ is the set of edges connecting some vertex pairs and f is a mapping $f : E \rightarrow V \times V$. Figure 2.1 (b) illustrates the most generic class of G, a *general subgraph*. This subgraph has a vertex set V_s and an edge set E_s where $V_s \subset V$, $E_s \subset E$ and for all edges $f(e_h) = (v_i, v_j) \in E_s$. An *induced subgraph* is another generic class of the substructures shown in Figure 2.1 (c). In an induced subgraph G_s^i, if a vertex exists in G_s^i from G all edges belonging to that vertex from G must also be in G_s^i. Another class of the substructure is a *connected subgraph* where $V_s \subset V$, $E_s \subset E$ and all vertices in V_s are mutually reachable through some edges in E_s. An induced and connected subgraph is illustrated in Figure 2.1 (d).

An acyclic subgraph is called a *tree*. An *ordered tree* is a tree with the labels of edges introduced on the tree. In this case an edge label is always younger than the labels of its upper left edge. An unordered tree is not labelled or the labels are not ordered. Figures 2.1 (e) and (f) illustrate an ordered and ordered graph respectively. If the vertices of a connected substructure have the degree of either one or two, the substructure is called a *path* which is show in Figure 2.1 (f).

Subgraph Isomorphism Subgraph isomorphism is used when identical subgraphs are matched among two graphs. Given a set of graphs $\{G_K(V_k, E_k, f_k) | k = 1, ..., n\}$, in graph mining, the subgraph isomorphic problem is to find a set of subgraphs $\{G_sK(V_sk, E_sk, f_k) | k = 1, ..., n\}$, a subgraph

$G_s(V_s, E_s, f_s)$ and a bijection mapping f_s between the vertices of G_s and every G_sk for all $k = 1, ..., n$. When a mapping f_s is found satisfying the mentioned conditions, $G_s(V_s, E_s, f_s)$ is a common subgraph of the given set of graphs [62]. Similarly, subgraph isomorphism may be used to search for a graph within the set of subgraphs of another graph, i.e. if a graph is a subgraph of another graph [27]. One of the disadvantages of subgraph isomorphism is that it is NP-complete. Therefore, a measure is needed to reduce the computation time and remove unnecessary comparisons.

Graph Invariants Graph invariants are measurements to characterize the structure of a graph topologically. If two graphs are isomorphic, they are topologically identical which will result in them having identical graph invariants, however, the reverse of this property does not hold. Graph invariants may be represented as the number of vertices, edges, the degree of each vertex and the number of cyclic loops a graph may have [62]. Various and more complex values for graph invariants Include integer values such as the Wiener index, rational numbers such as the Kirchoff numbers and real numbers such as the average distance-sum connectivity [6]. If any of the graph invariants of a graph is different to the same graph invariant of another graph, the two graphs are not isomorphic [62]. The nauty algorithm, developed by McKay [44], is one of the most powerful graph isomorphism algorithms. It uses various graph invariants, chosen by the user, in order to increase the speed of subgraph isomorphism. Despite its high increase in computational efficiency, for certain families of graphs it is forced to run in exponential time [50].

Mining Measures: Similar to data mining, certain measures are needed in graph mining to mine substructures of graphs. Various measures exist such as information gain, information entropy and minimum description length [14, 62]. However, the most popular measure in graph mining is the *Support* measure. The support of a subgraph G_s, $sup(G_s)$, for a given graph dataset D is calculated by Equation 2.1.

$$sup(G_s) = \frac{\text{Number of graphs including } G_s \text{ in } D}{\text{Total number of graphs in } D} \tag{2.1}$$

The support measure has an anti-monotonic property where if G_sy is a subgraph of G_sx therefore $sup(G_sx) \leq sup(G_sy)$. Using a *minimum support* value *minsup*, only subgraphs which have support values more than *minsup* will be mined in order to avoid subgraphs which do not exist in many graphs. similarly a *maximum support*, *maxsup* may also be used [62].

By using the mentioned measures, a solution set is obtained which consists of subgraphs G_s that exist in more than a certain number of graphs (minsup) in D, where each subgraph satisfies constraints. However, in order to obtain these solution sets, search methods need to be used to find the suitable subgraphs. In section 2.3.2.2 various types of search methods will be described.

2.3.2.2 Solution Methods

The following are search methods used for solving a subgraph isomorphism problem among various graphs. Each of the following methods is either a direct or indirect matching method. A direct matching method is a method that finds a solution for the subgraph isomorphism problem. An indirect matching method is a method that does not solve the subgraph isomorphism problem, however, it may be used to solve the subgraph similarity problem. Each of the methods is also categorized as a heuristic or complete search method in terms of completeness of search [62].

Greedy Search: Greedy search is a type of search method used in graph mining [14]. It is a heuristic search and a direct matching method. The Greedy search method is categorized into two search methods, depth-first search (DFS) and breadth-first search(BFS). DFS initially maps a vertex from a candidate subgraph G_s to a vertex belonging to the graphs in D. This mapping, f_s1, is searched using a mining measure. Then another vertex is chosen which is adjacent to the vertex mapped by f_s1 and an extended mapping f_s2 is found to map these two vertices to another two vertices in the graphs of graph set D. This process is repeated until the mapping f_sn may have no more extensions where n is the maximal depth of the search of a DFS branch. An advantage of using DFS is that it will save memory consumption. However, a drawback of this method is that only a certain number of the isomorphic subgraphs are found if the search is stopped due to constraints on search time as the search space may be very large [62].

In recent research, BFS is more commonly used as an approach for graph mining rather than DFS. One of the advantages of BFS over DFS is that for a specified size for the subgraphs, BFS can ensure obtaining all isomorphic subgraphs. However, in some applications the large size of search space does not fit in the memory [62]. To solve this difficulty the beam search method is used with BFS [14, 65]. In this method, the maximum number of branches for the BFS is set and the search progresses downwards. Any branches that do not fit the maximum number of branches are pruned. As pruning is applied to the search paths, the process of searching for the isomorphic subgraphs is done in tractable time, but the completeness of the search is lost [62].

Inductive Logic Programming: Inductive Logic Programming (ILP) is the intersection of inductive learning and logic programming. ILP uses methods from both machine learning and logic programming. It may be used as an unsupervised approach with a combination of other methods [8] or a supervised approach [34]. The goal of ILP, inherited from machine learning, is to induce hypotheses from observations and to produce new knowledge from experience. From logic programming, ILP inherits its representational formalism [47]. This representation, one of the main advantages of using ILP, may be given by *first order predicate logic* which gives the ability to derive knowledge from background knowledge. In ILP, given background knowledge B and evidence E where E is divided into positive evidence E^+ and negative evidence E^-, a hypothesis H is found such that the following

semantic conditions hold [62].

1. Posterior Satisfiability: $B \wedge H \wedge E^{-} \not\models \square$,
2. Posterior Sufficiency: $B \wedge H \models E^{+}$

where $\square$ is $false$, and therefore $\not\models \square$ means that the theory is satisfiable. A disadvantage of ILP is the very large size of the search space and computational intractability. The ILP method may be any of the heuristic, complete, direct and indirect search methods depending on the background knowledge used to control the search process. If control knowledge is used to prune search paths, the method is heuristic, otherwise it is complete. When ILP uses knowledge on predetermined subgraph patterns to match subgraph structures, the method is indirect as only subgraph patterns including the predetermined patterns are mined. Therefore the subgraph isomorphism problem is not strictly solved [62].

By using data mining, information is obtained from the recipes which can be used for learning. The process of larning is described in the next section.

2.4 Machine Learning

Machine learning, a branch of artificial intelligence, is concerned with programming computers to build a model using example information or past experiences with the aim of optimizing a performance criterion. This mathematical model is built by processing and learning large amount of data using efficient algorithms which optimize the parameters of the model. The trained model may be *predictive*, making predictions of data items in the future, and *descriptive* which will be able to provide knowledge about the data. [4]. Event learning produces a *descriptive* model which may be used to learn events within data. This type of learning is described in the next section.

2.4.1 Event Learning

Defined by Sridhar [53], an activity in a certain domain consists of semantically significant occurrences referred to as *events* which belong to a finite set of *event classes*. Therefore, an activity may be represented as a set of event classes. For instance, activities in the kitchen domain may be represented by event classes such as making egg sandwiches or hot drinks. An event class is composed of events where each event may consist of various entities in relation to each other. In an event class, the events are also in relation with each other based on their temporal ordering, previously mentioned in section 2.2. For example in this project the event class making an egg sandwich, given as a recipe, is composed of various subevents such as frying eggs or toasting slices of bread where entities such as eggs oil and bread are used.

Using the relations between the events, a graph is obtained to represent the structure of the event class referred to as an interaction graph [53, 54]. An example of an interaction graph based on spatio-temporal relationships is illustrated in Figure D.1 in Appendix D. A problem that may occur is that an event class may have various interaction graphs. This will cause difficulty in finding a suitable graph which represents the event class. For example, various recipes may exist for making coffee. In one recipe the milk might go before the sugar and in another the sugar may go first.

However, by using event learning this problem may be overcome. Event learning uses machine learning techniques to find common patterns in interaction graphs to obtain the most likely interpretation of an event class. It may find subgraphs within the interaction graphs that uniquely identify an event class. Event learning may also be used to find patterns among event classes to identify or distinguish certain activities, for example distinguishing between making sandwiches and baking a cake. A drawback of using event learning in this project is that it may have a high complexity since finding common subgraphs is the subgraph isomorphism problem mentioned in section 2.3.2.1.

An important aspect that influences learning in algorithms is the level of supervision. The two major types of learning in machine learning are supervised and unsupervised learning which are explained in the following sections.

2.4.2 Supervised Learning

In supervised learning, supervision is given by assigning an attribute to each training item which specifies the correct class of that item [38]. The goal of the learning process in supervised learning is to associate the training data with their corresponding classification. This association may be represented by a mapping in a model which can be used to predict the class of test data whose classification is unknown. The accuracy of the model may be evaluated by comparing the predictions of the test data against ground truth values [23]. Classification is one of the most frequently used tasks in supervised learning and is divided into Probabilistic, Non-Probabilistic and Functional Object Classification [36]. Two non-probabilistic classification methods that may be used in the project are described below.

K-Nearest-Neighbour: K-nearest-neighbour (KNN) is a classification method used to predict the class of a testing item based on its euclidean distance to training items. This method consists of three steps. In the first step the distance between the test item and all training item is calculated. In the second step the K nearest training items are found. The class that has the majority of votes from the training items is assigned as the class of the test item. A drawback of this method is that the optimal value for K must be found [57].

Support Vector Machine: The support vector machine (SVM) is a classification method which consists of two steps, training and testing. In the training step, the SVM builds a model by dividing

the data items in the feature space using a hyperplane. One side of the hyperplane belongs to one class and the other side belongs to the second class. In the testing step, the test item will be placed in the feature space and depending on the side of the hyperplane it is in, it will be classified accordingly [12]. A drawback of the SVM is that when using a feature set with a high number of dimensions, it may become computationally expensive [20].

To divide the data set into a training and test set for the above methods, N-fold cross validation may be used. This method divides the data set into N partitions and each iteration chooses one of these partitions as a test set and the other partitions as a training set. In each iteration the test set is chosen as a partition that has not yet been a test set. Leave-one-out Cross validation is a special case of the N-fold cross validation where N is the number of data items. This allows each item in the data set to be considered as a test item and its class predicted based on the model trained from all other items in the data set [40].

2.4.3 Unsupervised Learning

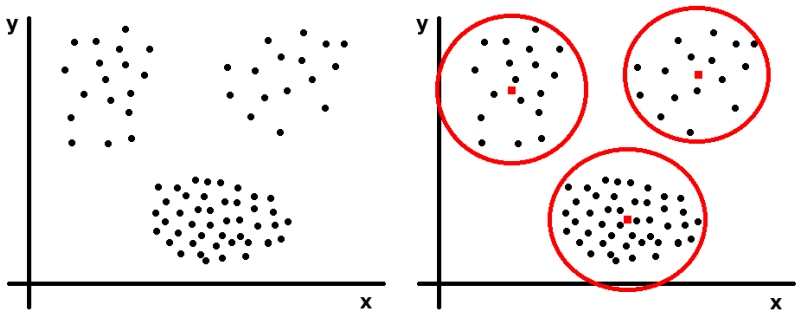
In unsupervised learning no supervision is given. This means that no attribute exists for the data items to provide their correct classification and a model is obtained by learning from input data only [38]. The aim of unsupervised learning is to build a representation from the entire dataset. For example, this representation may be a set of regions in the feature space, each containing the closest items in the data set with a certain distance measure [23]. This representation provides a structure for the input space which enables the ability to find certain patterns that have a high number of occurrences within the structure [4].

One of the most common techniques in unsupervised learning is clustering [4, 43]. The goal of clustering is to group items in a dataset into subsets or clusters based on the similarity of their attributes. Items in each cluster should be as similar as possible and all items in a clusters should be as dissimilar to items in other clusters. Figure 2.2(a) illustrates a two dimensional plot of items in a dataset with attributes x and y. Figure 2.2(b) illustrates the result of applying a clustering algorithm, such as K-means clustering which has been originally proposed by MacQueen [42], on the dataset in Figure 2.2(a).

The following are various types of clustering which may be used to solve unsupervised problems.

Flat Clustering: Flat clustering creates an unstructured, flat set of clusters where the clusters would not relate to any other cluster. An example of Flat clustering is the K-means algorithm which is the most commonly used algorithm for flat clustering, due to its simplicity and efficiency [43].

The K-means algorithm works by initially placing cluster centre points (seeds) randomly on the



(a) A plot of a two dimensional dataset (b) Clustering the dataset into 3 clusters

Figure 2.2: An example of the clustering process with dots showing data items, circular regions representing cluster regions with squares as cluster centre points.

input space. Then each item in input is assigned to their closest cluster centre point where closeness may be measured using euclidean distance or any other distance measurement. The cluster centre point is then moved to the average location of all items assigned to it. This process is repeated until the program converges or is stopped by a stopping criterion [13]. Drawbacks of K-means are that the search is prone to local minima and the number of clusters to be found must be given to it as input. One way to find a suitable value for number of clusters is to run the algorithm multiple times and find the most suitable value [43]. A more efficient way is to use X-means, a clustering algorithm developed by Pelleg [48] which overcomes the mentioned drawbacks and finds the most suitable value for the number of clusters automatically.

An alternative to using flat clustering is hierarchical clustering which is more informative than flat clustering as it provides a structure for the set of clusters found. An advantage of hierarchical clustering is that it does not require a pre-specified number of clusters, but the main disadvantage is the loss if efficiency.

Hard Clustering: Hard clustering is the hard assignment of items to clusters, resulting in each item in the dataset being a member of exactly one cluster. A drawback of this type of clustering is that if an item is similar to two clusters, it is only assigned to one cluster, disregarding its similarity to the other [43]. An alternative to this is the partition hierarchy clustering. Partition hierarchy clustering is similar to hard clustering in that a member of a cluster belongs to exactly one cluster. However, the member of the cluster is also a member of its parent cluster.

Soft Clustering: In soft clustering, also known as fuzzy clustering, each item is given a soft assignment to clusters. This provides a distribution over all clusters for each item. An example of soft clustering is illustrated in Figure D.1 in Appendix D. A suitable algorithm for soft clustering is the Expectation-Maximization (EM) algorithm. The EM algorithm, developed by Dempster et. al. [16],

is a generalization of the K-means algorithm which may be applied to various types of representations and distributions. It is an iterative method which alternates between two steps namely, the expectation (E) step and the maximization (M) step. The expectation step calculates the expectation of the computed log-likelihood using the estimated hidden variables, whereas in the maximization step, the parameters maximising the found expected log-likelihood in the E step is computed. The parameter estimates found in the M step are then used in the next E step and this cycle is continued until the algorithm converges [43].

One of the drawbacks of using EM algorithm in soft clustering is that EM algorithm may easily get stuck in a local optimum if the seeds are not chosen wisely. To overcome this problem, soft clustering may use both K-means and EM algorithm. This is achieved by using the K-means algorithm to find the initial assignment of items to clusters and giving this assignment to the EM algorithm to apply soft clustering [43].

Various libraries and tools exist which provide the ability to use the methods mentioned in this chapter. These tools are mentioned in the next section.

2.5 Tools & Libraries

A suitable toolkit to use in this project is the Natural Language Toolkit (NLTK) [10]. NLTK is an open source suite of programs and libraries widely used in linguistics and computer science. It covers statistical and symbolic natural language processing by providing modules for parsing, tagging, visualizing and text classification, developed for the python programming language [9].

Another language processing toolkit is the General Architecture for Text Engineering (GATE) developed by Cunningham [15]. It is an open source java suite which provides a variety of tools for NLP methods such as a parsing tool called SUPPLE. Plugins to use Weka and WordNet are also developed in GATE.

WordNet is a lexical reference system which may be used to provide synonyms and antonyms for words, produce a group of words semantically equivalent to a target word (referred to as a *synset*) or various other methods to obtain lexical or syntactic information about a word [45]

LibSVM developed by Chan et al. [11], is a library which provides the ability of using a support vector machine. This library is originally written in *C*, however, a MATLAB implementation of this library also exists. This library also provides a python program to calculate the optimal kernal parameters in an SVM for any data set.

