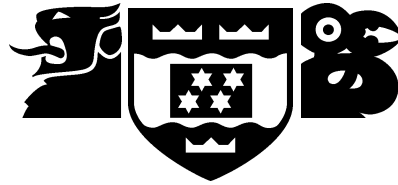


VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Mathematics, Statistics and Computer
Science
Te Kura Tatau

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@mcs.vuw.ac.nz

Understanding Spoken Commands for the Marvin Robot

Hayden Smith

Supervisor: Dr. Peter Andreae and A/Prof. Dale
Carnegie

Submitted in partial fulfilment of the requirements for
Bachelor of Science with Honours in Computer Science.

Abstract

The Marvin robot is intended to act as a security guard for the School of Chemical and Physical Sciences capable of performing a number of simple tasks and responding to a number of simple queries via a voice recognition mechanism.

Interaction via spoken English requires that Marvin be capable of understanding commands and queries that are not necessarily complete or correct. The output from the voice recognition software might also incorrectly interpret utterances and so our system must take this into account.

Acknowledgements

There are many I people I would like to thank for their contributions to this project:

- Peter Andreae and Dale Carnegie for their good humour, patience and helpfulness throughout the duration of the project.
- Stacey Walker for proof-reading this report.
- My fellow Memphis graduates for their support with \LaTeX , providing useful ideas and a lot of laughs.
- Rachel Perry for her assistance with the poster for this project.
- The crews of my Deloitte and Dragonflys dragonboat teams, my colleagues at Silver-Stripe and friends who provided many of the utterances used to train and test the $M\mu$ system.
- My Mum and Dad for their support through my school years.

Contents

1	Introduction	1
1.1	The $M\mu$ system	1
1.2	Understanding a sentence	1
1.3	Impact of speech recognition software output as system input	2
1.4	Outline of this report	3
2	Relevant ideas from the literature	5
2.1	“Experiences with Commercial Telephone-based Dialogue Systems”	5
2.2	“Cooperative Model Based Language Understanding in Dialogue”	6
2.3	“Natural Language Understanding Using Statistical Machine Translation”	8
2.4	“Fertility Models for Statistical Natural Language Understanding”	9
2.5	“The Media Equation”	9
2.6	Summary of relevant papers	9
3	The $M\mu$ system	11
3.1	Fast matching using a deterministic finite state automaton	11
3.1.1	Adding synonyms from the WordNet database during training	11
3.1.2	Parameterising known utterances	12
3.1.3	Using an augmented transition network to facilitate parameterised utterances	12
3.1.4	Modifying direct lookups to handle a single word out of place	14
3.1.5	Near-match graph traversal	14
3.2	Statistical methods to handle ‘noisy’ input	15
3.2.1	Naïve Bayes statistical analysis for both command and parameters	15
3.2.2	Naïve Bayes using effective hashmaps	16
3.3	Combining results of the two modules	17
3.4	Implementation	17
3.5	‘Chatterbox’ for web-based sample utterance collection	18
3.5.1	Raw retrieved samples	18
3.5.2	Ambiguities, spelling errors, and insufficient information in collected samples	18
3.5.3	Training data as direct and parameterised utterances	19
4	Results	23
4.1	Results with raw samples*	23
4.1.1	Results with corrected samples*	23

5	Conclusion	25
5.1	Effectiveness of these approaches in the real world	25
5.2	Impact of speech recognition software on accuracy	25
5.2.1	Effect of the environment on speech recognition	25
5.3	Future work	26
5.3.1	Improving accuracy	26
5.3.2	Expanding Marvin's lexicon	27

Chapter 1

Introduction

MARVIN (Mobile Autonomous Robotic Vehicle for Indoor Navigation) is a robotic security guard developed by the School of Chemical and Physical Sciences' Dale Carnegie.

The goal of the project is to develop a system capable of identifying commands delivered to the MARVIN as series of sentences, or utterances, spoken by users. A third party speech recognition system will convert the speech to text, as the speech recognition component is outside the scope of this project. The system will need be able to handle instructions articulated in normal, everyday New Zealand English. The system will be used to understand commands given to the MARVIN robot.

1.1 The $M\mu$ system

There are several significant components required for an autonomous robot: path finding, goal planning, speech recognition, dialogue systems and learning systems. The scope of this project, however, is limited to identifying spoken commands.

The system, $M\mu$ (MARVIN Understander), will have to be able to identify three kinds of commands:

- Requests for directions to a location within the Laby building. For example: *"Where is the school office?"*
- Requests to accompany a visitor to a location within the Laby building. For example: *"Can you show me to the exit?"*
- Queries about a particular staff member within the School of Chemical and Physical Sciences. For example: *"Who is Dale Carnegie?"*

The system should then output the appropriate command to execute along with the appropriate parameter.

Although the scope of the project does not include learning support, the system should be designed so that it would support on-line learning of new utterances.

1.2 Understanding a sentence

This project requires a natural language understanding system. In this case, the system will be responsible for taking English sentences as strings of words and determining an appropriate command to execute based on that input.

Natural language processing would be a trivial problem if not for the many possible ways to articulate a meaning from a sentence in natural languages. Consider the following sentences:

- “Where is the school office?”
- “How do I get to the school office?”
- “Where is the administration handled?”

All of the above are requests for directions to the school office – they all have the same meaning.

A visitor might not know the name of the person or place that they are looking for, and instead might describe a person or place in terms of their function or position. For example, a visitor searching for Dale Carnegie might ask: “*Where might I find a robotics lecturer?*” rather than “*Where is Laby 502?*” The sentences received by the system must understand the sentence in order to identify its meaning.

There are several parts of a sentence that must be considered in order to understand it: the vocabulary or lexicon used, the structure of the sentence and the context of the dialogue.

The lexicon used in the sentence is vital to the underlying meaning of the sentence. Each word has an associated meaning and, combined with the other words in the sentence, contributes to the meaning of the entire sentence. Therefore, each word must be considered when attempting to evaluate the meaning of the sentence.

However not all of the words in the sentence will have an impact of the meaning of the sentence. Consider the two sentences: “*Can you take me to the school office?*” and “*Can you please take me to the school office?*”. The use of the word “please” is simply manners on the part of the speaker but contributes little to the meaning of the sentence – a request to be accompanied to the school office. The system will therefore have to attach some meaning to each of the words.

While some words might not contribute to the meaning of a sentence, the structure will always affect its interpretation. Consider the two sentences: “*Can you take me from the office to Laby 502?*” and “*Can you take me to the office from Laby 502?*” Both sentences have exactly the same set of words. However, swapping the prepositions “to” and “from” give the two sentences different meanings. It is therefore important that the system consider the structure of the sentence when parsing.

The structure of a sentence is an important part of grammatically correct written and spoken English. However everyday speech is much less formal. People tend to use slang terms, mismatch tenses and omit unnecessary words altogether. The $M\mu$ system will need to be flexible when handling the query as it will often be grammatically incorrect, but comprehensible. For this reason a strict, traditional parser will likely be unsuitable for the $M\mu$ system.