Minimum-Redundancy Maximum-Relevance (MRMR) is a feature selection toolkit developed by by Peng et al. [49]. This toolkit uses the maximal statistical dependency criterion. This method is based on mutual information which measures the mutual dependency between two variables. By using this toolkit, the features which are most suitable in representing classes are extracted from a feature set. This toolkit may be used in this project to find patterns that distinguish between the classes.

2.6 REDVINE

Relational Description of Video Scenes (REDVINE), developed by Sridhar [53], was originally developed for learning event classes such as unloading or refuelling a plane in videos of airports. As this project had a limited amount of time, using the REDVINE programs for graph mining saved the implementation time significantly. Only a few of the procedures available in REDVINE were used in this project; of these, some were used unaltered, whereas others were significantly modified. A list of the programs obtained from REDVINE is available in Appendix B.

Chapter 3

Learning Relational Patterns

3.1 Introduction

Based on the aims and objectives provided in section 1.2, the following tasks were created to achieve the goals of the project:

1. Corpus Creation: A set of recipe corpora were created by extracting recipes from websites so that the framework may be applied on. Each corpus consisted of multiple classes and various number of recipes.
2. Pre-processing: This stage was required to process raw text to a form that facilitates the extraction of relations between verbs and nouns. This component uses four methods, namely dictionary lookup, resolving of missing nouns, substituting verbs and encoding to achieve this.
3. Graph Based Relational Representations: Relations between verbs and nouns are represented as a three layered graph. The three layers of the graph are namely the entity, relation and temporal layers. These intervals between the layers correspond to the relations between nouns and verbs, and verbs and temporal relations respectively.
4. Feature Representation: For categorisation and clustering, the three layered graphs are mined using level wise mining to extract subgraphs to be used as fetures in a feature set.
5. Feature selection: Feature selection was applied to obtain the most suitable features, in order to obtain subgraphs that ensure maximum discrimination between classes in the context of the supervised approach.
6. Learning: Both supervised and unsupervised learning was applied on the selected feature set for classification and clustering respectively.

To incorporate the above tasks, the framework illustrated in Figure 3.1 was used. This framework consists of 5 levels. Level 1 represents the creation of the corpus of recipes. Level 2 shows the pre-processing stage where data is extracted and encoded. In level 3, the graph mining stage is carried out, where the recipes are mined for patterns. Level 4 is the collection stage of common extracted patterns, providing a feature set. This feature set may be enhanced or modified. Finally, Level 5 represents the classification and clustering stage where accuracies and clusters are provided for evaluation.

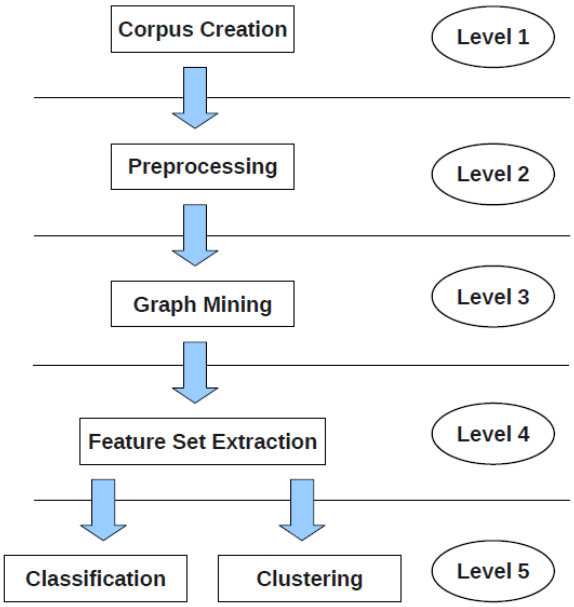


Figure 3.1: The system architecture

In the following sections, the methods and techniques used to achieve the above tasks are described. Moreover, the detail of building the components of the framework will be disclosed.

3.2 Corpus Creation

To investigate the benefits of using relational patterns, a domain consisting of a set of temporally sequenced actions was required. This largely occurs in text describing human activities. Manuals on building objects, daily schedules, techniques in martial arts and cooking instructions are all domains where instances in each contain a set of temporally ordered tasks to be completed by a human. It was found that the cooking domain was suitable for the purpose of this project.

To apply pattern learning, a corpus of recipes was built from this domain. To create this corpus, recipes belonging to multiple classes were extracted from cooking websites such as Deliaonline [52], Goodfood [58] and Allrecipes [3]. The mentioned websites were chosen as they contained various classes of recipes, where each class contained a large number of user rated recipe examples. The

corpus recipes are in different formats, varying in grammatical tense and number of lines. All classes consist of the same number of recipes. An example of an obtained raw recipe is provided below. For the rest of the chapter, this example recipe is used to explain the theoretical framework.

Prawn and Noodle Stir-fry:
Cook the noodles according to the packet instructions and drain. Heat 1tbsp of the sesame oil in a wok on a high setting and add the chilli, garlic and ginger. Cook for 30 seconds before adding the prawns. Stir-fry the prawns until they are pink and then remove them and keep to one side.
Add the remaining 1tbsp of oil to the wok over a high heat, and add the red pepper, peas and spring onion. Stir-fry for a few minutes, until the vegetables are starting to go tender but retain some crunch. Add the rice wine or sherry and soy sauce. Turn the vegetables in the wok, cooking for a minute, return the prawns to the pan and add the cooked noodles. Toss around until the noodles are heated through. Scatter on the sesame seeds and serve straight away.

3.3 Pre-processing

The pre-processing stage was applied on recipes in the corpus with the purpose of transforming the text into a form which facilitates the extraction of relationships between entities (nouns) and relations (verbs). This stage consists of the four steps described below. Each step is illustrated in Figure 3.2 with an example obtained by applying pre-processing on the example recipe provided above.

1) Dictionary Lookup
As text in natural language contains irrelevant and extra information for our task, using an accurate method to extract relevant entities and relations is needed. As POS taggers were observed to have inaccuracies in finding nouns and verbs in recipes, a dictionary lookup technique was used to perform the task of noun and verb extraction from text. A dictionary of verbs and a dictionary of nouns in the domain of cooking was used. Each dictionary contains cooking words with the grammatical position that the dictionary represented. A subset of each dictionary is provided in section E.5 of Appendix E. To apply the dictionary lookup step, each word in the recipe was compared against all items in the two dictionaries, and if a match was found, the corresponding dictionary provided the grammatical position of the word. If a match was not found the word was ignored. This enabled the lookup of grammatical position of words in recipes to be highly accurate which allowed more patterns to be extracted.

The example for this step illustrated in Figure 3.2 represents the pre-processed recipe, provided in two columns. This recipe was obtained by using the dictionary lookup on the raw recipe provided in section 3.2 by finding the grammatical position of words. Each instruction in the pre-processed recipe is given as a verb followed by various number of nouns.

2) Resolving Missing Nouns

To make linguistic description less repretitive, nouns are often implicitly referred to by pronouns. While humans have the ability to infer the noun, the problem of mining nouns is a problem for natural language processing. For example, instructions exist such as *put them in the pan* where the verb *put* does not have any nouns to be given as entities. This occurs as in the recipe text, the preposition *them* is used to represent the nouns in previous instructions. To find the missing nouns, a method such as *anaphora resolution* [64] may be used. This method initially finds pronouns or prepositions, referred to as an *anaphor*, to find the missing noun, referred to as *antecedent*. This is done by using an incremental learning technique where a knowledge system is built from the anaphors and antecedents.

In this project a simple approach was taken to find missing nouns and to aid in fully representing the meaning of an instruction with verb and noun extraction. This method uses the NLTK POS tagger to find words which are prepositions. Since the preposition *them* in the example *put them in the pan* referred to nouns in a previous instruction, it could indicate missing nouns. Therefore, if nouns did not exist after a verb in an instruction, however a preposition existed, all nouns in the previous instruction were assigned to the verb. Although this method is simple, it was found to provide a reasonable solution to address the problem of mining nouns in simple recipes.

The modification of the pre-processed recipe obtained by dictionary lookup, as a result of applying POS tagging and using the proposed method is highlighted in bold in Figure 3.2. The modification was done for the verb *remove* in the raw recipe instruction *then remove them and keep to one side*, due to the verb not having any nouns. However, as the preposition *them* occurs in the instruction, the noun *prawns* of the previous instruction in the pre-processed recipe is given to the verb *remove*. The final modification applied to the preprocessed recipe is verb substitution.

3) Verb Substitution

As the number of verbs in the cooking domain can be very large, patterns created from the verbs will be less common. To increase the similarity of patterns, verbs were substituted by a common synonym verb and stemming was applied in this step. In the example provided for this step in Figure 3.2, verbs *scatter* and *adding* have been substituted by the verbs *sprinkle* and *add* respectively. Verb substitution has three main advantages. Firstly, it increases the occurrence of verbs among various preprocessed recipes, secondly, the substituted verbs will have no occurrence, removing them from the verb feature space which leads to reducing the dimensionality. A third advantage is that it eliminates the need to use stemming on verbs which can lead to inaccuracies. For example verbs such as *adding* and *added* may be replaced by the verb *add*.

4) Encoding

Encoding is the final pre-processing step to convert text into a form that explicitly represents the no-

tion of episodes in [54]. An episode in this project represents a set of nouns, the verbs relating them and finally a temporal artificial interval during where these verbs may hold. Example of an encoded episode is illustrated in Figure 3.2. The colums before the value *999* represent the nouns, the two colums after this value represent the artificial temporal intervals and the final column represents the verb.

While these temporal intervals can be inferred from cooking videos, this information is usually not available from text. However, to be able to represent temporal relationships, the temporal intervals are artificially introduced. In principle it is possible to represent the temporal structures of a recipe with reasonable precision using words such as *while*, where by using all of Allen temporal relations to represent the complete temporal structure, however, a more simpler approach was taken. A recipe is represented as a sequence of episodes using the unique IDs of words in the dictionaries, Thus, only the two Allen relations *meet* and *before* are applicable in this representation.

3.4 Graph Based Relational Representations

Sridhar [53] uses a graph representation, referred to as an interaction graph, where it is based on the relational properties of events. For the purpose of demonstration, the interaction graph of only the highlighted gray region of the encoded recipe provided in Figure 3.2 is illustrated in Figure 3.3(a). This graph representation consists of three layers:

Entity Layer The entity layer is represented as the bottom layer in an interaction graph. Each node in this layer corresponds to a noun from an encoded recipe. For example, the nine unique nouns used in the gray highlighted recipe in Figure 3.2 are represented as nine nodes in the interaction graph in Figure 3.3(a) in the entity layer.

Relation Layer The relation between verbs is represented as the middle layer of an interaction graph. Nodes in this layer correspond to verbs in a recipe. Each node is connected to two or more nodes in the entity layer by edges in the graph. This connection represents the relation of a verb with multiple nouns as given by an episode. Each relation represents an instruction in a recipe. The four verbs illustrated in the gray highlighted area in Figure 3.2, are represented as four nodes in the relation layer where a verb is connected to the nouns each has in its instruction.

Temporal Layer The temporal relation is represented by the top layer in an interaction graph. A node in this layer represents the temporal relations between the intervals associated to pairs of verbs that are represented by second layer nodes. Each node in the temporal layer connects all possible paired combinations of verbs, which are represented by Allen relations.

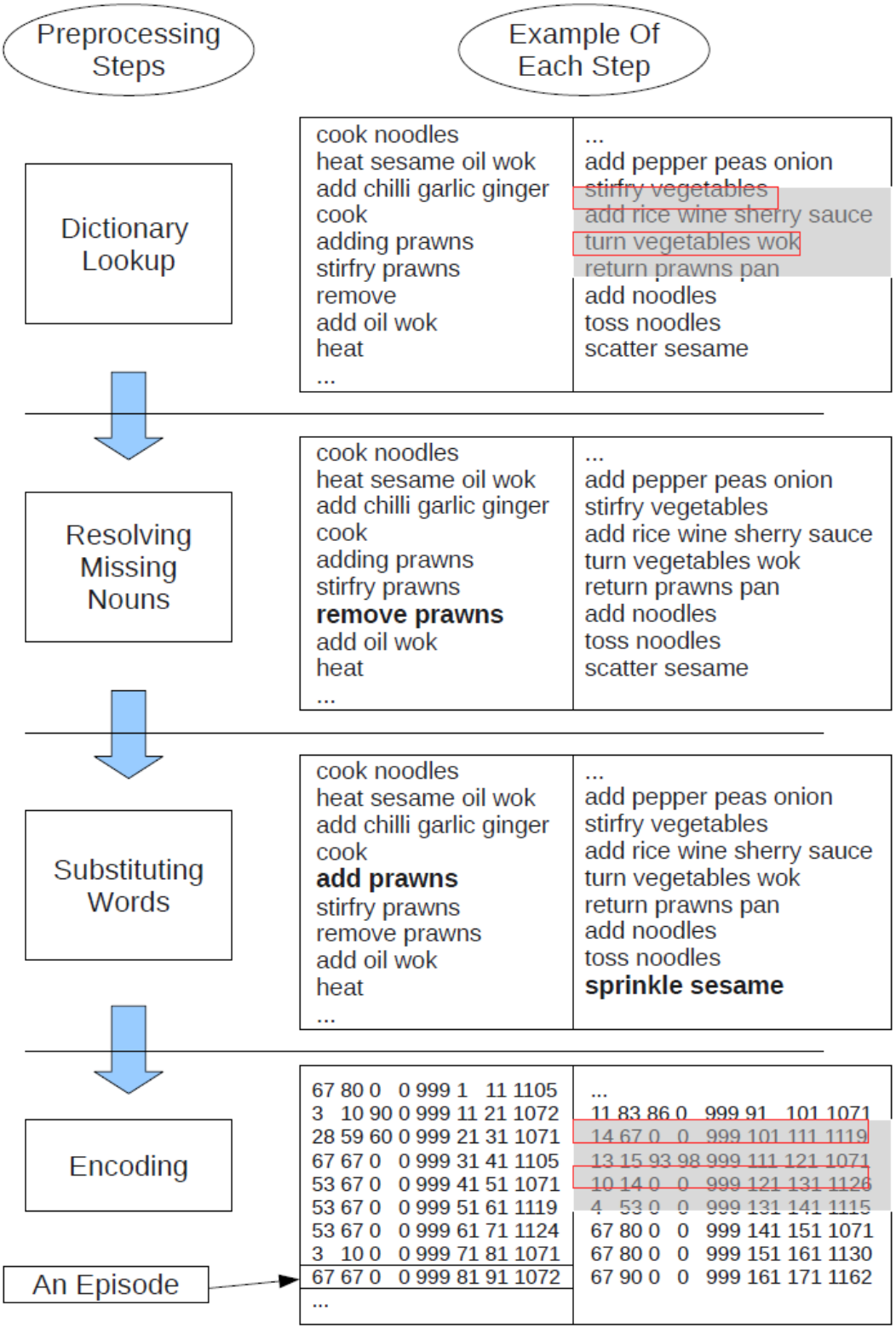
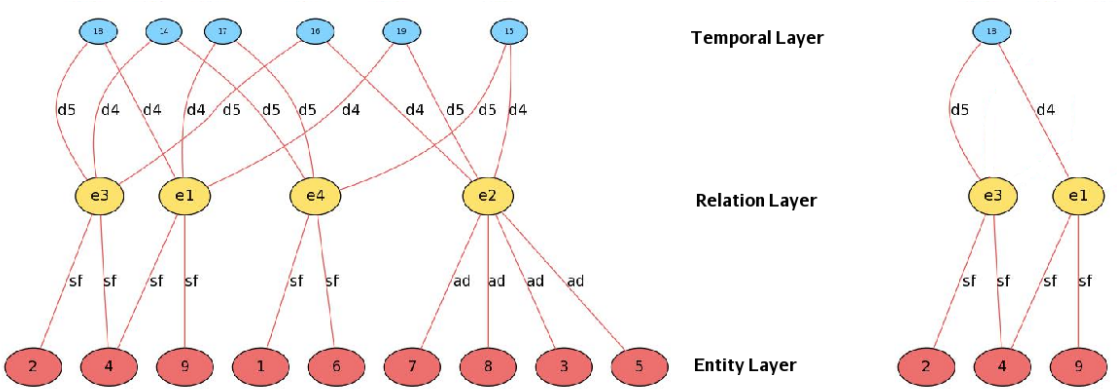


Figure 3.2: The four steps of pre-processing along with an example for each step using the recipe provided in section 3.2



(a) Part of the interaction graph made from the encoded recipe obtained in 3.3

(b) A permitted subgraph in Figure 3.3(a) to be used as a feature

Figure 3.3: An interaction graph obtained from a part of the encoded recipe which is highlighted in gray in Figure 3.2 along with an existing subgraph

3.5 Feature Representation

To apply learning on a set of interaction graphs, a suitable similarity measure is needed to compare and distinguish the graphs. Deshpande et al. [18] use a similarity measure based on a feature representation using a bag of subgraphs approach. To obtain this, initially graph mining using a greedy approach is used to obtain a set of frequent subgraphs occurring in training examples. Each example is then represented as a feature vector where the size of this vector is determined by the total number of subgraphs mined. The value of each feature in this feature representation is the frequency of the occurrence of the mined subgraph in the example. However, the approach used by Deshpande obtains all possible subgraphs in the training examples, whereas for the interaction graphs only a set of semantically meaningful subgraphs is of interest. Therefore, the bag of subgraphs approach will be used as a feature representation, as illustrated in Figure 3.4 produced from the interaction graph in Figure 3.3(a). Each subgraph, also referred to as a pattern, is required to have the following two requirements to be chosen as a feature.