A sentence can have different meanings in different contexts, and this can only be discovered by looking at the entire conversation. In this case, examining the context would be a good way to deal with situations in which the system does not understand a sentence and would seek clarification. This, however, is outside the scope of this project.

1.3 Impact of speech recognition software output as system input

MARVIN will interact with visitors via a speech recognition system, so the $M\mu$ system will either have to use the output of a third party speech recognition system, or provide speech

recognition directly. Speech recognition requires a different set of techniques and is outside the scope of this project.

The third party system will likely introduce errors into the $M\mu$ system's input. This is more likely in this situation as most speech recognition systems are developed for dictation tasks and are used with a high quality headset and are trained for a single user. In this case, the software will not be analysing the speech of a single user over an extended period of time, but a variety of users for very short durations. The visitor will also be further away from MARVIN's microphone, so background noise will have a greater impact. The $M\mu$ system will have to account for errors in the input.

1.4 Outline of this report

Chapter two examines some existing approaches, both technological and sociological, and how these approaches might influence the design and implementation of the system.

Chapter three provides details on the $M\mu$ system and its individual components. It also outlines how the system deals with the problems inherent in natural language processing. It also outlines the data set used to train and test the $M\mu$ system.

Chapter four covers the results of our testing of the system.

Chapter five discusses the effectiveness of the system in theory and expectations of the system's performance in practise.

Chapter six covers other possible extensions to the $M\mu$ system.

Chapter 2

Relevant ideas from the literature

Natural language driven interfaces are very intuitive for many applications, so there are many systems already in use that have a similar approach.

This chapter outlines the papers relevant to the design or implementation of the $M\mu$ system. Each section covers how the key ideas presented in the paper are applicable to the $M\mu$ system.

2.1 “Experiences with Commercial Telephone-based Dialogue Systems”

[4], unlike most of the others, is not a technical paper. It only mentions briefly the techniques used by the Sympalog system, an automated, telephone-based dialogue system. The paper’s main focus is how the system performed in several different organisations. Three applications are mentioned: A car rental agency’s helpdesk, a soccer information system and a movie timetable helpline.

All systems assist the novice user by directing the dialogue of the system. This is a possible strategy for Marvin for more complex queries in the future, but for the initial scope of the system the queries are not likely to be complex enough to warrant a directed dialogue. A directed dialogue might still be useful when requesting clarification of an ambiguous or unclear request.

The Sympalog system also allow the user to interrupt the dialogue when the system is responding to the user. This usually indicates that the user is frustrated or in a hurry. The system should react appropriately to interruptions by the user. If the system is seeking clarification of an ambiguous parse, and is interrupted, the interruption will likely contain extra information that may help clarify the user’s request. During Marvin’s deployment, logging interruptions with the assumed request may help provide further insight into which parses are ambiguous or incorrect.

The paper covers several features or aspects that would help to create a more successful telephone-based dialogue system. Some of these points are applicable to the $M\mu$ system:

- The Sympalog systems use a dialogue-based approach, rather than a menu-driven approach. The conversation nature should take previous correspondence into account. This is natural in normal face-to-face interaction. This can help the system establish context. A telephone dialogue system has the two advantages that we will not have with the MARVIN robot: A definitive manner with which to associate an utterance with a conversation, and a reasonably confident history of past dialogues, especially with a cellphone number.

In the former case, the telephone conversation is established between the system and a known connection to a known telephone number. MARVIN will typically handle requests that do not require extended dialogue. Utterances received by MARVIN in a short period of time can be considered part of the same conversation. A more accurate way would be to match the voice of the user, as several users might approach in a short space of time. This approach relies on the dictation software being capable of matching voices with queries.

In the latter case, the system could associate previous conversations with the phone number, if it is applicable to the domain. This will be difficult if not impossible with the Marvin system unless the software is capable of matching voices to previous dialogues. Even if this is possible, it is highly unlikely the same user will have the same requests.

- While not related to any system in particular, the paper suggests that input and output be analysed in order to investigate how people interact with the system. *The Media Equation* covers how people interact with technology in more detail. Section 3.5 describes how sample utterances for training and testing were collected.
- The paper also suggests matching words that are spelled differently but are pronounced in the same way. This is an important factor to consider in the design of the system. The paper also points out that pronunciation of names can also be problematic for two reasons: The dictation software might misinterpret the name depending on the context (if it has that capability), and a user might not be able to correctly pronounce a name when asking about a person. The system will have to be able to try to find the most likely name, if mispronounced.
- The paper also suggests that directed dialogue systems will receive desirable responses if they are polite. This makes sense as a user will likely become hostile towards Marvin if his dialogue is abrasive. It suggests that a request for clarification; "I'm sorry. I didn't hear you properly. Could you repeat that?" will be better received than "Speak loudly and clearly".
- The paper insists that a response time of two seconds or more is too long and will confuse the user. This is a very valid point as the user may infer that the $M\mu$ system is having difficulty understanding the request, when in fact it is processing using a processor-intensive algorithm. The user might attempt to rephrase the request unnecessarily, and this may confuse the $M\mu$ system further.

While the paper did not provide any technical approaches to the system, it did provide a number of very valuable non-technical considerations that may lead, if not to improved accuracy, to improved user interaction. This may indirectly improve accuracy by making the user more tolerant of the system's inaccuracies. The points outlined in this paper will feature highly in the design of the prototypes.

2.2 "Cooperative Model Based Language Understanding in Dialogue"

[2] outlines a similar strategy to the overall design I had chosen for the $M\mu$ system; and goes into more technical detail for each algorithm. The paper describes a system with an accurate but inflexible finite state machine model and a more flexible but less accurate statistical learning model. The system was deployed for the Mission Rehearsal Exercise, a system to

help army recruits experience the real-world scenarios. This system is unlikely to receive grammatically complete sentences. The $M\mu$ system will have a similar problem.

Feng also explains the "frames" or formal semantic representation that is created from the input sentence. The frames decompose a sentence into its type (question, action and proposition) along with extra information, including but not limited to; the time frame, subject of the utterance and negation (if any).

The paper describes two finite state machine models: a series model and a single model. The series model uses predefined contexts to match predefined slots, or sets of key words. It will end gracefully with an empty frame if no key words are found, but it does not take into account words that have different meanings in different contexts. The single model provides a single, large finite state machine to parse the sentences. If the sentence exists in the machine, it will always return the correct result. However, this model requires that all sentences in the domain be known before the state machine is constructed, and it must be constructed by hand, which is time-consuming and tedious.

Having outlined the failings of a direct match with a finite state automaton, Feng goes on to introduce the Naïve Bayes statistical learning model. Naïve Bayes expresses the probability that the word W has meaning M as:

$$P(M|W) = \frac{P(W|M)P(M)}{P(W)} \quad (2.1)$$

where $P(W|M)$ is the probability that meaning M is expressed by the word W , $P(M)$ is the probability that any word has meaning M and $P(W)$ is the probability of word W occurring in the sentence.

The paper recognises that a word's meaning is combination of the word's meaning and the context in which it appears. Feng briefly outlines a Hidden Markov model approach. This approach determines the meaning of the word, based on the probability of the meaning and the words preceding the word in the sentence.

This approach can lead to very noisy results, as Feng points out. The paper outlines the methods used to control large variations in meaning. The first ignores any meaning obtained by a large jump in probability given a new word in the sentence. The second groups slot-meaning pairs and takes the first meaning.

The above approaches are combined together to produce a confidence score for the sentence from each model. If the sentence exists in the single finite state model, then the system will immediately produce the correct frame with a confidence score of 1.0. Failing that, the sentence can be run through the series finite state machine and the statistical learning model to try to approximate the frame. Moreover, the output of the statistical learning model can be passed to the series finite state machine in order to reduce the inaccuracies of the statistical model.

Feng covers the results of experiments with the system in this paper. As expected, both the single and series finite state machines have 100% accuracy when passed a known sentence. The statistical model produced frames with 85% precision and 95% recall. The results for new patterns from a blind test are not empirical for the finite state machines, resulting in a partial frame from the series model and an empty frame from the single model. The statistical model produced frames with 75% precision and 92% recall. The reduced accuracy of the system on the blind set is to be expected.

Feng also lists the precision and recall of the system at points where increasing portions of the blind set are added to the system. As expected, the precision and recall improve significantly. Given Marvin will be operating autonomously the ability to learn new commands without supervision will be a great asset. Although the recall and precision for this system

seems low for the statistical learning model, the finite state machines would help to reduce these error rates for the system as a whole.

The system mentioned in this paper faces similar difficulties that our system will face. This paper provides a useful confirmation of the earlier decision. The combined use of accurate finite state automata and flexible statistical methods is a feature that the $M\mu$ system employ to improve understanding. The $M\mu$ system will use a different finite state automata to the system outlined in this paper.

2.3 “Natural Language Understanding Using Statistical Machine Translation”

[3] takes a purely statistical approach to the problem of natural language understanding. It argues that stochastic grammars are inflexible and are not easily reused. Stochastic grammars are often written by hand and this would make extending the Marvin parser difficult.

The paper introduces alignment templates, and cites an earlier work by some of the authors of the paper: “Alignment templates have proven to be very effective in statistical machine translation because they allow many-to-many alignments between source and target words. [5]” This paper also points out, like many of the others, that a single mapping on a word is insufficient to infer meaning. The example used shows ‘Cologne’ used as the origin of a journey, but this is only apparent in the context: ‘from Cologne’.

The alignment templates provide two levels: a top level to align phrases and a bottom level to align words within a phrase. The paper points out that three probabilities must be trained by the system, but does not cover how they are trained, only how they can be decomposed. The details of the training are in a referenced paper which I have not read.

The paper measures success by counting the number of words mapped to the correct concept or meaning. Words were also represented by word sets where applicable. The use of the city name set was one example, as this provides the required information without requiring any changes to the structure of the sentence.

Macherey *et al* define several metrics to measure the accuracy of the system: the concept error rate, the sentence error rate and the concept–alignment error rate. The sentence error rate is the simplest and most appropriate metric for us to consider, as it corresponds to the overall error rate that I will use to measure the accuracy of our system. This error rate is simply the number of incorrectly translated sentences. The results are listed with respect to the underlying algorithms from which the actual algorithm is derived. This demonstrates that the algorithm was worthwhile as its performance is better than the simpler algorithms. The error rate of the final algorithm was 4.3%, or a success rate of 95.7%.

The notion of two-level alignment templates seems to be a very good approach to the problem of matching a formal meaning language to a natural language input, taking into account instances when words are out of order. I will use a similar system in order to match the output of the dictation software to Marvin’s possible actions when the utterance is not already found in the system’s cache.

2.4 “Fertility Models for Statistical Natural Language Understanding”

[6] mentions a system that must be trained with an annotated corpus. It uses an EM¹ algorithm to learn the probabilities that a given “clump” or “clumps” are associated with a word in the formal language. A clump is a word or set of consecutive words that are associated with a prior word in the sentence.

The paper does not explicitly say how the algorithm handles words grammatically out of place. It is important that the $M\mu$ system can handle words out of place in an utterance, so this algorithm might not be appropriate for the $M\mu$ system. The nature of EM algorithms require that they be run several times until the variance of the results are reduced below a certain threshold. With a large corpus of words, this might take a long time and [4] suggests it is important to have a timely response.

A possible application of this kind of learning could be learning or training the $M\mu$ system to associate particular words with particular subjects (associating secretaries or secretary with the office for example). This would almost certainly have to be off line, as this information is probably not likely to come up in practice, and learning to associate a new term with a given location would only be possible if the user rephrases the question in such a way that the $M\mu$ system can recognise the command.

The system was tested by using a program that constructed SQL queries against an ARPA test database. The highest accuracy was 83.04% when using an annotated corpus, and 79.91% without. These values are too low to meet our goal accuracy and would need some sort of additional checking in place to improve accuracy. The use of clumps of words was also suggested in [3] and would be a useful feature in the statistical implementation to reduce individual words in the utterance to words in the formal language.

2.5 “The Media Equation”

[7] is quite different to the other sources and it covers the psychology and sociology of people as they interact with different media.

Reeves and Nass’s research concludes that people will often react to media in a way that they would normally act with another person. The book includes an example in which an entire audience personalised a puppet, directing questions to it rather than the puppeteer.

Taking this research into consideration, it would seem that it is highly likely that people will use natural dialogue when addressing MARVIN, rather than trying to use language that they think a robot may understand. This is an important justification for the project in general.

2.6 Summary of relevant papers

The papers outlined above have provided some excellent starting points for the design of the $M\mu$ system. Not only have the papers covered some existing technical approaches, they have also provided some sociological and psychological insight that may well prove invaluable in practice.

¹An Expectation-Maximisation algorithm estimates values for hidden variables and performs the calculation with the hidden variables and uses the output to adjust its estimates for the hidden values until variance drops below some threshold

Chapter 3

The $M\mu$ system

The $M\mu$ system will have to consider both the vocabulary and partial structure of a sentence in order to understand its meaning. It will also have to take into consideration the likelihood of errors introduced by the speech recognition software. It should also provide support for a learning mechanism that adds to the knowledge base through interactions with visitors.

The $M\mu$ system uses a high-level approach very similar to the approach outlined in [2]. It has two complimentary methods: a finite state automaton to store complete phrases known to the system, and a Naïve Bayes module to parse utterances that are previously unknown, or have little structure.

The $M\mu$ system attempts to use the finite state automaton to match the utterance, or find a near-match. If no result is found in the finite state automaton, then the utterance is passed on to the Naïve Bayes module.

3.1 Fast matching using a deterministic finite state automaton

The direct lookup uses a large trie – a directed tree with each node having an arbitrary number of children. In contrast with a more general prefix tree, the trie used in $M\mu$ stores the next word in the utterance at each node. This way, the path to a node at depth n represents the first n words in the utterance. The transition to the next node in the trie is determined by looking at the next word in the utterance and using the transition from the current node that for that word.