Firstly, a pattern must be a 3 level subgraph, obtained from a recipe interaction graph, with a minimum of two relation nodes having at least one common entity node. Secondly, a pattern must frequently occur in a class of recipes to be used in the feature representation. In order to produce subgraphs of all interaction graphs in the corpus, graph mining is applied by using a greedy breadth-first search method.

By using level-wise mining each class of recipes is graph mined and the occurrence of all subgraphs in the class of recipes is calculated. If this value is larger than a minimum frequency, it will be

chosen in the feature representation, otherwise it will not be considered. The only permitted subgraph of Figure 3.3(a) is illustrated in Figure 3.3(b), where the two verbs *stirfry* and *turn* at the relation layer only share one noun *vegetables* in the entity layer. The two instructions containing each of these two verbs are shown in red rectangles in Figure 3.2.

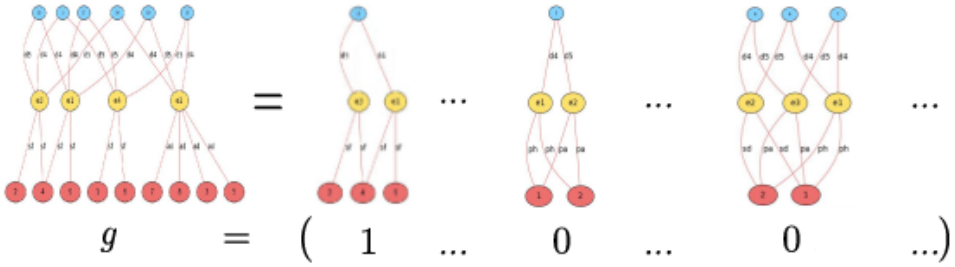


Figure 3.4: The feature representation for the interaction graph illustrated in Figure 3.3(a)

By obtaining all patterns that meet the two mentioned criteria, the feature representation for all recipes is obtained and placed in a feature set. This feature set is enhanced using a feature selection method to improve learning as described below.

3.6 Feature Selection

Since not all features in the feature set are useful to use, a feature selection method must be applied to provide the most suitable features for learning. The goal of feature selection is to obtain a feature set S with features x_i, where all features jointly have the largest dependancy on a target class c. This results in all x_i features in the feature set S to be very similar allowing them to disinctly represent the class C. Given two random variables x and y, mutual information is used as a measure to calculate the dependency between the random variables, using Equation 3.1, in terms of their probabilistic density functions $p(x)$, $p(y)$ and $p(x,y)$.

$$I(x;y) = \int\int p(x,y)\log\frac{p(x,y)}{p(x)P(y)}dxdy. \tag{3.1}$$

Items in the set of selected features x_i, are each required to obtain the highest mutual information $I(x_i;c)$ representing the largest dependency on target class c [49]. This scheme is called maximum-dependancy and is obtained using equation 3.2

$$max\, D(S,c), D = I(\{x_i, i = 1,...,m\};c). \tag{3.2}$$

As maximum-dependancy was hard to implement, the minimum-redundancy maximum-relevancy (MRMR) feature selection method developed by Peng et al. [49] which uses an alternative method, was used for feature selection. This method consists of the following two stages.

Max-Relevance Max-relevance is the *maximal relevance* criterion which is an alternative to using maximum-dependancy. This criteria approximates $D(S,c)$ in equation 3.2 by searching for features using the mean of all mutual information values between feature x_i and class c, satisfying equation 3.3.

$$max\, D(S,c), D = \frac{1}{|\, S\,|} \sum_{x_i \in S} I(x_i;c). \tag{3.3}$$

Min-Redundancy By using max-relevance it is likely that among the obtained features a high dependency may exist. By removing one of the two features that have a high dependency, the discriminative-class power of the remaining feature set will not significantly change. Therefore, by using the *minimal redundancy* criteria provided in equation 3.4, mutually exclusive features are chosen.

$$min\, R(S), R = \frac{1}{|\, S\,|}^{2} \sum_{x_i, x_j \in S} I(x_i;x_j). \tag{3.4}$$

By defining the operator $\phi(D,R)$ to combine the criterion Max-Relevance D and Min-Redundancy R, the minimum-redundancy maximum-relevancy (MRMR) is obtained by equation 3.5

$$max\, \phi(D,R), \phi = D - R \tag{3.5}$$

By applying the MRMR method on the obtained feature set in section 3.5, the most suitable subset of features in the feature set is obtained which are the most suitable features for distinguishing between classes. This subset of features is the new feature set to be used for learning, described in the next section.

3.7 Learning

The learning process is used to obtain a model from training instances which can be used to predict test items. A supervised approach was used as part of the learning process to investigate the properties of patterns and evaluate on their different types. By also using an unsupervised approach the ability to evaluate the framework as a whole was provided. The learning process was applied by using two supervised methods, namely K-nearest-neighbour (KNN) and the support vector machine (SVM) and an unsupervised clustering method X-means.

K-Nearest-Neighbour

By using a supervised learning approach, unknown items may be classified using instances in the feature set. Evaluating on the results will provide beneficial feedback on the suitability of the features used in the feature set. K-Nearest-Neighbour classification is an instance based learning method where it classifies items based on neighbouring items. To classify a point Q, it initially finds the

nearest k neighbours of the point by computing the distance between point Q and all points in the dataset $S = \{X_1, X_2, ..., X_n\}$, giving a total of n distances. The squared euclidean distance measure, provided in equation 3.6, is used to calculate the distance between point $Q = [q_1, q_2, ..., q_d]^T$ with d dimensions and a data point $X_i = [x_i1, x_i2, ..., x_id]^T$ in S. The k closest points to Q are its k nearest neighbours. By using the ground truth labels for the k nearest neighbours, the label with the highest frequency is the predicted label for point Q [41].

$$D(X_i, Q) = \sum_{j=1}^{d} (x_{ij} - q_j)^2. \tag{3.6}$$

Support Vector Machine

Another approach to supervised learning is by using a method based on regression. The support vector machine (SVM) is a regression based classification method. The goal of a SVM is to train on data examples and learn a model based on separating the example classes. The model may be used to predict the class of test examples. Due to the existence of multiple classes in the recipe corpora, a multi-class SVM was chosen with a one-vs-one decomposition scheme. This scheme was chosen because when compared to the one-vs-all scheme, it has a shorter training time and there is no significant difference in accuracy [29]. As a multi-class SVM is based on the binary SVM classification, the binary SVM is initially described followed by the multi-class SVM using a one-vs-one decomposition scheme.

• **Binary SVM Classification** In a binary SVM classification, the input vector x is mapped onto a high dimensional feature space using the mapping function $\phi(x)$. This is done to improve linear separability. Each item in the training set with size M is given as an output-input pair $(x_i, y(x_i))$, $i = 1, ..., M$ where $y(x_i) = 1$ if x_i belongs to class 1 and it belongs to class 2 if $y(x_i) = -1$. If the feature space is linearly separable for the training set, Equation 3.7 is used as the decision function [26, 33]:

$$f(x) = w^T \phi(x) + b, \tag{3.7}$$

where $y(x_i) f(x_i) > 0$ for $i = 1, ..., M$, w is a weight vector and b is a bias term. For an unknown test item x, if $f(x) \geq 0$ it is classified as class 1 and if $f(x) \leq 0$ it is classified as class 2. The plane which divides the two classes using Equation 3.7 is referred to as a hyperplane. The distance between the hyperplane and the nearest training data is called the margin. The hyperplane that separates the two classes with a maximum margin is referred to as the optimal separating hyperplane [33].

If the classification problem in the feature space is not linearly separable, the optimal separating hyperspace can be found by solving the following optimization problem [33]:

$$Minimize\ Q(w,\zeta) = \frac{1}{2} \parallel w \parallel^2 + C\sum_{i=1}^{M} \zeta_i, \tag{3.8}$$

$$subject\ to\ y(x_i)(w^T \phi(x_i) + b) \geq 1 - \zeta_i, for\ i = 1,...M, \tag{3.9}$$

where ζ_i is the slack variable for x_i and C, referred to as the regularization parameter, determines the tradeoff between classification error and maximization of the margin. By introducing the Lagrange multipliers α_i, the following dual problem is obtained [33]:

$$Maximize\ Q(\alpha) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y(x_i) y(x_j) K(x_i, x_j), \tag{3.10}$$

$$subject\ to \sum_{i=1}^{M} y(x_i)\alpha_i = 0,\ 0 \leq \alpha_i \leq C. \tag{3.11}$$

In the solution obtained by Equations 3.10 and 3.11, if $\alpha_i > 0$, x_i are called suport vectors, if $\alpha_i = C$, bounded support vectors and if $0 < \alpha_i < C$, unbounded support vectors. One of the advantages of a SVM is that if all non-support vectors are removed from the training set, the same solutions are obtained. The decision function to predict unknown items, using the support vectors, is given by [33]:

$$f(x) = \sum_{i \in S} \alpha_i y(x_i) K(x_i, x_j) + b, \tag{3.12}$$

In Equations 3.10 and 3.12, $K(x_i, x_j)$ is a kernel function given by [33]:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j). \tag{3.13}$$

A kernel that may be used in a SVM is the radial basis function (RBF) kernel and is obtained by [33]:

$$K(x_i, x_j) = exp(-\lambda \parallel x_i - x_j \parallel^2), \tag{3.14}$$

where λ is a positive parameter for slope control.

• **Multi-class SVM Classification** The multi-class SVM classification with a one-vs-one scheme for N classes, initially divides the multi-class problem into $N(N-1)/2$ binary classification problems. In the learning phase of each of the binary problems, a subset of the training data is given which only include the class labels being used in the binary classification [21].

When the binary SVM classification problems are solved using the methods previous outlined,

thier models are stored. The test phase of the multi-class SVM classification problem is then initiated. Each test vector is given to each of the binary classifiers. The prediction of the the binary classifier is given by $r_{ij} \in [0,1]$, which represents the confidence of the binary classifier between classes i, j in favour of the former class. The outputs of all binary classifiers, for a test vector, are represented in a score matrix R, illustrated in Figure 3.5. For the multi-class SVM to output the predicted classification of the text vector, a voting strategy based on the prediction of all binary classifiers provides is applied. The predicted class of the test vector is the class with the highest frequency [21].

$$R = \begin{pmatrix} - & r_{12} & \cdots & r_{1m} \\ r_{21} & - & \cdots & r_{2m} \\ \vdots & & & \vdots \\ r_{m1} & r_{m2} & \cdots & - \end{pmatrix}$$

Figure 3.5: Score matrix R produced by the predictions of binary SVMs in a multi-class SVM [21]

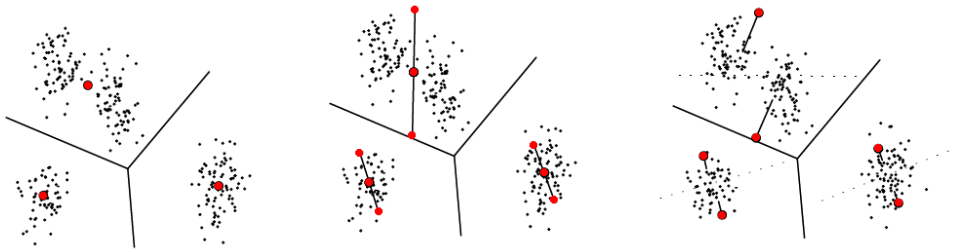
X-Means

Classifying data using an unsupervised approach provides detail on the classification of the feature set as a whole, rather than subsets of the feature set in comparison with supervised learning. X-means is based on the unsupervised K-means clustering algorithm which provides the ability to group items in a data set based on their position in the feature space. X-means was chosen as it overcomes the first two of the three disadvantages of K-means which are firstly, the number of clusters to be found must be given to K-means as input. Secondly, the clustering of data is prone to local minima. Thirdly, it computationally performs poorly [48]. The clustering process of a naive K-means algorithm is described below followed by describing the improvements made in the X-means algorithm in overcoming the drawbacks of K-means.

K-means divides the data points in a feature set into K subsets where all points in a subset belong to some centre point also referred to as a centroid. The algorithm stores the positions of all centroids, which are initially set to random values, and runs in iterations. The program stops if the position of the centroids do not change after an iteration or a stopping criteria, such as a maximum number of iterations, is reached. The following two tasks are performed in each iteration of the algorithm [48]. (i)For each point x in the feature set, find the distance between this point and all the centroids. Assign point x to the closest centroid. (ii) After all points have been assigned to a centroid, re-estimate the position of each centroid by assigning it to the average position of all data points associated with it.

To avoid the need of inputting the value K, X-means searches for the most suitable K in range specified by the user which also avoids a local minima. This is done by iteratively applying 2 steps, namely the *improving parameters* and the *improving structure* steps. X-means stops these steps and provides the scoring model if K become larger than the range provided by the user [48].

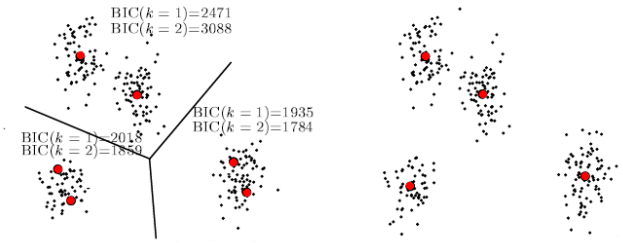
In the *improving parameters* step, the K-means algorithm is run until convergence (Figure 3.6(a)). After convergence, the *improving structure* step decides if new centroids should be added and if so, it must provide a position where they should be placed. This step starts by splitting each centroid into two children centroids (Figure 3.6(b)). Their positions are given with by using a distance proportional to the region of points covered by their parent centroid, in opposite directions along a randomly chosen vector. Within the region of each parent centroid, a local K-means (with K=2) is run for each pair of children centroids. The first step of the K-means algorithm is illustrated in Figure 3.6(c), and the re-estimated centroid positions after convergence are given in Figure 3.6(d) [48].



(a) The result of applying K-means with K=3

(b) Splitting each original centroid into 2 children

(c) First step of applying the local K-means using the children centroids



(d) Converge local K-means

(e) Final results of X-means where by applying BIC suitable children centroids replaced their parent centroids or were not considered

Figure 3.6: The steps of the X-means clustering process using an initial 3 centroids.

The next step is to perform a test in order to decide whether the two produced children centroids model real structures or the parent node was suitable to model the distribution by itself. The result of this test will lead to either the parent centroid or the children centroids being removed (Figure 3.6(e)). To apply this test, the Bayesian Information Criteria (BIC) is used. This Criterion is used for model selection among a finite set of models where in this case, a model is needed for the children nodes and the parent node. For data D and a family of models M_j, the BIC of each model is scored. To find the BIC for the children nodes and the parent node, the following equation is used [48, 32]:

$$BIC(M_j) = \hat{l}_j(D) - \frac{P_j}{2} \cdot \log R. \tag{3.15}$$

Chapter 4

Experiments & Evaluation

4.1 Introduction

The purpose of this chapter is to evaluate the framework described in Chapter 3 by providing experiments which compare methods and approaches used in the components of the framework. These experiments are divided into natural language processing experiments, knowledge representation experiments and machine learning experiments. By evaluating the comparisons, the most suitable methods to be implemented in the final framework are found. Moreover, this chapter provides an understanding behind the reasoning of approaches taken to overcome the challenges that were met and present the contributions and findings of this project. In the following sections, each experiment is given by initially describing the problem that the experiment is to evaluate, a description of how the experiment is set, the results that are obtained by performing the experiment and finally the conclusion of the experiment.

4.2 Evaluation Strategy & Baseline

To evaluate different methods in the framework, the evaluation strategy used in this section is to obtain accuracies for the framework as a whole including the method to be evaluated and compare it with the framework without including the method. To evaluate the entire framework a baseline method is needed. The bag of words method using only verbs as features was chosen as this baseline since it resembles the system as it also only creates pattern from verbs. To implement the baseline method, a vector was outputted where the vector's columns represent the verbs in the dictionary and values

in the vector represented the number of occurrences of each verb in the recipe. Created vectors were combined into a feature set and used in the same supervised and unsupervised learning methods that were also used by the system.

4.3 Corpus Creation

To run each prototype and evaluate on its performance, a corpus was needed. The corpora provided in Table 4.1 were created by using the method outlined in section 3.2. The information in this table details the name in which a corpus is referred to in the experiments, the class names it consists of, the number of recipes in each class and the total corpus size.

	Name	Classes	Recipes per Class	Corpus Size
Corpus 1	Small Corpus	fry, hot-drinks	6	12
Corpus 2	Medium Corpus	bake fry	10	20
Corpus 3	Large Corpus	bake fry	30	60
Corpus 5	Unrestricted Corpus	bake, fry, pastry, roast	60	240

Table 4.1: The corpora used during the implementation of the prototypes

The small and medium corpora contain manually preprocessed artificial recipes. Each of these recipes were found from web pages and were manually modified to the format of the preprocessed example provided in Figure 3.2. The large corpus was created from real recipes, extracted from websites without any modifications being applied to them and was used for improving the prototypes.

The recipes in this corpus were chosen based on the the number of instructions that they may have, since a longer recipe may provide more patterns. It must be noted that not all instructions in the chosen recipes provided suitable patterns as many of them contained noise, such as cooking advice, or were irrelevant to cooking. This ensured that recipes with few or no patterns also existed in the corpus. Recipes that consisted of too many occurrences of instructions which contained verbs without nouns such as *mix everything* or *stir until smooth* were not considered in the corpora. As recipes can become very complex for text mining, this threshold of choosing recipes was applied to create an initial version of the preprocessing algorithm. Experimentation of the framework using each of the provided corpora is given in section 4.6.1.

The unrestricted corpus consists of recipes from websites, chosen without any restrictions or limits. It consists of more classes and recipes compared to previous corpora. This corpus was made to be used for a final evaluation of the system, provided in section 4.7. However, as applying the framework required a very long time on this corpus, all experiments until section 4.7, unless mentioned otherwise, (i) will be applied on the large corpus, (ii) applying by the framework which uses patterns that are created from a minimum of one common noun (iii) with the same parameters, (iv) where the class and overall accuracies are obtained by using the KNN method.

In the next section, various methods used in preprocessing are compared by applying them on the corpora outlined above.

4.4 Natural Language Processing Experiments

Outlined in section 3.1, after corpus creation, preprocessing is the next system component to be applied. This section provides experiments in order to evaluate the approaches taken in the preprocessing stage. These experiments are the following:

4.4.1 Experiment 1: Using Relations Based on Entities

One of the first challenges met in the implementation of the preprocessing stage was the case in which a verb had less than two nouns in its instruction. For example in the instruction *peel the banana*, the verb *peel* only has the noun *banana*. This was problematic as the system required each relation (verb) to have two entities (nouns). For cases where only one noun exists, an *empty* object is given by a keyword "empty" to the instruction. Therefore the example given will be changed to *peel the banana empty*.

The case where a verb has no nouns, but, a preposition is found in its instruction it discussed in section 4.4.2.2. If no nouns or prepositions exist in the instruction that the verb was in, the episode could possibly be ignored. However, initial explorations indicate that removing such verbs/episodes can remove significant information present in a recipe. Therefore, such verbs were assigned with two *empty* keywords and stored as an episode described in section 3.3.

The following experiment compares the three approaches of, (i) not using empty objects, (ii) using one empty object in cases where only one noun is found and third and (iii) using two empty objects where no nouns are found. The evaluation of the three methods is made by using the evaluation strategy previously described using a KNN classification approach.

Results

In Figure 4.1 the class and overall accuracies of the three methods given for the lack of nouns is provided. From the results, an increase in accuracy is observed as more empty objects are used.

Conclusions

From the results illustrated in Figure 4.1, it can be concluded that by using the method of adding empty objects, the system obtains higher accuracies where the tow empty object approach is most suitable. By using the empty object more verbs are stored which leads to more patterns being created.

Figure 4.1: Results on using the empty object approach to verbs without nouns.

4.4.2 Experiments 2 & 3: Using Domain Knowledge

During the implementation of prototypes, two different approaches to further increase the accuracies of classification in the preprocessing stage were considered. Firstly, expanding the domain knowledge which is described in this section. Secondly, increasing the number of recipes in the corpus which is described in section 4.4.3. Using examples of patterns in recipes, the basis for choosing the proposed methods will be explained.

4.4.2.1 Substituting Verbs

One of the methods considered early in this project was to improve the accuracy by substituting synonyms of verbs. This method, also described in section 3.3, takes advantage of verbs that have similar meanings by substituting one for another to increase the likelihood of finding common patterns. For the three reasons given in section 3.3 in which substituting verbs is beneficial, the following generalisation in which this method in theory may benefit the system is given.

Let's assume the patterns in Table 4.2 each exist in only the recipe that they are provided in. *Recipe A* has been correctly classified due to its pattern also existing in other recipes that are not shown in Table 4.2. *Recipes B* and *C* have been misclassified as their patterns have not been found in any other recipe. Although each of the three recipes contain different patterns, their patterns have the same meaning since the verbs *add, put* and *place* have the same word sense. The verbs that have similar meanings, in this case (*add, put, place*) can be substituted by a verb that can represent them, for example *add*. By applying this substitution, the patterns in Table 4.2 will be the same for all recipes and since the pattern in *Recipe A* was the reason it was correctly classified, *Recipe B* and *Recipe C* will also be correctly classified.

Although in theory this may work, in practice it may cause difficulties. For example, a verb which is unique for a class may be replaced by another, generalizing the pattern among more classes.

Recipe A	
add eggs sugar bowl	19 40 54 0 999 21 31 1071
whisk eggs sugar bowl	19 40 54 0 999 51 61 1087

Recipe B	
put eggs sugar bowl	19 40 54 0 999 41 51 1076
whisk eggs sugar bowl	19 40 54 0 999 81 91 1087

Recipe C	
place eggs sugar bowl	19 40 54 0 999 11 21 1099
whisk eggs sugar bowl	19 40 54 0 999 41 51 1087

Table 4.2: Examples of patterns in recipes

Initially the system was applied by substituting all synonyms of a verb where the system performed very poorly. Therefore it was concluded that if substituting verbs is beneficial, it is only when applying substitution for certain verbs. Therefore, in this experiment the effects of substituting a verb by another verb is presented compared to the baseline of not substituting verbs. The system which is used in this experiment is the system that was concluded the most suitable in section 4.4.1 which was using empty objects. The mentioned system was applied on 4 cases. (i) On a preprocessing method that does not use the substituted verbs, (ii) a preprocessing method where the verb *combine* is substituted by the verb *mix*, (iii) where the verb *place* is substituted by the verb *add* and (iv), where the verb *put* is substituted by the verb *add*.

Results

In Figure 4.2, the results of each case using the overall and class accuracies is presented. In this figure, each row represents a case where the first verb is the target verb and the second verb is the substitute verb.

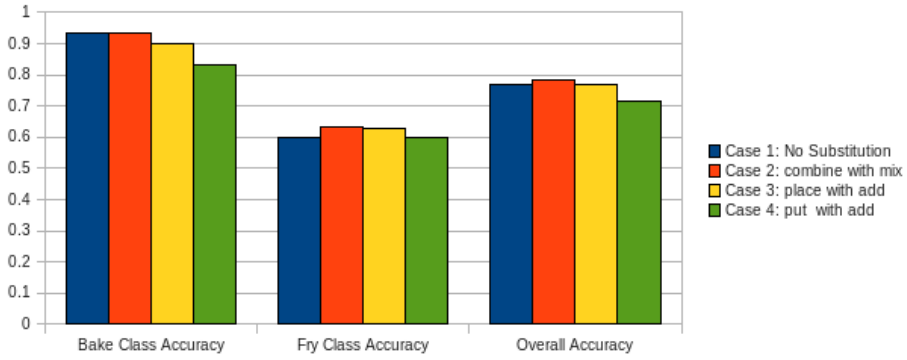


Figure 4.2: Results on substituting verbs.

From the results it can be observed that *Case 2* has improved *Case 1* by increasing the accuracy

of the *fry* class where the *bake* class remains the same. This indicates that by replacing the verb *combine* with the verb *mix*, a higher accuracy is obtained. In *Case 3*, the accuracy of the *fry* class has increased while the accuracy of the *bake* class has decreased. In *Case 4* the accuracy of the *bake* class is significantly reduced while the *fry* class remains the same.

Conclusions

From the results provided, it can be concluded that although the method of substituting verbs may increase the accuracy in certain cases, it may also reduce them. This depends on the verbs that are substituted which may or may not have the same effect on different corpora. In expert hands this method may prove to be useful. As finding these verbs are out of the scope of the project, finding which verbs are beneficial for substituting is set for future work. In future experiments, verb substitution will only be used to avoid using stemming where different verbs are replaced by its most simple form.

4.4.2.2 Prepositions

Described in section 3.3, by finding prepositions in instruction, nouns from a previous instructions are given to verbs that do not have nouns. To evaluate if this process is beneficial for the framework, the overall and class accuracies of using and not using this method will be compared.

Results

Figure 4.3 illustrates the comparison of considering and not considering the preposition method.

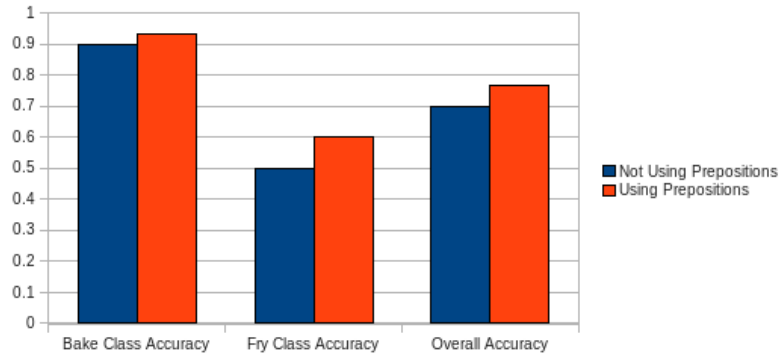


Figure 4.3: Comparison of using the preposition method and not using it.

Conclusions

From the results provided in Figure 4.3 it can be concluded that by using the preposition method the framework obtains a higher accuracy. Therefore, this method will be used in all experiments and the final framework.

From the results obtained in section 4.4.2, it can be concluded that by using domain knowledge the system may be improved to find more suitable patterns which leads to increased accuracy. The next method predicted to improve the system is described in the next section.

4.4.3 Experiment 4: Increase In Corpus Size

The second method predicted to improve accuracies in classification is to increase the number of recipes in the corpus. This was thought to increase the occurrence of patterns in the corpus. A pattern is stored and used in the feature set if it occurs in more than one recipe. However, if it only exists in one recipe it will not be considered for the feature set. Let's assume that in a corpus, *Corpus 1*, *Recipe C* in Table 4.2 exists where the only pattern that this recipe has is the one provided in its table. Since this pattern does not occur in any other recipe in *Corpus 1*, *Recipe C* is misclassified. Then another corpus is created, *Corpus 2*, where it consists of all recipes in *Corpus 1* and in addition it consists of *Recipe D* which has the same pattern of *Recipe C*. By increasing the corpus size, the pattern in *Recipe C* occurs in *Recipe D*, allowing it to be stored in the feature set. This will provide a distinction between *Recipe C* and other recipes belonging to another class. Moreover, if *Recipe D* contains another pattern that is distinct to its class, the feature vector of *Recipe C* will be closer in distance to other recipes in its own class in the KNN classification, providing it with a higher probability of being correctly classified compared to when it had no patterns of representing it.

The experiment set to prove that by adding additional recipes to the corpus, misclassified items may be correctly classified, initially the framework is applied on the large corpus. By choosing a misclassified item, a similar recipe which contains the same patterns of the misclassified item is added to the corpus. After running the initial experiment, two misclassified items where chosen, one from the *bake* class and another from the *fry* class. After choosing a pattern from each of these recipes, two additional recipes containing the same pattern were extracted from websites and placed in the corpus. The preprocessed form of the misclassified and additional recipes are presented in Appendix E in section E.4 where the common patterns are shown in bold. The system was run on the corpus containing the additional recipes and a comparison based on the overall and class accuracies.

Results

The obtained accuracies by applying the framework before and after the addition of the two recipes are provided in Figure 4.4.

It can be observed from the results that by adding recipes, not only were the additional recipes correctly classified but also the misclassified recipes. Coincidently, another misclassified recipe in the *fry* class was also correctly classified.

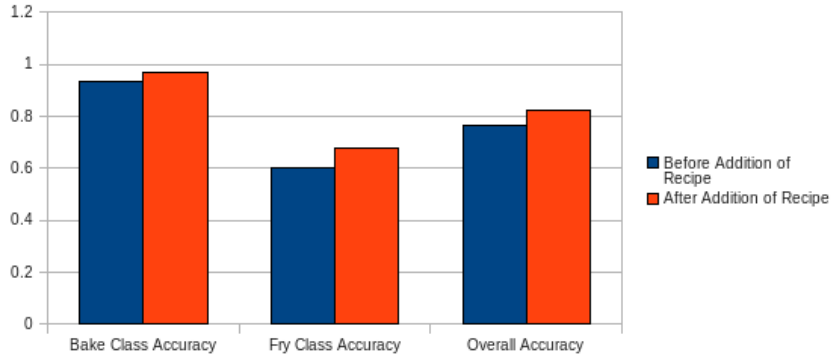


Figure 4.4: Accuracies obtained before and after adding recipes to the corpus

Conclusions

The experiment provided in this section provided results that by increasing the corpus size, misclassified items may be correctly classified. This experiment proves that by increasing the corpus size greatly, all results in this project may be increased. As creating the proposed corpus requires a large amount of time to be made, only the certain sized corpora outline in section 4.3 were made.

The mentioned experiments provided in section 4.4 described how the preprocessing stage of the framework may be improved in order to overcome challenges. Other challenges that were found in the preprocessing step were verbs with multiple POS tags such as *cream* and cases where terms such as *stir in* indicate adding an item to a mixture followed by stirring it where in the proposed system on the verb *stir* was extracted. Many more challenges in converting natural recipe texts into a suitable representation for the extraction of patterns format exist. However, due to the time limit of the project, the focus was given to pattern representation and learning aspects of the system.

4.5 Knowledge Representation Experiments

After preprocessing, the next stage is to apply the knowledge representation aspect of the system. This stage consists of dealing with the extraction and representation of patterns. In this section, experiments are provided that deal with the improvement of this stage.

4.5.1 Experiment 5: Finding Optimal Parameters & Distinct Patterns

The framework is applied using four main parameters. Two parameters are used in level wise mining where the first indicates the number of levels to be used, in this experiment represented as L, and the second represents the minimum frequency each pattern must obtain so that it can be stored as a feature. As patterns in the third level do not create any distinctive feature since they are very unlikely to occur, only the values one and two will be assigned the parameter L. The parameter for minimum

frequency enables the framework to only obtain suitable features for the feature set. However, as feature selection is applied, described in section 3.6, this parameter can be set to the value one for all levels used, which leads to creating all patterns in the corpus, where by applying feature selection the suitable features are found. Therefore in this experiment, the evaluation of the parameter for minimum frequency is replaced by the third parameter which is used to limit how many of the most suitable features are to be chosen in the feature set. The third parameter is represented as F. The fourth parameter is the value chosen for K in the K-nearest-neighbour algorithm described in section 3.7. Using initial exploratory experiments, the value $K = 1$ provided the highest accuracies. Therefore in this experiment the value one will be used in KNN.

To evaluate on the optimal values of the two parameters L and F, the accuracies of using the different combinations of the parameters is obtained using the large corpus. The system used for this experiment will be using the representation of patterns without considering nouns which is evaluated in section 4.5.4. By finding the most optimal pair and using them on the unrestricted corpus, the most distinct patterns that represent a class can be extracted. For each class, two patterns will be provided. The first is the pattern with the highest number of occurrences for a class. The second is the pattern which is the most discriminative pattern for a class which will be found by using the MRMR feature selection tool. Each pattern is represented as a sequence of verbs.

Results

In Figure 4.5, the parameter F is plotted against the accuracies for each value of L. In Table 4.3 the most distinct patterns using the previously described methods are provided by using the optimal values in Figure 4.5. This table provides the two mentioned types of patterns for each class where a pattern is represented in one column.

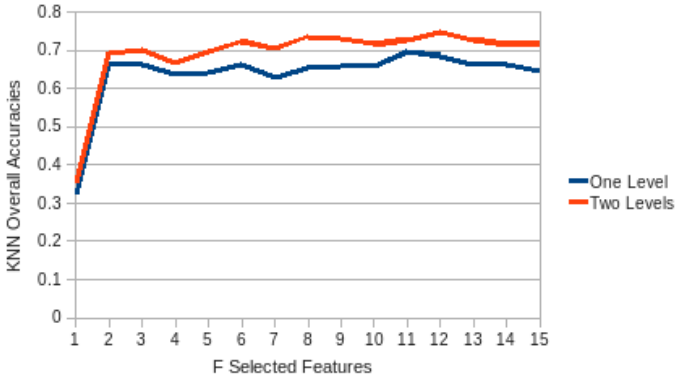


Figure 4.5: Comparison of results using different values for the number of levels L and the F best features given by feature selection from a feature set.

	bake	fry	pastry	roast
Most Occurrence	mix place	heat add	fry cut	heat remove
Most Distinctive	fold mix bake	add fry add	chop drain	prepare place

Table 4.3: The most distinct patterns and the highest occured patterns in the

Conclusions

From the results illustrated in Figure 4.5 it can be concluded that level-wise mining with more than 1 level can produce more beneficial patterns. As a result all experiments will be using patterns obtained by level-wise mining in two levels. This experiment also provided patterns that distinctly represent their classes. It can be observed that certain verbs such as *fry* occur in other classes such as *pastry* which in later experiments, it is concluded as one of the reasons that the bag of words performs poorly in caparison to the framework.