This module is instance-based, storing each individual utterance that is known by the system in advance. This module is designed to quickly match utterances that are already known to the $M\mu$ system, as many of the phrases known to the system are likely to be found in many of the input utterances.

This approach however relies on the $M\mu$ system receiving completely accurate input from the speech recognition system. Given that most speech recognition systems are designed for use with a headset rather than a microphone at a distance, errors in the input are highly likely. It also relies on having all possible utterances stored in the system. The $M\mu$ system has three extensions to trie: caching synonyms for each word when loading utterances into the trie, separating the noun phrases in utterances from the utterance and modifying the search to include utterances the trie that have only a one word difference.

3.1.1 Adding synonyms from the WordNet database during training

The English language has a rich vocabulary, with many words having the same meaning. Princeton University's WordNet database [1] has a large corpus of English words and con-

tains a broad network of relations between words, including synonyms.

By locating synonyms for each word during training, the lexicon of the system that is stored in the trie can be expanded without having to manually annotate the training data, nor add extra training utterances.

3.1.2 Parameterising known utterances

Many of the utterances known to the system and stored in the trie will have the same underlying command with a varying noun phrase representing the person or location of interest. For example, consider the sentences:

- “How do I get to the school office?”
- “How do I get to the first year chem lab?”
- “How do I get to Dale’s office?”

All of these sentences are requests for directions to different locations. Separating the noun phrase, or parameter, from the rest of the sentence allows greater generalisation of the stored utterances.

By representing locations as metawords in an utterance, we can model the request for directions generically as: “*Where is $\langle location \rangle$?*” and “*How do I get to $\langle location \rangle$?*”, and then substitute locations in order to increase the vocabulary. This substitution can be made either when the known phrases are loaded into the lexicon in the system, or dynamically when the utterance is parsed.

The former option is trivial to implement, but would drastically increase the size of the trie, which would increase the search space and slow the search down significantly. The latter option reduces the search space and conveniently categorises the noun phrases into different classes that can be referred to from different utterances. This allows the system to match utterances that have not explicitly been added to the system.

Storing known phrases in the trie

Utterances that share a common sequence of words at the beginning of the sentence will share a path in the trie until the two utterances differ. For simple requests and queries this is a simple and effective optimisation, as they will often share a similar structure and branch later, when describing the object in the sentence.

Figure 3.1 represents the state of the trie after the contents of the training file in figure 3.5 are entered.

3.1.3 Using an augmented transition network to facilitate parameterised utterances

As mentioned above (section 3.1.2), parameters can be parsed at runtime easily if they are stored together, and can be parsed as easily as existing utterances. We use an augmented transition network to separate groups of common parameters from the utterances in which they appear.

Rather than a singly-connected trie, an augmented transition network is a set of singly-connected components, with each component representing a distinct classification or grouping. In our case, we use an component to store utterances and metawords to reference the

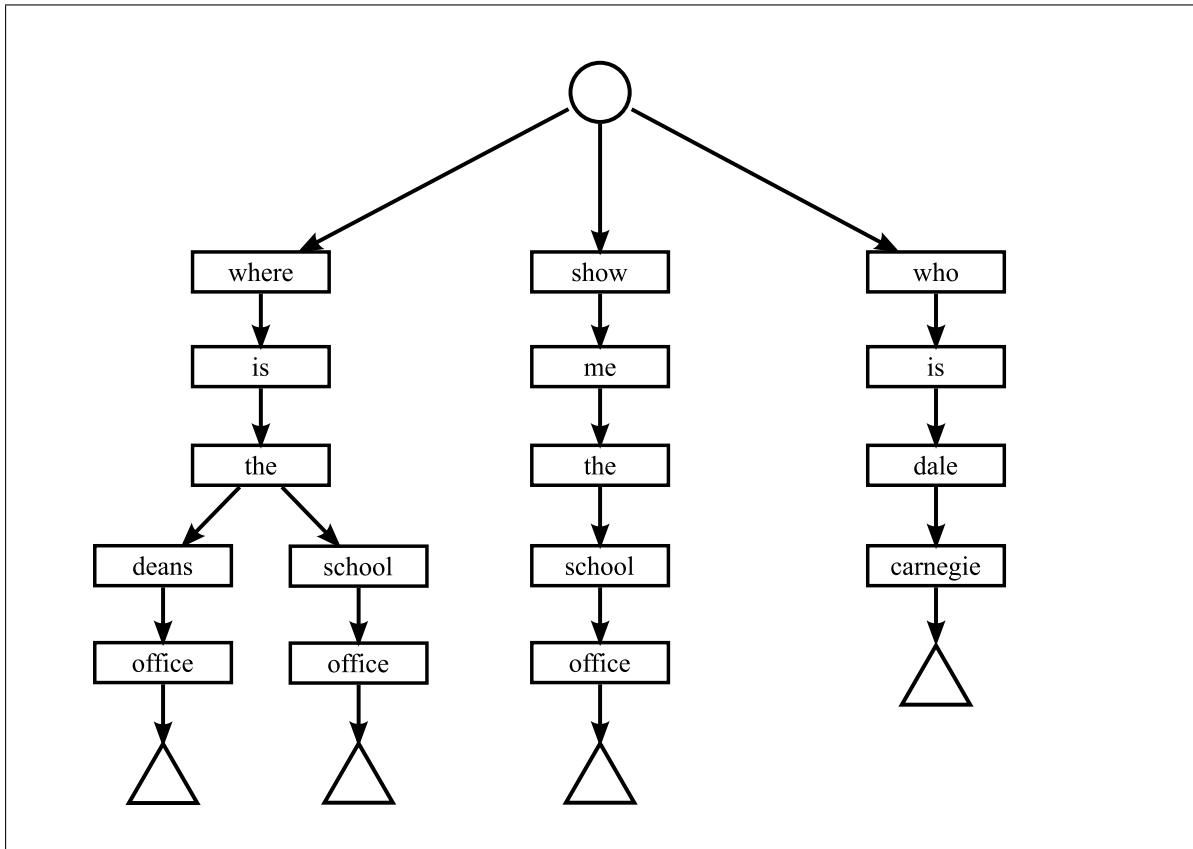


Figure 3.1: The trie after loading the training file in figure 3.5.

classes of the parameters. Parameters of a class are then all grouped into a single component. Locations and people, for example, are treated as two distinct classes. This reduces the number of nodes in the trie further thereby reducing the time required to search the trie.

When traversing the trie, a node can reference a meta-transition by a metaword. The component corresponding to the metaword is then traversed with the remaining words in the utterance and a stack with the most recently encountered metaword on top (this allows classes to be parameterised also). When the class component is traversed and an end node is encountered, the parameter stored at the last node is returned and mapped against the metaword. This is later used when parsing the action to discover the parameter for the action that the utterance corresponds to.

Figure 3.2 shows the above trie separated into the structure trie at the top and the two parameter tries: “locations” and “people”. As well as the reduced storage requirements for very large sets of commonly used noun phrases, these separate tries could be useful for a parser that attempts to break an utterance down into parts of speech. The “people” trie could even be used in conjunction with tries that use other languages.

This approach however requires that the noun phrases in the sentences in the training set be marked so that the system can add them to the correct tries. Currently, this must be done by hand.

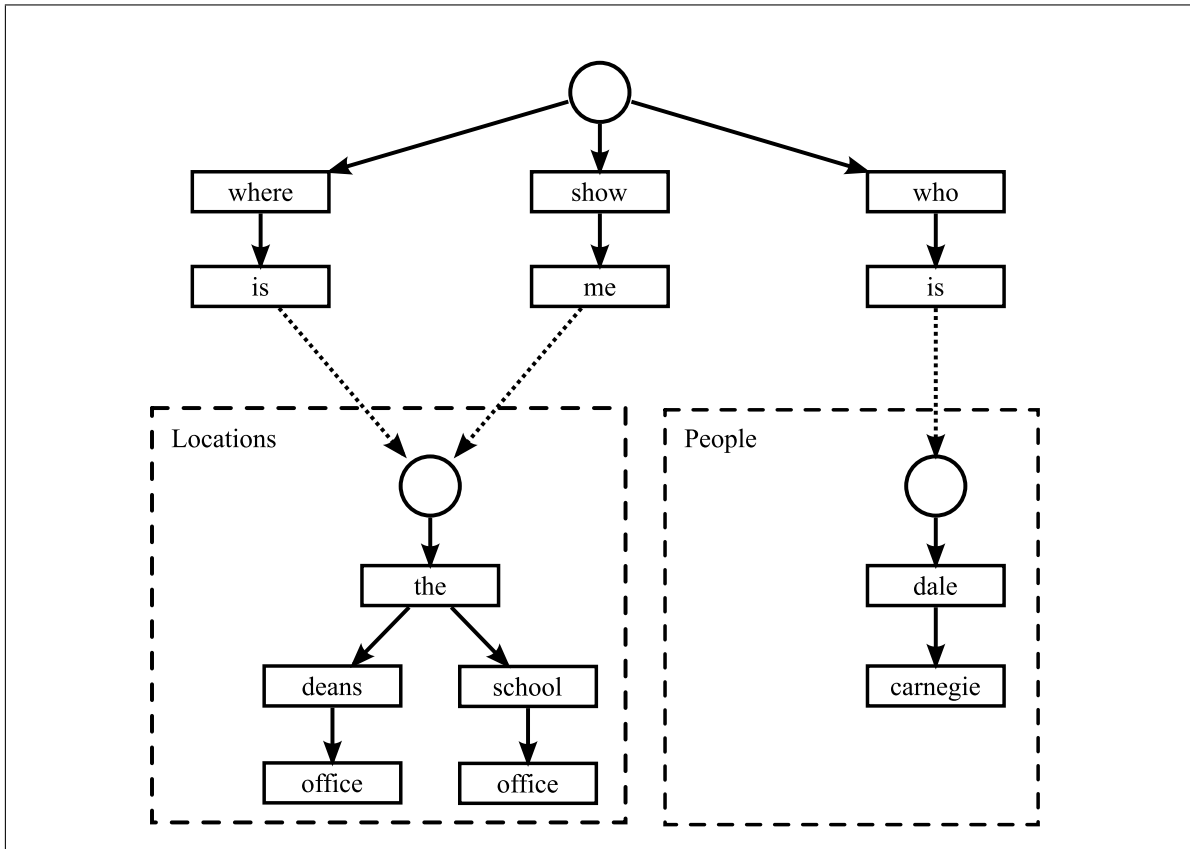


Figure 3.2: The trie in figure 3.1 with the common noun phrases separated into a “locations” trie and a “people” trie.

3.1.4 Modifying direct lookups to handle a single word out of place

Our system relies on input from third-party speech recognition software. At the time of writing, this will be Dragon NaturallySpeaking. Dragon NaturallySpeaking is designed for use with a user’s vocal profile and is trained by the user to their voice and accent. It is also used with a headset that puts the microphone near the user’s mouth. This setup helps improve the software’s accuracy.

We will not have the two advantages that the training and microphone offer. Most users will interact with MARVIN briefly and as such the system cannot be trained to one visitor’s voice. The visitors will also be further from the microphone and so the spoken commands will be heard along with noise from the surrounding environment. This increases the chances of misheard or misinterpreted words significantly.

In order to cope with these situations, we have modified the direct match traversal to handle situations in which the speech recognition software has incorrectly matched a word, added an extra word, or missed a word altogether. This requires a modified depth-first search of the trie containing the known utterances.

3.1.5 Near-match graph traversal

Complete, known utterances are stored in the system in a trie structure. This structure combines reduced storage requirements with rapid traversal. Future work could also use the structure to infer interchangeable noun phrases by recombining paths in the trie after they have branched closer to the root node. Figure 3.3 outlines the leave-one-out search for the

trie.

Traversing the augmented transition network

Direct traversal of the basic trie is trivial. However, traversal of the augmented transition network including all matches differing from the input utterance by one word is not. Figure 3.3 is modified slightly to include transitions to the components containing the different noun phrases.

3.2 Statistical methods to handle 'noisy' input

As mentioned in the introduction (section 1.2), conversational English tends to be informal, and is often grammatically incorrect. Words that add little to the meaning, or that can be inferred from the sentence itself, are often skipped.

In these situations a strict grammar parser will fail as the sentence itself, while easily understood by a fluent human audience, is not formally correct English. These situations will also be the norm rather than the exception.

In order to handle 'noisy' input of this kind a statistical model will be employed. Typically these statistical models use a combination of the probability of a word in the utterance being heard by the system with the probability that the word is associated with a particular meaning. In this case, a Naïve Bayes approach was taken and is described in more detail below (section 3.2.1).

3.2.1 Naïve Bayes statistical analysis for both command and parameters

Naïve Bayes is a statistical method based around Bayesian inference. The formula below forms the basis of Bayesian inference:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \quad (3.1)$$

where $P(H|E)$ is the likelihood of the hypothesis H given the evidence E . This likelihood is necessary to determine the probability of the hypothesis given the evidence. However, this is unknown, so the Bayes formula must be used to calculate the probability. $P(H)$ and $P(E)$ are the probabilities if the hypothesis and evidence occurring in the data respectively. They are known as priors.

For our problem, the evidence E can be represented as a sequence of words, similar to a term vector¹ \vec{w} . This gives the formula:

$$P(A|\vec{w}) = \frac{P(\vec{w}|A)P(A)}{P(\vec{w})} \quad (3.2)$$

where $P(A|\vec{w})$ is the probability of performing some action given the sentence defined by the term vector \vec{w} . Expanding the equation further yields:

$$P(A|\vec{w}) = \frac{\prod_{i=1}^n P(w_i|A)P(A)}{\sum_{j=1}^m \prod_{i=1}^n P(w_i|A_j)P(A_j)} \quad (3.3)$$

where n is the number of words in the utterance and m is the number of actions that the system can perform.