4.5.2 Experiment 6: Using the *before* $\vee$ *meet* Relation

This experiment was conducted to compare against n-grams. By only using the *meet* relation a method similar to n-gram is used. However this only produced patterns of instruction that are in a sequenced. Therefore a new relation was implemented referred to as the *before* $\vee$ *meet* relation. This new relation enables finding patterns where instructions have been separated by other instructions. In order to evaluate on the benefits of this new relation, the accuracies obtained by including and not including it in the framework will be compared.

Results

In Figure 4.6, the results of performing the experiment of using the relation *before* $\vee$ *meet* is compared to not using it.

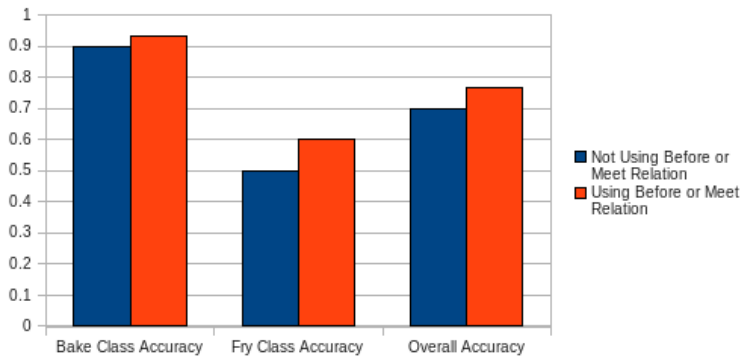


Figure 4.6: Comparison of using the relation *before* $\vee$ *meet*

From the results provided it can be observed that both classes have an increase in accuracy by using the relation *before* $\vee$ *meet*.

Conclusions
From the results obtained, it can be concluded that the *before* $\vee$ *meet* relation improves the system by creating more pattern. The additional patterns created represent patterns obtained from separated instructions during a recipe.

4.5.3 Experiment 7: Pattern Creation by Considering Common Entity

As describe in section 3.5, a pattern represents a temporally sequenced set of verbs which share common nouns. The initial implementations of the system only considered sequences of verbs as patterns where the verbs had exactly two nouns in common. One of the early challenges was that the representation for a pattern was not suitable to use for instructions in the raw recipes. This was due to the fact that in an instruction in a real recipe, a verb may have any number of nouns e.g. *Combine flour, oats, cinnamon and sugar*. Therefore the representation in the preprocessed recipes was changed to consider verbs with various number of nouns. However, it was observed that in recipes, finding patterns which consist of two verbs having the same number and value of nouns, is very unlikely to occur. This method created very few patterns which made the feature representation not suitable for representing some recipes, leading to their misclassification. To overcome this problem, patterns were also created from a sequence of two verbs where they at least shared one common noun. An example of this representation is provided in Table 4.4 where the noun *flour* is common for the verbs.

Recipe pattern
add oil flour sugar
stir egg flour

Table 4.4: A mined pattern with the new representation where two verbs only require one noun in common

The new method of representing patterns increases the computation time of the system to several minutes compared to before which was instantaneous. However, by using this representation, the number of patterns (features) in the feature set was significantly increased. The experiment set to find the most beneficial method of representing pattern is to apply the framework using the two methods described and compare the overall and class accuracies.

Results
The accuracies obtained by applying the framework using the two methods of creating patterns are provided in Figure 4.7.

From the results obtained, it can be observed that by using the representation of patterns based on one common noun the accuracy of the *fry* class significantly improves by 30% at the cost of

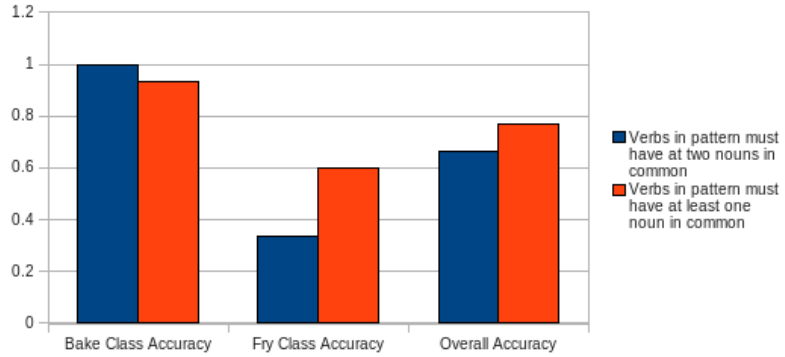


Figure 4.7: Comparison of two pattern creating methods where one requires two common nouns and the second requires only one common noun for a pattern to be created.

misclassifying 2 items in the *bake* class.

Conclusions
Due to the significant increase of the fry class, it can be concluded that the pattern representation of using one common noun is more suitable than using two common nouns as it provides more patterns.

4.5.4 Experiment 8: Learning Patterns Without Considering Nouns

Another challenge in the project were cases where recipes did not provide distinct patterns that would represent their class. This occurred for two reasons. Firstly, it was not common for verbs in recipes to have the same nouns which is required for a pattern. Secondly, due to the nature of written language, nouns mentioned in one instruction are rarely mentioned in instructions that come after. Therefore the representation of patterns where verbs require to have the same nouns may not be a suitable one. A new approach to creating patterns is to only consider verbs for pattern creation.

This approach differs from the previous pattern extraction method where it obtains all paired combinations of verbs, regardless of their nouns, rather than a paired combination of verbs which have common nouns. The experiment set is to apply the system using the new pattern representation on the large corpus and compare it with the results illustrated in Figure 4.12 which were produced by using the pattern representation using common nouns and the bag of words method on the large corpus.

Results
The overall and class accuracies obtained by applying the bag of words method and the system using the pattern representation of verbs with and without common nouns is illustrated in Figure 4.8. In Figure 4.8, the new representation of patterns created by verbs without considering nouns obtains

higher accuracies compared the when considering nouns. Its results are very similar to the bag of words method where it only misclassifies one more item in the fry class.

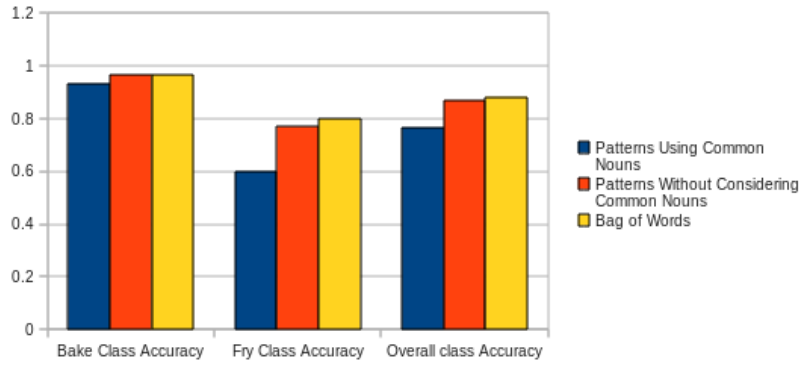


Figure 4.8: Comparison of using the pattern representation of verbs without considering their nouns and verbs with common nouns

Conclusions

From the results in Figure 4.8 It can be concluded that by creating patterns based on temporally paired verbs without considering their nouns is more suitable to use as a representation than considering nouns. To further investigate the reason that this method misclassified seven recipes in the fry class, the experiment in section 4.5.5 is set.

4.5.5 Experiment 9: Investigating Misclassifications

In order to investigate the reason for the prototype's seven misclassified items in the fry class, a histogram was created based on the score in which each recipe in the fry class has obtained in the feature set. This score is represented as the total number of occurrences of features in each recipe's feature vector.

Results

The obtained histogram is illustrated in Figure 4.9 where it can be observed that the first three bins in the histogram consist of seven recipes that obtained the lowest score which represents them not having many patterns. By comparing each of these recipes and its prediction in KNN, out of the seven lowest scoring recipes, six of them were misclassified. This strongly suggests that the reason the recipes were misclassified is not due to the functionality of the system but the lack of patterns in the recipes. An example of one of these recipes in raw and preprocessed form is provided in Appendix E in section E.2.

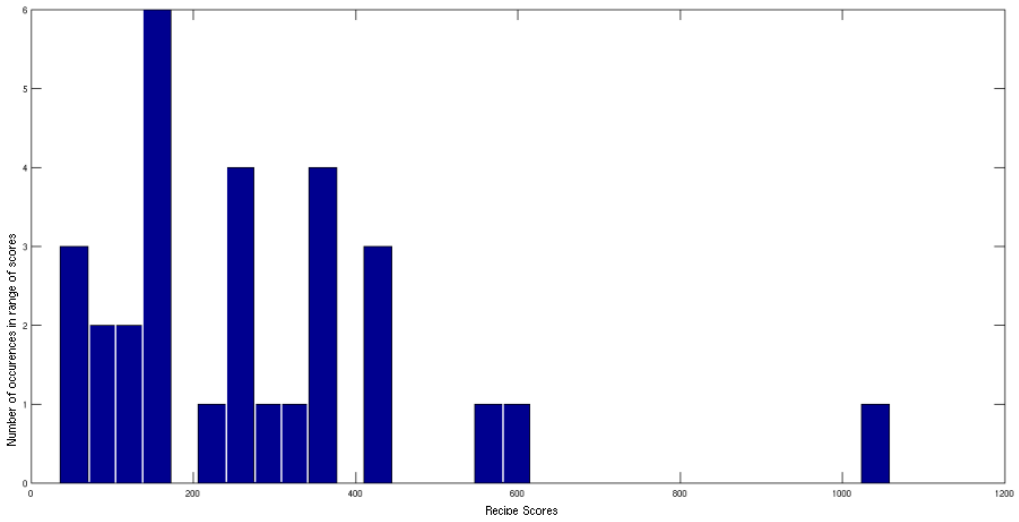


Figure 4.9: A histogram based on the total number of occurrences of features for each recipe in the feature set.

Conclusions

With this experiment is was found out that the main reason for misclassified items is due to them not containing many patterns. As this method does not use patterns based on nouns, it proves that their may be recipes which naturally do not contain many patterns. A modification that can be applied on the prototype to allow it to correctly classify recipes which only consist of a small number of patterns is to store nouns in the patterns. This means that in addition to using the name of verbs in a pattern, the nouns they have in common are also stored. In the current system, a pattern is known as the structure and the verb names. However, by also using nouns, in addition to it containing the structure and the same verb names, it must also contain the same noun names. Due to this method being outside of the scope of the project, this approach was set for future work. For comparison purposes only, a bag of words method using the occurrence of nouns and verbs is illustrated in Figure D.5 in Appendix D where it is compared to the system and bag of words method used in section 4.7

4.6 Machine Learning Experiments

The final stage of the system is the learning aspects of it. This stage consists of applying supervised and unsupervised methods of learning on the feature set obtained from the graph mining stage. In this section various experiments set to evaluate the performance of methods used to develop the final system is performed. In section 4.6.1 the system is evaluated against the bag of words method using occurrences of verbs as features.

4.6.1 Experiment 10: Graph Mining Vs. Bag of Words

In previous experiments, various methods were compared where by combining the most suitable methods a framework was created. To evaluate the performance of this framework, the bag of words method is chosen as baseline. During the implementation of the framework, various comparisons were made with the baseline using the different corpora described in section 4.3. By using each corpus, the framework was improved by applying the methods described in previous experiments. The results of applying the framework and the bag of words method on each corpus is the given below.

Results

In Figures 4.10, 4.11 and 4.12, the results obtained by applying the framework (referred to as graph mining in figures) using all modifications evaluated as beneficial in the experiments and the baseline method on the small, medium and large corpora are illustrated.

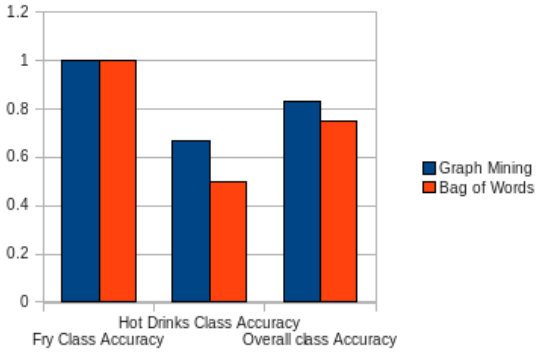


Figure 4.10: Results obtained by applying the system and the bag of words method on the small corpus.

From the results in Figure 4.10 it can be observed that both methods have obtained the same accuracy for the *fry* class, however, the bag of words method has obtained a lower accuracy for the *hot drinks* class. This indicates that the system produces features which are more suitable to use than only verb occurrence features used in the bag of words metho. After more experimentation it was concluded that the *hot drinks* class was not suitable to use. This was due to the fact that the class did not have many distinct patterns (features) representing it, while the patterns it did have were also frequent in the *fry* class. For example, a pattern that occurred in the *hot drinks* class contained the verb *pour* followed by the verb *stir*. Although this pattern frequently occurred in the *hot drinks* class, it also occurred in the *fry* class. Since this pattern was not distinct in representing the class, it led to many *hot drinks* recipes being classified as the *fry* class. As the small corpus was created for the purpose of developing the system a more suitable class with distinct patterns was needed. Therefore, the corpus was changed by replacing the *hot drinks* class with a new *bake* class, while keeping the *fry* class while also increasing the number of recipes in each class. This corpus is the medium corpus. The results of applying the system and the bag of words method on the medium corpus is illustrated in Figure 4.11.

Figure 4.11: Results obtained by applying the system and the bag of words method on the medium corpus.

By comparing the KNN accuracies of the *fry* class in Figure 4.11, it can be observed that the system has obtained a higher accuracy compared to the bag of words method. Although the results were obtained from a corpus of artificially preprocessed recipes real recipes and also the bag of words method obtained a higher accuracy for the *bake* class, the results indicate that there are cases where the methods and representations used in the system have an advantage over the bag of words method. This advantage may be due to using temporal relations in addition to using verbs. To fully evaluate the system and the bag of words method, they are applied on the large corpus which unlike the previous corpora, it contains automatically preprocessed recipes. The results of using the large corpus is illustrated in Figure 4.12

Figure 4.12: Results obtained by applying the system and the bag of words method on the large corpus.

By comparing the results in Figure 4.12, we can conclude that the bag of words method outperforms the framework using patterns of one common noun in classifying the *fry* and the *bake* classes.

Conclusions

The main reason for the system's shortcomings can be concluded to be that nouns are not frequently repeated in sentences in natural language recipes. It is very likely in recipes that when an ingredient is used in an instruction it is never mentioned again in the recipe. This creates a significant problem where many patterns are not created. The patterns that are created are likely to occur in all classes which makes them less distinctive for representing classes. By overcoming this problem using the method outlined in section 4.5.4 and incorporating it in the framework, the final framework is created which is evaluated in section 4.7.

4.6.2 Experiment 11: Applying Feature Selection

One of the methods of improving the learning process is to apply it on a more suitable feature set. By using the MRMR feature selection, described in section 3.6, the features that are not suitable for representing classes are removed. This method outputs the N most suitable features in a feature set. To evaluate the MRMR feature selection method, it will be compared to a baseline using standard deviation. In this baseline, the occurrence of each feature for each class was calculated. This produced a vector for each feature which had the size of the total number of classes used. By applying standard deviation on this vector, the suitability of a feature was measured where a high standard deviation means the feature is very suitable in representing certain classes as there is a large contrast between the values in the vector. The N feature with the highest standard deviation values where given as suitable features to be chosen in the main feature set. The experiment set to evaluate the MRMR and the standard deviation method is to apply the framework on feature sets created by using the N most suitable features provided by each method where N is changed between one and one thousand. This experiment was set using the unrestricted corpus as it contains the most recipes among all corpora which leads to giving the most number of features. Using feature selection on more features will better illustrate the benefits of using this method.

Results

The overall accuracies using the MRMR and standard deviation feature selection methods for various values of N is illustrated in Figure 4.13. Figure 4.13 illustrates that the feature set provided by the MRMR method obtains a higher accuracy compared to the standard deviation method.

Conclusions

By using the MRMR feature selection method the framework can obtain higher accuracies since more suitable features in representing the classes are used. The benefit of using the MRMR feature selection method in the final framework is described in section 4.7.