¹Normally a term vector is a vector of length n where n is the length of the system's vocabulary, and the i th entry is a boolean value indicating whether the i th word in the vocabulary occurs in the utterance.

The above equation multiplies the likelihood of each word occurring in the utterance given the action together and then multiplies that by the prior probability of that action being required. Therefore, the actions more commonly requested will have a greater prior $P(A)$, so the calculation will favour the more common actions.

The denominator of the function simply normalises it, ensuring that a valid probability from 0.0 to 1.0 is produced. As mentioned above, our term vector includes only those words in the utterance. This can raise problems if a previously unknown word occurs in the utterance. How should we deal with this new word?

One approach is to ignore the probability of the word $P(w_i|A)$ where the i th word is the unknown word. However when considering the equation given above and also considering that the factors in the sum are all probabilities (and hence fall in the range 0 – 1), it becomes intuitively clear that ignoring the unknown word will give higher value for the product in the denominator for utterances that do not contain the word, as each additional factor in the product will make the result smaller.

We can also not use 0.0 as this will result in a probability of 0.0 for the action. We do not want to rule out the action, as it could easily be the correct action, based on an utterance with an extraneous word. Instead, we need to know the probability that the action has a previously unseen word associated with it. We use the following formula:

$$P(w_i|A) = \frac{\text{number of distinct words for action}}{\text{total number of words for action}} \quad (3.4)$$

3.2.2 Naïve Bayes using effective hashmaps

The Naïve Bayes algorithm relies on the system having stored a table containing the values for $p(w_i|A_j)$ and $p(w_i|NP_j)$ in a table. Rather than storing the probability in each cell of the table, we store the number of times during training that the word is associated with the given noun phrase or action. It is easier to store the count and then, during training, locate the appropriate cell and increment its value, rather than having to recalculate the probability of the word given the action or parameter each time.

The tables 3.1 and 3.2 on page 17 are produced after the system is trained with the following sentences:

- “Where is the dean’s office?” (Give directions to Laby 406a)
- “Where is the school office?” (Give directions to the school office)
- “Show me the school office.” (Move to the school office)
- “Who is Dale Carnegie?” (Play recording about Dale Carnegie)

The tables shown are small and are very sparse, even for a small set of actions, noun phrases and words. In much larger data sets, most of the counts will be zero. Rather than allocating a large and unnecessary multidimensional array to store the counts, we instead use a series of nested `HashMap` objects. The outermost `HashMap` maps the action or noun phrase to another `HashMap`. The internal `HashMap`’s keys are the words that have been associated with the action or parameter that the row is mapped to, and the values are integers counting the number of times the given word is associated with the given noun phrase or action. Missing words in the internal `HashMap` indicate that the word has not been associated with the action or noun phrases during training. Looking up an entry in a hash table is typically a constant time operation ($O(1)$).

Words	Actions		
	query	directions	moveto
the	0	2	1
deans	0	1	0
office	0	2	1
where	0	2	0
me	1	0	0
who	1	0	0
show	0	0	1
is	1	2	0
dale	1	0	0
school	0	1	1
carnegie	1	0	0

Table 3.1: Table of number of instances of the given word for the given action

Words	Noun phrases		
	Laby 406a	Dale Carnegie	office
the	1	0	2
deans	1	0	0
office	1	0	2
where	1	0	1
me	0	0	1
who	0	1	0
show	0	0	1
is	1	1	1
dale	0	1	0
school	0	0	2
carnegie	0	1	0

Table 3.2: Table of number of instances of the given word for the given noun phrase

3.3 Combining results of the two modules

Attempting to find a direct match in a trie is accurate if the trie contains the utterance (and its correctly associated action), and fast whether or not the trie contains the utterance. Statistical methods like Naïve Bayes are flexible enough to handle partial mismatches in the system and can make a good estimate of the utterance if the utterance is not already known by the system.

[2] combines a direct match algorithm and statistical method together to increase the resulting accuracy of the system. We have adopted a similar approach, using Bayesian analysis to attempt a parse when the direct lookup fails.

3.4 Implementation

Our implementation is written in Java 5.0. The genericity of the data structures in the Java SDK are used heavily throughout the system.

The standard Java libraries also tend to employ method names and types that are ver-

bose. It is also well-documented. The language itself is also widely-used. These characteristics make the Java language easy to port to other imperative languages.

The latest version of Carnegie Mellon University's open-source speech recognition software, Sphinx 4.0, is also written in Java. Future testing should involve data from a speech recognition system and having our system written in Java would facilitate integration of the Sphinx software.

Java programs can also be executed on multiple operating systems via the Java Virtual Machine. This is always an advantage, but is not of high importance in our case, as we can tailor the system to a particular operating system or hardware.

3.5 'Chatterbox' for web-based sample utterance collection

When training and testing the system it is important to find a large set of unique training sentences. However, if one person constructs all of the training sentences, they will likely reflect the idioms of that person, rather than a broader range sampling the population likely to interact with the system.

An early stage of the project involved the development of the Chatterbox tool. The tool is used to collect a wide variety of

The tool stored 10 goals that a visitor might have, and picked one at random every time a person visited the web page. It was necessary to state some goals so that the site's visitors had an understanding of the queries that the system was capable of dealing with. On the other hand, people that used the site pointed out that the way the questions were phrased lead the user – giving the impression that a particular response was needed. Despite this, very few of the collected samples were repetitive.

The Chatterbox tool used direct text input. Ideally, we would like to collect utterances as spoken English, rather than written, as people tend to use different language in speech than they would when writing or typing. Spoken utterances would have variations in accent that could be useful for testing the speech recognition software.

The Chatterbox program strips out punctuation and excess whitespace at the end of the response and between words, converts the response to lower case text, and then is stored in the database. These modifications allow the script to find repetitions regardless of capitalisation and whitespace. It does not perform any spelling check, so this must be performed manually.

3.5.1 Raw retrieved samples

258 responses were collected from the Chatterbox tool, 231 of them being unique.

3.5.2 Ambiguities, spelling errors, and insufficient information in collected samples

Some of the collected samples were unsuitable for training and had to be corrected.

The most common errors were spelling mistakes. As our system will use the output of a speech recognition program, the system is highly unlikely to encounter spelling errors. After correcting the spelling, a number of them were found to be duplicates of existing requests.

There were some ambiguous phrases. These were typically responses that expected a request to be accompanied to a location, but the response suggested that only directions were required. These requests were annotated as being requests for directions to the location instead.

Some responses were in fact several responses combined together. These responses were split into individual responses.

The corrected set contains 216 unique responses.

3.5.3 Training data as direct and parameterised utterances

Our training data is stored in an easily-readable format in a text file. Each line contains an utterance and the associated action delimited with a colon (see 3.5).

Treat the utterance as an indexable sequence of words.

Create queue to holds tuples containing possible searches.

The queue should be ordered such that tuples are ordered by increasing number of mismatches and decreasing positions in the utterance.

Set the global minimum number of mismatches to the maximum number of mismatches allowed.

Push a tuple of the form $\langle \text{start node}, 0, 0, \text{empty} \rangle$ onto the queue.