Although the bag of verbs method obtained a slightly higher overall accuracy compared to the framework using only verbs as patterns, this framework can be considered as a final framework. To

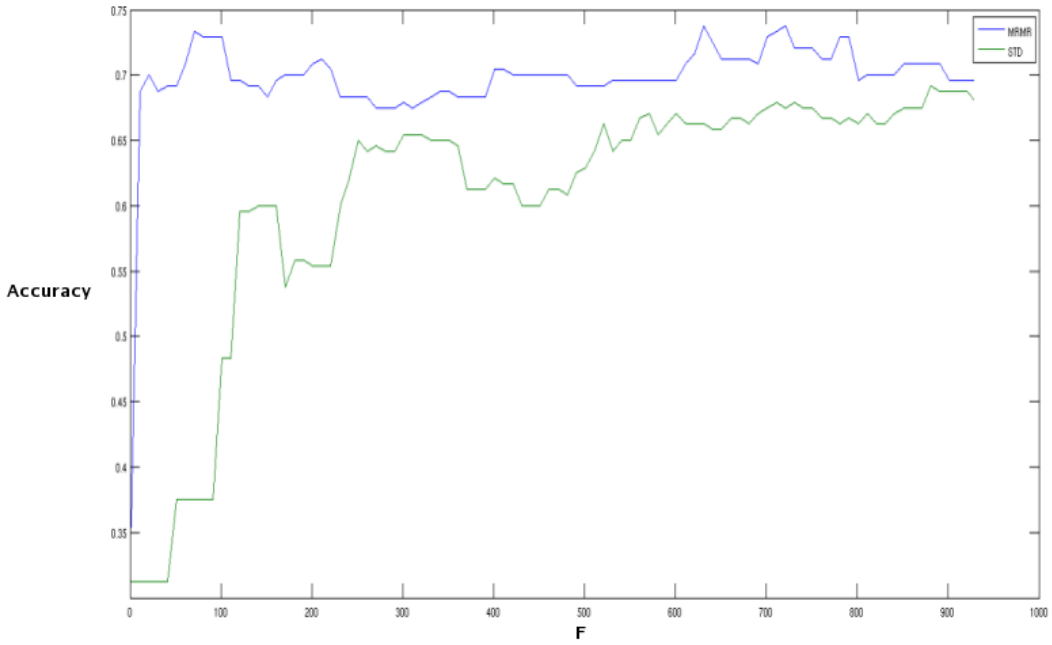


Figure 4.13: Overall accuracies obtained by applying the framework on feature sets using two feature selection methods, MRMR and standard deviation where F represents the number of extracted features from the feature set.

make an evaluation of this framework, it must be applied on a corpus which contains multiple classes and does not have the restrictions of extracting recipes, described in section 4.3, where the previous recipes experimented on had. The evaluation of the framework applied on the unrestricted corpus is given in section 4.7.

4.7 Experiment 12: Applying Final framework on the Unrestricted Corpus

The final framework consists of all the methods that were evaluated in the previous experiments and where concluded as improving the framework. To evaluate the final framework, it is applied on the unrestricted corpus. This corpus contains the four classes of *bake*, *fry*, *pastry* and *roast* where each class contains sixty recipes. As the number of patterns are very large, MRMR feature selection, evaluated in feature section 4.6.2, is applied on the feature set obtained by graph mining before the learning stage of the framework. The experiment set for the evaluation of the final framework is to apply it on the unrestricted corpus by using the KNN ($K = 1$) and SVM (using default values for parameters) methods and compare it to the results obtained by the bag of words method with the same learning methods. To also evaluate on the benefit of using feature selection, the framework will be applied with and without using feature selection. An unsupervised evaluation is provided in section 4.7.3 using the X-means algorithm.

Results

Figure 4.14 illustrates the overall and class accuracies of the final framework, with and without using feature selection, and the bag of words method using the unrestricted corpus.

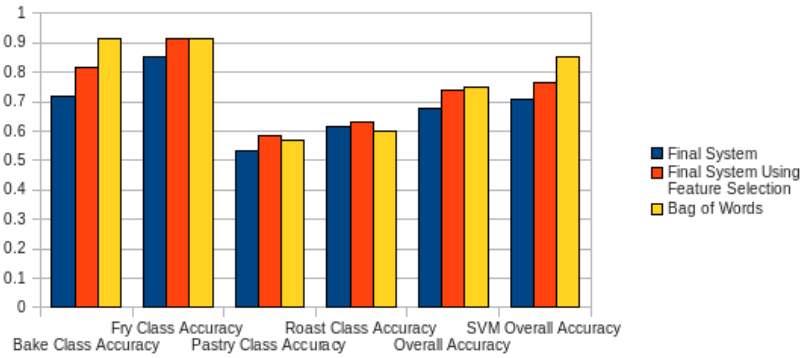


Figure 4.14: Results of the final framework, with and without using feature selection, and the bag of words method.

From the results illustrated in Figure it can be observed that the final framework using feature selection has obtained a higher accuracy in all classes compared to the final framework which is not using feature selection. Compared to the bag of words method, it can also be observed that the final framework using feature selection has obtained a lower accuracy for the *bake* class, the same accuracy for the *fry* class, and a higher accuracy for the *pastry* and *roast* class. It has also obtained an overall accuracy very similar to the bag of words method.

Conclusions

From the results obtained it can be concluded that by using a KNN method the Final framework is very performs very similarly to the bag of words method, however by using an SVM the bag of words obtains a higher overall accuracy. As each method obtained higher accuracies in certain classes, experiment in section 4.7.1 combines feature sets from both methods.

4.7.1 Experiment 13: Combining Feature Sets

This experiment was set to combine the final framework with the bag of words method. This was done to use the most suitable features from both methods. Four various ways to combine the feature sets are experimented on. The first two methods are to combine both features sets and used with and without feature selection being applied on it. As the bag of words method obtained higher accuracies for the *bake* and *fry* classes and the framework obtained higher accuracies for the *pastry* and *roast* classes, the third combined feature set consisted of the features of both methods, however, unlike before in the bag of words feature set all feature vectors for the *pastry* and *roast* classes were set to zero while in the feature set for the framework, all vectors for the *bake* and *fry* classes were set to zero. In other words, only the feature vectors of the classes that each method was most suitable for representing were used

in the final combined feature set. The fourth method is to apply feature selection on the third method. In the results, the first method is referred to as *SYS+BoW*, the second as *SYS+BoW+FS*, the third as *SYS+BoW+CLASS* and the fourth as as *SYS+BoW+CLASS+FS*.

Results

The results of applying the framework on the described four feature sets are illustrated in Figure 4.15. These results are compared against the original feature set obtained by the bag of words method as it obtained the highest accuracy compared the the feature set obtained by the framework.

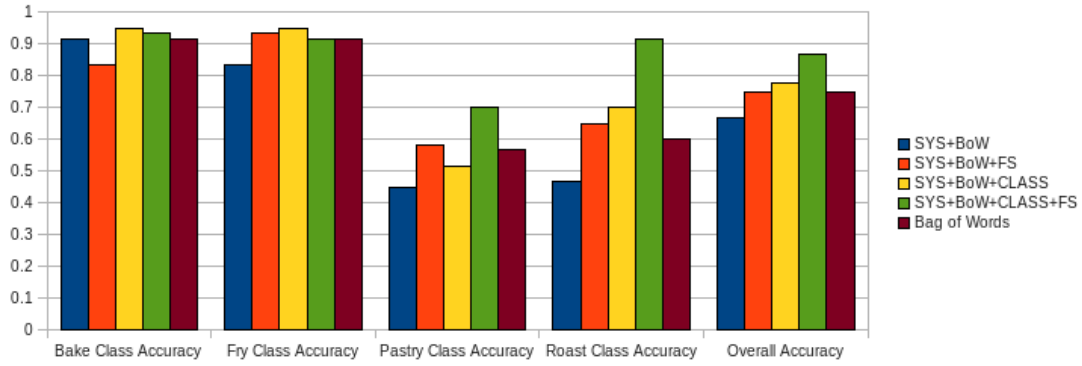


Figure 4.15: Results of combining the feature set using four different methods

From the results it can be observed that the fourth method of combining the feature sets has obtained the highest accuracy where compared to the original bag of words method it has obtained a significantly higher overall accuracy.

Conclusions

The results in Figure 4.15 illustrates that by combining the feature set and applying feature on it, higher accuracies are obtained. It can also be concluded that the *pastry* and the *roast* classes are better represented as temporal patterns by the framework while the *bake* and the *fry* classes are better represented as the occurence of verbs using the bag of verbs method. One of the reasons that the bag of words method has difficulty classifying the *pastry* class compared to the framework is that in the experiment in section 4.5.1 it was found that the most occurred pattern in the *pastry* class is the sequence of the verbs *fry, cut*. As the verb *fry* frequently occurs in the *pastry* class, recipes in having this class are very likely to be misclassified. This illustrates one of the strengths of the developed framework over the bag of words method.

4.7.2 Experiment 14: Application of the Framework

This experiment was set to evaluate the performance of the framework as an information retrieval tool used by users. The class of the accuracy may be found by using the model built by the SVM and by

finding the closest neighbour of this recipe, the most similar recipe to it is found. In this experiment a participant was asked to write a recipe in textual form. This was a *chocolate cake* recipe which belongs to the *bake* class. This recipe is provided in Appendix E in section E.2. The recipe was classified by using a SVM and the most similar recipe to it was found by using a KNN method.

Results
By applying the SVM the class of the recipe was predicted as the *bake* class and the most similar recipe to it was the *Chocolate Fairy Cakes* recipe. This recipe is provided in Appendix E in section E.2.

Conclusions
From the results obtained it can be concluded that the framework may be used to classify the class of the recipe provided by the user. The framework can also be used to search for similar recipes to the recipe the user requested.

4.7.3 Experiment 15: Clustering Evaluation Using Cognitive Clustering

To evaluate the clustering of the framework, it is compared to the bag of words method used, where in both X-means clustering is applied on their feature sets. Two feature sets are chosen for the framework where the first is the original feature set obtained by the framework where feature selection is applied on and the other is the feature set which obtained the highest accuracy in section 4.7.1. This feature set was created by combining the feature sets of the framework and bag of words method and modifying it based on the strengths of each. To evaluate on the outputted clusters against the ground truth, the rand index [51] is used which defines the similarity measure between two different clusters. The overall accuracy obtained by using the rand measure will be used to compare the different feature sets.

To further investigate the clustering process that was applied, histogram of the frequency of items in each cluster is provided for the original feature set where feature selection is applied on it. To evaluate the clustering for this feature set, it is evaluated against a cognitive clustering where participants cluster recipes. This method of evaluation was thought of based on the works of Ul-Qayyum et al. [59]. If the number of clusters is close to human clustering, the framework will be considered as suitable for unsupervised learning.

Results
The obtained accuracies by using the mentioned three methods are illustrated in Figure 4.16. The results illustrate in Figure 4.16 show that the bag of words method obtains the highest accuracy. The histogram of the original feature set where feature selection is applied on is illustrated in Figure 4.17. It can be observed from the histogram that the first cluster contains a large number of recipes and in total there are 11 clusters.

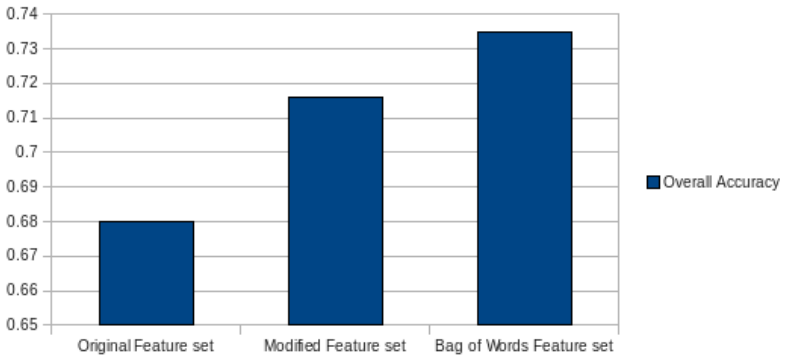


Figure 4.16: Results clustering for the evaluation of the most suitable feature set.

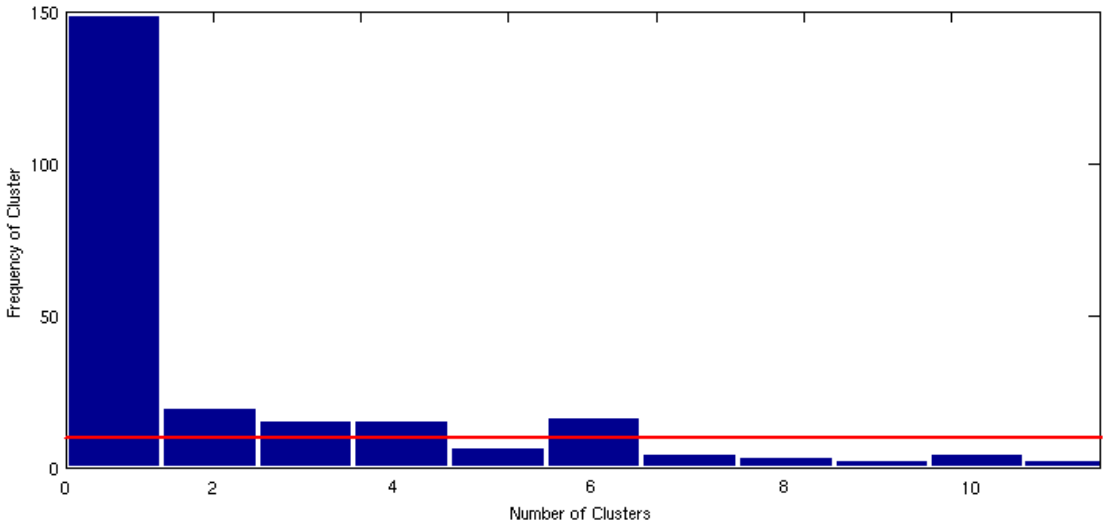


Figure 4.17: Results of obtaining a histogram based on the frequency of item belonging to a cluster using the original feature set obtained by the framework where feature selection is applied on.

Conclusions

The clustering accuracy obtained by using the bag or words approach is higher than the other two methods. The low accuracy obtained by clustering using the original feature set obtained by the framework is due to finding more than four clusters (where each would represent a class). This may be as a result of multiple classes existing in either the *fry* or the *bake* classes. The classes that exist in other classes will be referred to as inter-classes. For example, the recipes in the *bake* class consist of inter-classes such as baking cakes, tarts and breads where each of the inter-classes may have a unique structure. As the prototype is implemented to find temporal structures, each of the inter-classes in the *bake* class may also have a unique structure which will result in inter-classes being separated in the feature space where a cluster may be found for each inter-class. As the ground truth values for clustering were set for only 2 classes and the X-means clustering algorithm finds more than 2 clusters due to the inter-classes, the accuracy is significantly reduced. However, the bag of words method does not use structure which results in ignoring the unique structures of the inter-classes. The high clustering accuracy obtained by using the bag of words method suggests that the recipes in each class use common verbs.

As previously described, certain recipes do not contain many patterns and are therefore not represented suitably for clustering. A threshold was applied which is illustrated as a red line in Figure 4.17. This threshold of a value of ten is given so that if a cluster has less frequency of items than this threshold, the items are not considered to be representative for clustering due to their lack of patterns. Therefore their cluster is not considered. The value 10 was chosen as it is one sixth of the number of items in each class which was considered as a suitable threshold for items that do not have a suitable pattern representation. By applying the threshold we can conclude that the framework clusters the feature set into 5 clusters which is not very different from the ground truth value of 4. To evaluate this further Cognitive clustering was performed.

As it requires a long time to manually cluster preprocessed recipes, only 5 items of recipes for four classes where given to participants, giving a total of 20 recipes. Four participants clustered the recipes where 2 clustered the recipes into 4 clusters, another participant into 5 and the last participant clustered the recipes into 6 clusters. As the framework very closely obtained the average of the participant clusterings, it can be concluded that the unsupervised clustering in the framework using the threshold may provide a reasonable number of clusters to represent the number of classes of recipes used.

4.8 Disscussion

In this chapter the framework described in Chapter 3 was evaluated. The evaluation of the framework was applied by using a supervised method against a baseline bag of words method and an unsupervised cognitive clustering method. In both cases the framework proved to get very similar results it

was being evaluated against. It was concluded that the major shortcoming of the framework is the lack of patterns in the recipes. Various methods in this chapter were evaluated and concluded to improve the framework. However, the most accurate is the method which combines the feature sets of the bag of words and the framework and applies learning on only parts of the feature set.

Many more methods to be implemented and challenges to be overcome are present for the case of the framework which will be described in the next section.

Chapter 5

Conclusion

In this chapter the overall project is concluded by evaluating whether the minimum requirements have been met and the research questions initially set have been answered. Moreover, this chapter describes the exceeding requirements that were achieved, the limitations of the system and any future work that may be set to continue this project.

5.1 Project Evaluation

In the following sections, the achievements and research questions of the project are described followed by an overall conclusion of the project.

5.1.1 Aim & Minimum Requirements

In this project the following aims and minimum requirements were set. In each case a description of the way that it was achieved follows:

1. **Designing a representation for the temporal relations.** In section 3.4 the graph structure used to represent recipes is provided and In section 3.5 the definition of patterns which are extracted from the mentioned graph structure are described. The representations are then evaluated in section 4.5.3 and 4.5.4.
2. **Producing an event learning system.** The learning system, described in section 3.7, was developed using a supervised and an unsupervised approach. In order to improve the performance of the learning system various methods were used and evaluated. These methods are described in experiments in chapter 4.

3. **Developing a prototype system to classify corpora recipes based on their graph structures.** In section 4.7, the final prototype was applied on an unrestricted recipe and evaluated based on the unsupervised and supervised aspects of the prototype in the same section.

5.1.2 Research Questions

The following research question were set to be answered by completing the project.

1. *Can nouns and verbs be accurately mined from a recipe using Natural Language Processing techniques?* Although the nouns and verbs can be extracted with a certain precision, for this project a dictionary lookup method was used which provided very accurate results for the porpose of this project. The drawback of this method however was the problem of missing nouns which were represented as prepsitions. By using the proposed *resolving missing nouns* method this problem was overcome to a level which was more beneficial than ignoring the case of missing nouns. To further extend the project and to avoid the need of an accurate noun and verb extraction method various pattern representations were introduced such as patterns using only one common noun or patterns not considering nouns at all. These methods are further discussed in sections 4.5.3 and 4.5.4.

2. *Is it possible to find the most representative pattern in a class?* In section 4.5.1 for each class of recipes, two of the most distinct patterns in representing it were provided. The patterns provided by using the feature selection method evaluated that the patterns distinctively represent a certain class. Therefore, it is possible to find the most representative pattern for a class.

3. *What is the most suitable structure for a pattern representation?* The most suitable representation for a pattern was concluded to be a pattern which is created from paired combinations of verbs. This was evaluated in section 4.5.4. However, it was proposed that by using nouns as types, the structure of the pattern will be more unique among classes as the noun name will be stored in the representation. However, due to it being outside of the scope of the project, this method was not implemented.

4. *Can the class of a recipe be predicted from its instructions?* In section 4.7.1 by combining the feature sets obtained from the system and the bag of words method, recipes belonging to four classes where classified with 87% accuracy. By further improving the framework this accuracy is very likely to increase.

5. *Will an unsupervised approach be more suitable than a supervised approach to be applied by the framework?* Although no evidence was found that suggests an approach is more suitable than another, However, each method has its strengths which can benefit the framework. For example, evaluated in section 4.7.2, a SVM is able to predict the class of an item while the KNN

method finds the most similar recipes to the item. The benefit of using clustering, described in section 4.7.3, is to group all classes together and if one items belongs to only one class, it may be concluded that either that recipe does not belong to the class it was given to or it contains noise.

6. *Will the framework be able to automatically find the number of clusters which represent the classes of recipes accurately.* In section 4.7.3, it was concluded that by applying a certain threshold to remove clusters that contain recipes without patterns, the framework finds the number of classes as clusters very simalar to the methods it was evaluated against.

7. *Is it possible to classify cooking recipes with graph representations using unsupervised methods?* In section 4.7.3, the results provided indicated that although the framework can provide the number of clusters, it lacks the ability to be able to accurately group all items of the classes together. This is due to the feature representation being unsuitable for a clustering process.

5.1.3 Exceeding Requirements

1. Although, it was not the the aim of the project to compete with the bag of words method, various new methods were found that improved the accuracy of the system. This led to the system obtaining higher accuracies in certain classes.
2. Originally the project was set to only consider an unsupervised system, however for the purpose of further investigation of the system a supervised approach was also used.
3. Two method where proposed and applied to further increase the performance of the system. The two methods are using domain knowledge and increasing the corpus size. These methods are evaluated in sections 4.4.2 and 4.4.3.

5.1.4 Conclusion

As an overall conclusion, the project was successful in modifying a system originally created for an airport domain and applying it on a cooking domain. Various methods were proposed and implemented which increased the performance of the system and compared to its baseline method it obtained higher accuracies for certain classes.

5.2 Challenges

1. The main challenge of the project was that it was very research based and due the time limit, many approaches where not implemented with their current state of the art method. Therefore, basic implementations were used to substitute approaches.
2. Another challenge in the project was that it applied three different areas of artificial intelligence, namely natural language processing, knowledge representation and machine learning. Various methods were required from each of these areas where using the state of the of art way of using

these methods was not possible. This led to using simple methods such as the dictionary lookup method, described in section 3.3, which was a simple alternative to using POS taggers.
3. One of the early challenges in the project was that no recipe corpus was available and the required corpora had to be created.
4. Another challenge in the project was the lack of repetition of nouns in recipes. This extended the implementation and experimentation period as a suitable method had to be found which did not require accurate methods of extracting nouns.
5. As the system was created by modifying existing code, a large amount of time was spent on debugging and finding bugs.

5.3 Future Work

Enhancements

The following are future enhancements that may be applied on the proposed framework to further increase its classifying performance:

1. By using recipes where their instructions may partially overlap, more Allen relations may be used. This may also include spatial relations where for example chicken is being fried in a pan, while simultaneously noodles are being boiled in a pot.
2. Investigate which verbs are beneficial for substitution.
3. Use parsers to find the correct objects for verbs and give any missing nouns nouns in the sentence to a verb.
4. Find a suitable approach in dealing with words such as *cream* that have multiple word senses and word such as *stir in* that represent two actions
5. Modify the system to run without the need of dictionaries where pre-processing will be applied automatically.

Future Projects

The following are proposed project which may be based on this project:

1. Apply the system to a new domain such as Manuals on building objects, daily schedules and techniques in martial arts to that represent actions.
2. Learn human action such as running, kicking, waving, since the sequence of movements may be stored and learned.
3. Use recipes along with video to find the missed detections in parts of the video.
4. To compete with the bag of words method using noun and verbs investigate using nouns as types to represent a more structured patter.

Bibliography

[1] ABULAISH, M., AND DEY, L. Biological relation extraction and query answering from medline abstracts using ontology-based text mining. *Data and Knowledge Engineering 61*, 2 (2007), 228–262.

[2] ALLEN, J. F. Towards a general theory of action and time. *Artif. Intell. 23* (July 1984), 123–154.

[3] ALLRECIPES. Recipes. `http://allrecipes.com/Recipes/main.aspx`. Last Accessed: 22/08/2011.

[4] ALPAYDIN, E. *Introduction to machine learning*, 2 ed. MIT Press, 2010.

[5] ANAEKWE, B. N. Histogram visualization for file-system exploration. Msc project, University of Leeds, 2010.

[6] BALABAN, A. T., LIU, X., KLEIN, D. J., BABIC, D., SCHMALZ, T. G., SEITZ, W. A., AND RANDIC, M. Graph invariants for fullerenes. *Journal of Chemical Information and Computer Sciences 35*, 3 (1995), 396–404.

[7] BELLAZZI, R., AND ZUPAN, B. Predictive data mining in clinical medicine: Current issues and guidelines. *International Journal of Medical Informatics 77*, 2 (2008), 81–97.

[8] BHATIA, K. An unsupervised learning algorithm for inductive logic programming using expectation maximization. Master's thesis, University of California, San Diego, 1995.

[9] BIRD, S. NLTK: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics* (2002).

[10] BIRD, S., KLEIN, E., LOPER, E., AND BALDRIDGE, J. Multidisciplinary instruction with the natural language toolkit. In *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics* (Stroudsburg, PA, USA, 2008), TeachCL '08, Association for Computational Linguistics, pp. 62–70.

[11] CHANG, C., AND LIN, C. LibSVM - a library for support vector machines. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

[12] CHEN, Y., WANG, G., AND DONG, S. Learning with progressive transductive support vector machine. *Pattern Recognition Letters 24*, 12 (2003), 1845 – 1855.

[13] CLARKE, B., FOKOUE, E., AND ZHANG, H. H. *Principles and Theory for Data Mining and Machine Learning*, 1 ed. Springer Publishing Company, 2009.

[14] COOK, D. J., AND HOLDER, L. B. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research 1* (1994), 231–255.

[15] CUNNINGHAM, H. GATE, a General Architecture for Text Engineering. *Computers and the Humanities 36*, 2 (2002), 223–254.

[16] DEMPSTER, A. P., LAIRD, N. M., AND RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological) 39*, 1 (1977), 1–38.

[17] DENIS, P., AND MULLER, P. Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)* (July 2011), pp. 1788–1793.

[18] DESHPANDE, M., KURAMOCHI, M., WALE, N., AND KARYPIS, G. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. on Knowl. and Data Eng. 17* (August 2005), 1036–1050.

[19] DYBKJAER, L., HEMSEN, H., AND MINKER, W. *Evaluation of Text and Speech Systems*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[20] ET AL., J. A.-R. Computational load reduction in decision functions using support vector machines. *Signal Processing 89*, 10 (2009), 2066 – 2071.

[21] GALAR, M., FERNNDEZ, A., BARRENECHEA, E., BUSTINCE, H., AND HERRERA, F. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition 44*, 8 (2011), 1761 – 1776.

[22] GALTON, A. Spatial and temporal knowledge representation. *Earth Science Informatics 2* (2009), 169–187.

[23] GASTEIGER, J. *Chemoinformatics: a textbook*. Wiley VCH, 2003.

[24] GIUDICI, P. *Applied Data Mining Statistical Methods for Business and Industry*, 1 ed. Wiley & Sons, 2003.

[25] GRUNE, D., AND JACOBS., C. J. H. *Parsing Techniques: A Practical Guide*, second ed. Springer, 2008.

[26] GUO, G., LI, S. Z., AND CHAN, K. L. Support vector machines for face recognition. *Image and Vision Computing 19*, 9-10 (2001), 631 – 638.

[27] HAN, J., AND KAMBER, M. *Data Mining: Concepts and Techniques*, 2 ed. Morgan Kaufmann Publishers, 2006.

[28] HEARST, M. A. Untangling text data mining. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics* (1999), ACL '99, Association for Computational Linguistics, pp. 3–10.

[29] HSU, C.-W., AND LIN, C.-J. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on 13*, 2 (mar 2002), 415 –425.

[30] HUGHES, B., AND COTTERELL, M. *Software Project Management*, fourth ed. McGraw-Hill Companies, Berkshire, 2006.

[31] INDURKHYA, B. Emergent representations, interaction theory and the cognitive force of metaphor. *New Ideas in Psychology 24*, 2 (2006), 133–162.

[32] KASS, R. E., AND WASSERMAN, L. A Reference Bayesian Test for Nested Hypotheses and its Relationship to the Schwarz Criterion. *Journal of the American Statistical Association 90*, 431 (1995), 928–934.

[33] KATAGIRI, S., AND ABE, S. Incremental training of support vector machines using hyperspheres. *Pattern Recognition Letters 27*, 13 (2006), 1495 – 1507.

[34] KAZAKOV, D., AND KUDENKO, D. Machine learning and inductive logic programming for multi-agent systems. In *Multi-Agent Systems and Applications*, vol. 2086 of *Lecture Notes in Computer Science*. Springer, 2006, pp. 246–270.

[35] KHREISAT, L. A machine learning approach for arabic text classification using n-gram frequency statistics. *Journal of Informetrics 3*, 1 (2009), 72 – 77.

[36] KOTSIANTIS, S. B. Supervised machine learning: A review of classification techniques. *Informatica 31* (2007), 249–268.

[37] KUMAR, E. *Natural Language Processing*. I.K. International, 2011.

[38] LANGLEY, P. *Elements of machine learning*. Morgan Kaufmann Publishers, 1996.

[39] LEROY, G., AND RINDFLESCH, T. C. Effects of information and machine learning algorithms on word sense disambiguation with small datasets. *International Journal of Medical Informatics 74*, 7-8 (2005), 573–585.

[40] LI, D.-C., FANG, Y.-H., AND FANG, Y. F. The data complexity index to construct an efficient cross-validation method. *Decision Support Systems 50*, 1 (2010), 93 – 102.

[41] LIAW, Y.-C., LEOU, M.-L., AND WU, C.-M. Fast exact k nearest neighbors search using an orthogonal search tree. *Pattern Recognition 43*, 6 (2010), 2351 – 2358.

[42] MACQUEEN, J. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), vol. 1, University of California Press, pp. 281–297.

[43] MANNING, C. D., RAGHAVAN, P., AND SCHÜTZE, H. *Introduction to Information Retrieval*, 1 ed. Cambridge University Press, 2008.

[44] MCKAY, B. D. *nauty User's Guide (Version 2.4)*. Department of Computer Science, Australian National University, 2009.

[45] MILLER, G. A., BECKWITH, R., FELLBAUM, C., GROSS, D., AND MILLER, K. Wordnet: An on-line lexical database. *International Journal of Lexicography 3* (1990), 235–244.

[46] MOHRI, M., AND ROARK, B. Probabilistic context-free grammar induction based on structural zeros. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter* (2006), HLT-NAACL '06, Association for Computational Linguistics, pp. 312–319.

[47] MUGGLETON, S., AND RAEDT, L. D. Inductive logic programming: Theory and methods. *Journal of Logic Programming 19*, 20 (1994), 629–679.

[48] PELLEG, D., AND MOORE, A. W. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning* (San Francisco, CA, USA, 2000), ICML '00, Morgan Kaufmann Publishers Inc., pp. 727–734.

[49] PENG, H., LONG, F., AND DING, C. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27* (2005), 1226–1238.

[50] PRESA, J. L. L. *Efficient Algorithms for Graph Isomorphism Testing*. PhD thesis, King Juan Carlos University, 2009.

[51] RAND, W. M. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association 66*, 336 (1971), 846–850.

[52] SMITH, D. Deliaonline. `http://www.deliaonline.com/`. Last Accessed: 22/08/2011.

[53] SRIDHAR, M. *Unsupervised Learning of Event and Object Classes from Video*. PhD thesis, University of Leeds, UK, 2010.

[54] SRIDHAR, M., COHN, A. G., AND HOGG, D. C. Unsupervised learning of event classes from video. In *In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)* (2010).

[55] SUZUKI, M., YAMAGISHI, N., TSAI, Y.-C., ISHIDA, T., AND GOTO, M. English and taiwanese text categorization using n-gram based on vector space model. In *Information Theory and its Applications (ISITA), 2010 International Symposium on* (October 2010), pp. 106 –111.

[56] TAVANAI, A. Liver tissue classification using texture features. Bsc final year project, University of Leeds, 2010.

[57] TOYAMA, J., KUDO, M., AND IMAI, H. Probably correct k-nearest neighbor search in high dimensions. *Pattern Recognition 43*, 4 (2010), 1361 – 1372.

[58] UKTV. Good food. `http://uktv.co.uk/goodfood/homepage/sid/566`. Last Accessed: 22/08/2011.

[59] UL-QAYYUM, Z., COHN, A., AND KLIPPEL, A. Psychophysical evaluation for a qualitative semantic image categorisation and retrieval approach. In *Trends in Applied Intelligent Systems*, N. Garca-Pedrajas, F. Herrera, C. Fyfe, J. Bentez, and M. Ali, Eds., vol. 6098 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010, pp. 321–330.

[60] URAMOTO, N., MATSUZAWA, H., NAGANO, T., MURAKAMI, A., TAKEUCHI, H., AND TAKEDA, K. A text-mining system for knowledge discovery from biomedical documents. *IBM Syst. J. 43* (July 2004), 516–533.

[61] UZZAMAN, N., AND ALLEN, J. Extracting events and temporal expressions from text. In *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on* (September 2010), pp. 1 –8.

[62] WASHIO, T., AND MOTODA, H. State of the art of graph-based data mining. *ACM SIGKDD Explorations Newsletter 5* (July 2003), 59–68.

[63] WILSON, T., AND RAAIJMAKERS, S. Comparing word, character, and phoneme n-grams for subjective utterance recognition. In *INTERSPEECH-2008* (2008), pp. 1614–1617.

[64] WU, D.-S., AND LIANG, T. Zero anaphora resolution by case-based reasoning and pattern conceptualization. *Expert Systems with Applications 36*, 4 (2009), 7544 – 7551.

[65] YOSHIDA, K., MOTODA, H., AND INDURKHYA, N. Graph-based induction as a unified learning framework. *Applied Intelligence 4* (1994), 297–316.

[66] ZHOU, G., LI, J., FAN, J., AND ZHU, Q. Tree kernel-based semantic role labeling with enriched parse tree structure. *Information Processing & Management 47*, 3 (2011), 349–362.

Appendix A

Personal Reflection

One of the main challenges of the project was that the framework to achieve the goals of the project consisted of combining research on three areas of artificial intelligence - Natural language processing, knowledge representation and machine learning. The project revolved around addressing research problems in all these three areas within a very short duration as stipulated. Therefore, while the project introduced the a breadth of materials to study and investigate, it also involved knowing which of these methods would be worth pursuing. Pursuing these chosen areas in depth was equally challenging within a short duration.

The problem of not taking an in-depth approach was pointed out by my assessor in the earlier part of the project during the progress meeting. This advice helped me focus in-depth on aspects such as the pattern representation and learning aspect in greater detail.

Another challenge was that from the beginning of the project until the later stages, it was more or less unknown whether the methods chosen and the respective experiments would be fruitful. However, by constantly motivating myself, I tried to explore several possible options and at each stage found a step closer to the final goal. As I wish to continue my academic career by undertaking a PhD course, this project showed me the challenges that I may face in a PhD project. It has given me the confidence which I need in order to continue my academic career. It has also provided me with valuable experiences, which I may use to encounter many more challenges in the future.

In addition to the above, I would like to advise MSc students to write their reports as early as they can by initially placing the section titles in their report. By doing this they can add any findings

or ideas in the report in its appropriate section any-time during the project. I found this helpful in addition to keeping a logbook as when the main write-up starts, many of the main key ideas to write are already in the report which only requires further describing.

Finally, I would like to mention that I greatly enjoyed this project and I believe it taught me valuable skills for projects that I may have in the future.