While the queue is not empty...

Pop the first tuple from the queue. This is the current tuple.

If the current tuple doesn't reference the first node, and the current tuple does not contain position 0 and the word at that position doesn't match the word at the node reference by the tuple...

Add one to the number of mismatches contained in the tuple.

If number of mismatches contained in the tuple is greater than global minimum...

Continue to the next iteration of the loop.

If the current tuple's position corresponds to the last word in the utterance, and the node referenced by the tuple has an action...

Create a response from the action on the node and the utterance match contained in the tuple.

Otherwise...

If the number of mismatches in the tuple is less than the global minimum...

For each child of the node referenced by the current tuple...

Add a tuple of the form $\langle \text{child node}, \text{mismatches in current tuple} + 1, \text{position in current tuple} + 2, \text{the utterance match in the current tuple with the word on the child node appended to the end} \rangle$

Add a tuple of the form $\langle \text{child node}, \text{mismatches in current tuple} + 1, \text{position in current tuple}, \text{the utterance match in the current tuple with the word on the child node appended to the end} \rangle$

For each child of that node...

Add a tuple of the form $\langle \text{child node}, \text{mismatches in current tuple} + 1, \text{position in current tuple} + 2, \text{the utterance match in the current tuple with the words on the first child and second child nodes appended} \rangle$

For each node reachable from the node referenced by the current tuple along an edge labelled with the word referenced by the current tuple...

Add a tuple of the form $\langle \text{child node}, \text{mismatches in current tuple} + 1, \text{position in current tuple} + 1, \text{the utterance match in the current tuple with the word on the first child appended} \rangle$

Return all the possible actions found by the algorithm.

Figure 3.3: The look up algorithm used to traverse the augmented transition network.

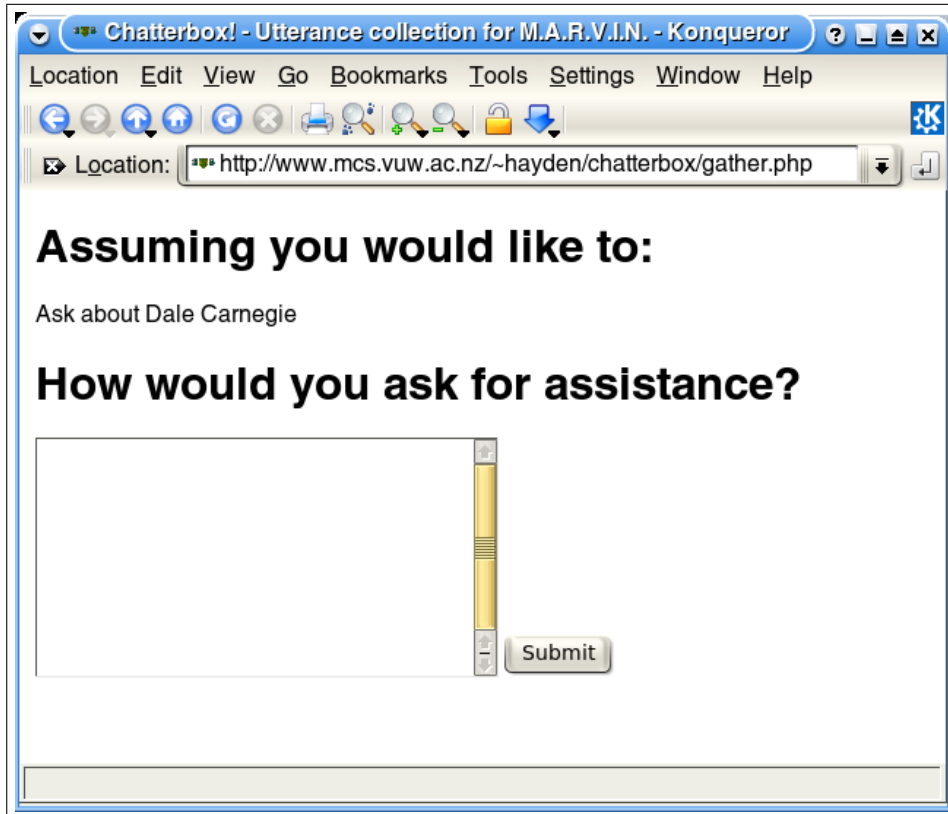


Figure 3.4: The *Chatterbox* web-based sample collection tool

```
where is the deans office:directions⟨laby_406a⟩  
where is the school office:direction⟨office⟩  
show me the school office:moveto⟨office⟩  
who is dale carnegie:query⟨dale_carnegie⟩
```

Figure 3.5: Training data file used to train the system shown in tables 3.1 and 3.2.

Chapter 4

Results

The training set collected with the Chatterbox tool is small compared to the number of possible ways to articulate the commands that the $M\mu$ system is capable of identifying.

It is therefore necessary to test the $M\mu$ system with a set of utterances that are not previously known by the system, as it can retrieve the appropriate command when given a known utterance.

The system can be tested using a leave-one-out test. This test involves removing a record from the test set, training the system with the remaining records, and then comparing system's output when parsing the omitted record with the expected result. This test is repeated for all the entries in the test set. Synonyms were not loaded into the trie for this training set.

4.1 Results with raw samples*

The *Combined* method uses the near-match first. If no results are found the Naïve Bayes module is used instead.

The Sympalog paper[4] mentions that the visitor might assume that the system has not heard them if a response is not given, so it is important that the modules execute quickly. The average time for each query was recorded. The individual times for each query were recorded from the time the system receives the utterance to the time it returns a response. This time does not include any processing by the speech recognition software.

Method	Accuracy	Partial	Average Time
Near-match	12.2%	4.1%	57ms
Naïve Bayes	69.5%	20.0%	1ms
Combined	68.6%	21.4%	58ms

Table 4.1: Results of leave-one-out test on raw test set (220 phrases)

4.1.1 Results with corrected samples*

Method	Accuracy	Partial	Average Time
Near-match	15.0%	2.7%	55ms
Naïve Bayes	78.6%	15.0%	1ms
Combined	76.8%	16.8%	62ms

Table 4.2: Results of leave-one-out test on corrected test set (220 phrases)

Chapter 5

Conclusion

Despite having a relatively small training set, early results at this point have shown promise. Our system is fairly accurate for the limited range of actions that the system will have to perform, given the utterances presented in the training set.

5.1 Effectiveness of these approaches in the real world

The two modules of the $M\mu$ system that we have implemented are at opposite extremes in terms of flexibility. The near-match is extremely rigid, but is expected to be exactly accurate for known phrases, however this assumes a 100% accurate parse by the speech recognition software. The Naïve Bayes module is highly flexible, as it all but ignores sentence structure, but this will lead to some situations in which the command returned by the system is wrong.

It is very difficult to predict the range of sentences that MARVIN will encounter in practice. The training data set, while an accurate sample of some of the sentences, is unlikely to match the prior distribution of the sentences. The users that created the test utterances were aware of their applications, and were intentionally trying to come up with a wider range of sentences. Some of the more direct sample utterances will have a higher prior probability than some of the more obscure requests. Capturing these common sentences and the prior distributions is important for both modules.