Appendix B

Contribution

The following are the contributions made:

- ApplySVM: Applies a binary Support vector machine using LibSVM.
- AssignWeightedFeat: Uses the feature selection method of assigning wights to features.
- CombineFeatureSets: Combines feature sets in different ways.
- ConvertFeatureToSVMFormat: Converts a feature to SVM readable format.
- CreateRecipesnInteractionGraphs1: Sets all parameters and calls various programs obtained programs.
- EvaluateWeightedFeat: Evaluate weighted features
- ExtractGraphFeat_ObtainMainFeat: Calls the obtained methods for graph mining and learning.
- GetDistinctPatterns: Find the distinct patterns in a feature set.
- MostCommonVerb: Find the most common verbs in text.
- MultiClassSVM: Applies the Multi-class SVM in the LibSVM library.
- PreprocessingTagging: Applies the extraction of nouns and verbs.
- RecipeEncoder: Encodes a preprocessed recipe.
- SubWordSense: Substitutes verbs.
- RunAllPyCodes: Runs all python programs.
- OutPutHist: Output histograms for the bag of words method
- ClusterHist: Evaluate learning on the histograms for the bag of words method.

The following are obtained programs from Sridhar et al. [53]. The level of modification of each program is initially given followed by a brief description of the program

- EpisodesToLayeredGraphObjects (Slightly Modified): This program outputted the interaction graphs of encoded recipes.
- MineAndExtractGraphFeatures (Unaltered) Calls the program MineInterestingInteractionGraphsLevelWiseV1 for each class.
- MineInterestingInteractionGraphsLevelWiseV1 (Significantly Modified): Performes level wise mining and outputs all patterns of a class.
- RepresentsGraphsVectorSpace (Significantly Modified): Using the patterns found this program represents the recipes as a feature set.
- EvaluateSupervisedApproachKNN (Slightly Modified): Applies K-Nearest-Neighbour algorithm and outputs results.
- EvaluateClusteringApproach (Slightly Modified): Performs clustering using X-means.
- DisplayEvaluation (Significantly Modified): Outputs the results of the program as a figure. The modified version of this output is illustrated in Figure D.6
- graphmatch (Unaltered): finds the subgraphs in recipes.

In addition to the above, LibSVM [11] was used for applying the support vector machine and the MRMR feature selection method developed by Peng et al. [49].

Appendix C

Prototype Developement Plan

To implement the final prototype *P2* described in the 1.6, the following prototypes were set to be implemented:

Prototype 1: The first prototype was designed to determine if the framework can be applied to work with recipes by producing their interaction graphs. The following are tasks and functionalities set for this prototype.

- To create a manually preprocessed artificial corpus based on real recipes from websites.
- Encode the entities relations and temporal sequence of instructions within each recipe into numbers and place them in a matrix where the position of the encoded items is the same for all recipes.
- Obtain the relevant programs and modify them to use the encoded recipes.
- Use the modified programs and output the interaction graph for each recipe.

Prototype 1.1: A modification of prototype 1 by extending it through graph mining the graphs obtained from the recipes and using the features of each recipe for supervised and unsupervised learning. For this prototype, the following tasks and functionalities were set.

- Modify the remaining relevant programs to produce the feature set.
- Apply a supervised K-nearest-neighbour and an unsupervised X-means algorithm on the feature set.
- Implement a bag of words algorithm as a baseline, giving histograms of the recipes based on words.
- Increase the corpus size.

- Apply KNN and X-means on the produced histograms for classification.
- Obtain the results of the KNN algorithm using leave-one-out cross-validation and the outputted clusters from X-means.

Prototype 1.2: The main goal of prototype 1.2 was to create a corpus using unaltered text recipes from websites and to modify the prototype to work with the encoded recipes obtained from the raw recipes. The initial tasks and functionalities that were set to achieve the mentioned goal are presented below. However, the tasks and functionalities of this prototype increased in the later stages of the development of this prototype. More tasks will be described in Chapter 4.

- Collect text recipes which have been directly obtained from web pages without any changes being applied to them.
- Create a new algorithm that preprocesses the raw recipes by finding the part of speech tags of each word in the recipe and extracts the relations with their entities.
- Outputs of the preprocessing algorithm should be similar to the manually preprocessed files used in the previous prototypes for encoding.
- Change the representation of the matrix outputted by the encoding algorithm to implement the differences between the previous prototype preprocessed recipes and new preprocessed recipes.
- Create an algorithm to substitute a relation (verb) with another relation using word sense i.e. based on similarity in meaning.
- Modify all programs to run with the new representation.

Prototype 2: Prototype 2 was set to be a working system that fulfilled the aims and requirements of the project and in addition, implemented any methods or modifications that may benefit the system for obtaining the aim of the project. One of these modifications thought to improve the performance of classification and clustering was to modify the framework with the aim of finding more patterns in recipes in addition to those originally found. A more detailed description of these changes are provided in Chapter 4. The following are additional tasks set for prototype 2.

- Create a new corpus of recipes without any restrictions on choosing recipes.
- Further increase the domain knowledge.
- Modify the framework to create patterns from all verbs without considering common nouns.
- Combine the obtained feature sets from the bag of words and prototype 2 and apply classification and clustering.
- Implement the bag of words method using verbs and nouns.
- Evaluate using the new corpus without restrictions.

Appendix D

Figures

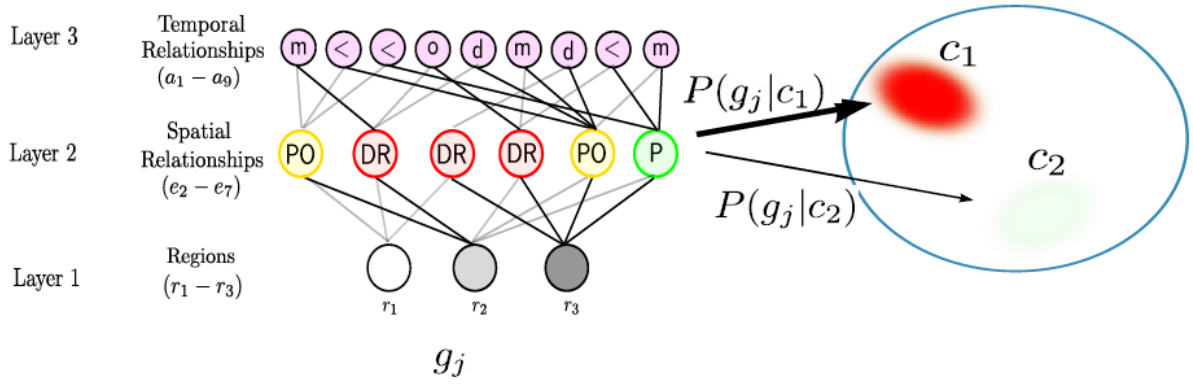


Figure D.1: Soft clustering an interaction graph of an event class. The event class has a higher probability of belonging to class $C1$ [53].

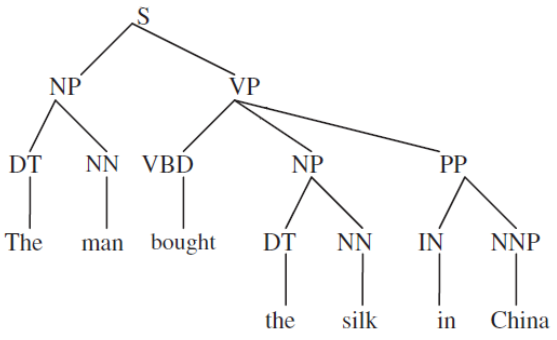


Figure D.2: An example of a parse tree providing the syntactic structure of the sentence ”The man bought the silk in China” [66].

Figure D.3: Project Schedule. The structure of the schedule used is similar and inspired by the schedule structure of [5].

Figure D.4: Revised Project Schedule. The structure of the schedule used is similar and inspired by the schedule structure of [5].

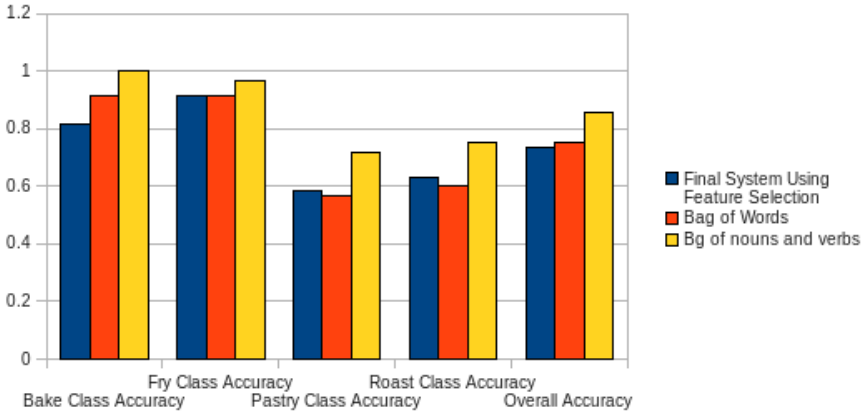


Figure D.5: Comparison using the bag of words method using nouns and verbs

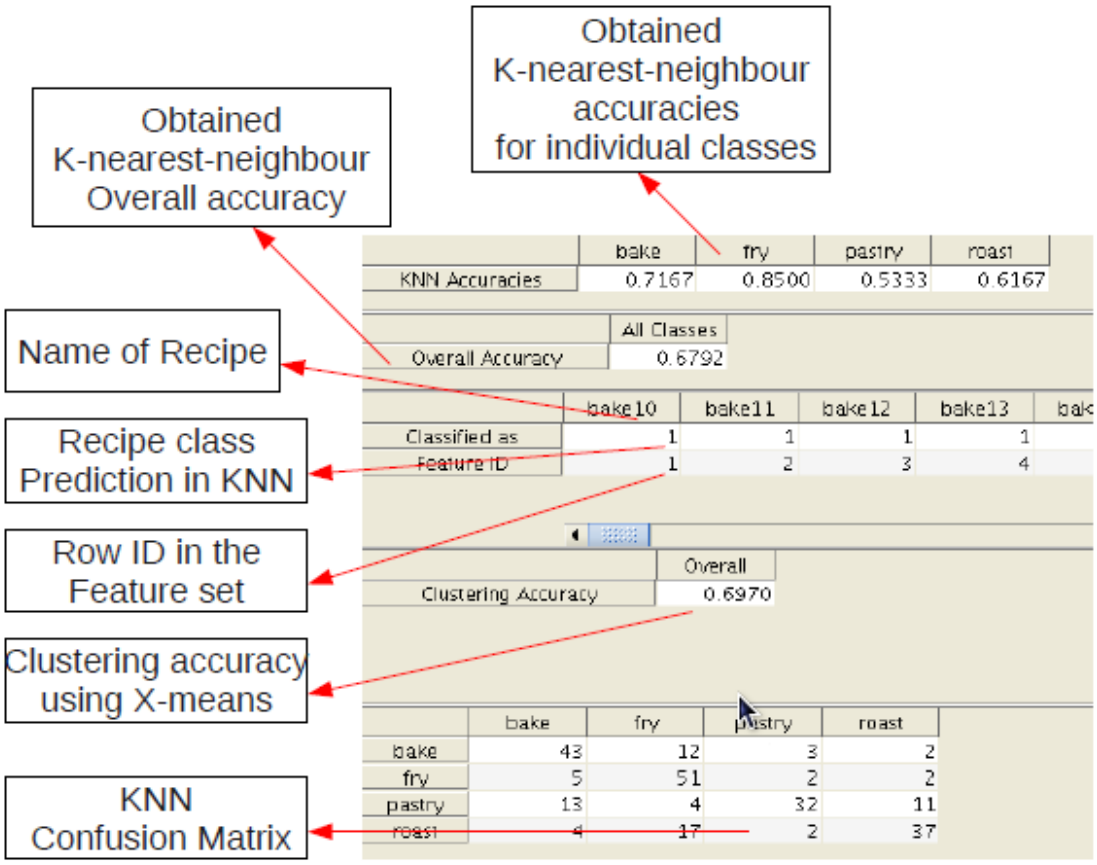


Figure D.6: A guide which details on what each component represents in the figures illustrating results in this report

Appendix E

Data

E.1 Example of a Raw Recipe

Chocolate Cake:

Preheat the oven to 180C/350F/Gas 4 and lin 2 x 12-hole fairy cake tins with paper cases. Whisk the eggs and sugar together in a bowl until light and fluffy. Carefully fold in the flour and butter. Pour the mixture carefully into the paper cases.

Bake the cakes for 15-20 minutes, or until golden-brown on top and a skewer inserted into one of the cakes comes out clean. Set aside to cool for 10 minutes on a wire rack before removing from the tin. To make the buttercream, beat the butter in a large bowl until soft. Add half of the icing sugar and beat until smooth.

Add the remaining icing sugar, cocoa powder and one tablespoon of the milk and beat until creamy. Beat in more milk if necessary to loosen the icing. Once the cakes are cool, spread the buttercream icing on top of the cakes. Decorate the cakes with the chocolate buttons.

E.2 Recipe Without Many Patterns

Raw Form Pan-fried salmon with broccoli:
Melt the butter in a frying pan. Add the salmon skin-side down, season with salt and freshly ground black pepper and squeeze over a little lemon juice. Fry for 2-3 minutes, then turn the salmon over and fry for a further 1-2 minutes, or until cooked through. Squeeze over a little more lemon juice and taste to check the seasoning. Remove from the pan and set aside. Place the broccoli into a pan of boiling salted water and boil for 3-4 minutes, or until al dente. Drain the broccoli. To serve, pile the broccoli onto a serving plate and top with the pan-fried salmon. Garnish with lemon wedges.

preprocessed form Pan-fried salmon with broccoli melt butter pan add salmon season salt pepper lemon fry salmon place broccoli pan water

E.3 Recipe Provided by Participant

Participant recipe:
Chocolate Cake
Preheat the oven to 180 degrees. Put a cup of flour and 1/2 cup of brown sugar in a bowl and stir with 200 grams butter.
Add an egg with a teaspoon of baking powder, 200 ml milk, one teaspoon of virgin oil, half a teaspoon vanilla and whisk everything. Melt some chocolate bits and add cracked walnuts and mix. Pour everything in a tray and place it in the oven.
Bake in the oven for about 20 minutes. Then leave to cool.

Most similar recipe found:
Chocolate Fairy Cakes
Preheat the oven to 180C/350F/Gas 4 and lin 2 x 12-hole fairy cake tins with paper cases.
Whisk the eggs and sugar together in a bowl until light and fluffy.
Carefully fold in the flour and butter.
Pour the mixture carefully into the paper cases.
Bake the cakes for 15-20 minutes, or until golden-brown on top and a skewer inserted into one of the cakes comes out clean. Set aside to cool for 10 minutes on a wire rack before removing from the tin.
To make the buttercream, beat the butter in a large bowl until soft. Add half of the icing sugar and beat until smooth.
Add the remaining icing sugar, cocoa powder and one tablespoon of the milk and beat until creamy. Beat in more milk if necessary to loosen the icing.
Once the cakes are cool, spread the buttercream icing on top of the cakes. Decorate the cakes with the chocolate buttons.

E.4 Additional Recipes To Correctly Classify Misclassified Recipes

Misclassified *fry* class recipe:
Southern Fried Chicken
cut chicken
wash chicken
heat oil
mix eggs water

add sauce sauce
whip mixture
add salt pepper flour garlic powder
dip chicken egg
place chicken oil
fry chicken oil
sprinkle salt

Additional *fry* class recipe:

Southern Fried Chicken
beat eggs water
add sauce egg mixture orange
combine flour pepper
season chicken
dip chicken egg flour mixture
heat oil pot
fill pot oil
place chicken oil
fry chicken oil

Misclassified *bake* class recipe:

Almond Biscuits
preheat oven
put butter syrup sugar pan
heat sugar
stir almonds flour
place mixture
bake biscuits

Additional *bake* class recipe:

Almond Biscuits
preheat oven
melt butter
put sugar pan
add flour almonds
heat sugar
bake oven

E.5 Noun & Verb Dictionaries

Verb Dictionary	Noun Dictionary
peel 1070	potatoes 1
add 1071	steak 2
heat 1072	oil 3
fry 1073	pan 4
cut 1074	onions 5
sprinkle 1075	butter 6
put 1076	salt 7
scramble 1077	cornstarch 8
mix 1078	salmon 9
slice 1079	wok 10
wash 1080	pepper 11
stir 1081	egg 12
boil 1082	rice 13
pour 1083	vegetables 14
crush 1084	sauce 15
divide 1085	pepper 16
bake 1086	bananas 17
whisk 1087	flour 18
sieve 1088	sugar 19
grate 1089	milk 20
roll 1090	powder 21
beat 1091	mug 22
spread 1092	mustard 23
whip 1093	cocoa 24
blend 1094	whiskey 25
rest 1095	lemons 26
rinse 1096	cinnamon 27
preheat 1097	ginger 28
melt 1098	mallet 29
place 1099	vanilla 30
fold 1100	syrup 31
sift 1101	coffee 32
stir-fry 1102	tea 33
cream 1103	water 34

Table E.1: Subsets of the originial dictionaries used for the dictionary lookup method