The near-match trie is there for necessary for the $M\mu$ system, as opposed to using the the Naïve Bayes module by itself. If the sentence exists in the near-match trie, then the system will find the correct command when presented with the same utterance. The Naïve Bayes system will be more likely to select the similar command with the highest prior probability.

5.2 Impact of speech recognition software on accuracy

When deployed, our system is highly dependent on the speech recognition software that provides input to our system. Inaccuracies introduced into the input by the speech recognition software will directly affect the system. The near-match system might still avoid these problems if only one word is incorrect. The Naïve Bayes system does not handle these inaccuracies and will process the incorrect word regardless. Adding homonyms for words or several words could help here.

5.2.1 Effect of the environment on speech recognition

Commercial speech recognition products are typically used for dictation and are therefore trained for a particular user. Our system will not be used in a single-user environment, nor

will the accents tend to be similar, as the population of the university is quite varied.

As well as being designed for a single user, the commercial products are typically used with a high-quality, noise-cancelling headset and microphone. This places the microphone closer to the speaker's mouth, so the volume of the speaker's voice relative to any background noise is greater.

MARVIN will typically be at a greater distance than this, so background noise will have more of an effect on the speech recognition software.

The speech recognition system has yet to be installed, so it has not been possible to evaluate $M\mu$ system with real data that it would receive from the speech recognition system. We expect that further modifications would be necessary to address errors introduced into the $M\mu$ system's input by the speech recognition system.

5.3 Future work

Understanding natural language requires accurate speech recognition, understanding of the syntax and semantics of the spoken language and a knowledge of the domain of the conversation.

5.3.1 Improving accuracy

The most immediate goal for our system for the future would be to increase the accuracy of the system for the utterances already stored in the system. Because the near-direct match does not have this problem, any immediate improvements must be made in the Naïve Bayes module.

Short of implementing complete Bayesian inference or a hidden Markov model, the Naïve Bayes module will still be relying on a word-by-word calculation. The meaning of sentence is a combination of several factors of which the words in the sentence are a major, but not the only, part. The context of the conversation, the structure of the sentence, the order of the words, and the relationship between those involved the conversation are all contributing factors.

Combining $P(A|\vec{w})$ and $P(NP|\vec{w})$ together in the Naïve Bayes algorithm

Some of the inaccuracies encountered during testing were partial matches with only one of the action or parameter correct, resulting in commands that have an inappropriate parameter for the given action.

The probabilities $P(A|\vec{w})$ and $P(NP|\vec{w})$ are interdependent, but our system does not combine them together. Currently, the system attempts to find both the most likely action and the most likely parameter, but not the most likely combination of the two.

We could use the Bayes formula to find a single command C consisting of an action and a parameter, however in a large dataset this will result in nm unique commands where n is the number of possible actions and m is the number of possible parameters.

A better solution would be to use a variation on the Naïve Bayes formula that computes $P(A, NP|\vec{w})$. *Describe formula here.* This should greatly reduce some of the inaccuracies we have encountered when using the Naïve Bayes module.

Different accents and pronunciation

As mentioned earlier (see section 1.3) it is likely that speech recognition trained to the voice and accent of a single user will have difficulty understanding the speech of different users

from the same background, let alone users that are from different countries and have completely different accents.

Some of these inaccuracies can be overcome by taking the context of the conversation into account. This will however require either a tighter coupling with the speech recognition software (so that the natural language understanding system can influence the probability that a word is recognised) or use some sort of phonetic encoding of each word to find possible utterances that sound similar.

We can however resolve misunderstandings by storing the utterances or words that were matched by the previous search and then using these words and utterances to increase the prior probability of these utterances when attempting to resolve the misunderstanding.

Examining the structure of the sentence

As mentioned previously in this chapter (section 5.3.1), part of a sentence's meaning comes from the order of the words in the sentence, and from its structure. While the system will likely encounter unstructured sentences, having the ability to infer a basic structure of the sentence would be useful for more complex commands, very long utterances, or utterances that describe a location through characteristics.

A more robust parse producing the structure of the sentence should help break the sentence into meaningful sub-phrases that can later be used to infer the intended command with greater accuracy, particularly for those utterances or commands that are potentially confusing. For example, the phrase: *"How do I get from the school office to Laby 502? I'm looking for Dale Carnegie."* contains two locations and mentions a faculty staff member. This will confuse the Naïve Bayes module as it potentially fits two commands with three possible parameters. A parse that produces the structure of the utterance and recognises the prepositions in the sentence will be able to determine both the origin and the intended destination for the directions. A more advanced system might even potentially indicate if Dale was in his office at that time.

5.3.2 Expanding Marvin's lexicon

Of the 216 unique utterances collected for testing, most were sentences that one would naturally expect to hear when asked for directions. This provides an excellent knowledge base to start with, but it is unlikely to cover the full range of sentences that real visitors might use.

Learning from encounters with visitors

Developing a system capable of learning new sentences through its encounters with visitors would be extremely advantageous for an autonomous robot like MARVIN. If the $M\mu$ system is given an unknown utterance and cannot identify the command, it could request clarification. If it can identify the command from the second utterance, then it can store the first utterance as having the same command as the second.

Including other languages

While the Naïve Bayes module's results are not as accurate as those produced by a grammar-based parser, it is capable of dealing with unstructured queries. It is also flexible enough to deal with foreign languages, as it does not rely on being able to infer the structure of a sentence for that language. A direct match will have the same advantage for a known query stored and retrieved in a foreign language. Most languages do not fully translate

from one language to another well, and idioms and colloquialisms are often lost. However, by storing the query in the native language, the near-direct match and Naïve Bayes match should perform very well.

A more structured parse of utterances in foreign languages would require a greater understanding of the language's sentence structure.

Bibliography

- [1] FELLBAUM, C. *WordNet: A Lexical Database*. MIT Press, 1998.
- [2] FENG, D. Cooperative model based language understanding in dialogue. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology* (Morristown, NJ, USA, 2003), Association for Computational Linguistics, pp. 1–6.
- [3] MACHEREY, K., OCH, F. J., AND NEY, H. Natural language understanding using statistical machine translation. In *EUROSPEECH 2001: 7th European Conference on Speech Communication and Technology*, pp. 2205–2208.
- [4] NÖTH, E., HORNDASCH, A., GALLWITZ, F., AND HAAS, J. Experiences with commercial telephone-based dialogue systems. *it - Information Technology* 46, 6 (2004).
- [5] OCH, F. J., TILLMANN, C., AND NEY, H. Improved alignment models for statistical machine translation, 1999.
- [6] PIETRA, S. D., EPSTEIN, M., ROUKOS, S., AND WARD, T. Fertility models for statistical natural language understanding. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics* (Morristown, NJ, USA, 1997), Association for Computational Linguistics, pp. 168–173.
- [7] REEVES, B., AND NASS, C. CLSI Publications and Cambridge University Press, 1996.