EXAMENSARBETE
VID CSC, KTH

Datadrivet lärande av vägbeskrivningar

Data-driven learning of the meaning of route descriptions

Lioubartsev, Dmitrij **dmitrijl@kth.se**

Hallqvist, Kristoffer **khallq@kth.se**

Exjobb i: datalogi

Handledare: Skantze, Gabriel

Examinator: Björkman, Mårten

Datadrivet lärande av vägbeskrivningar

Sammanfattning

Interaktionen mellan människor och datorer begränsas ofta av våra vitt skilda sätt att kommunicera på, exempelvis vid en beskrivning av en väg. I detta projekt försöker vi utveckla ett program som genom att analysera visuella och verbala vägbeskrivningar gjorda av mäniskor, kan gissa sig till ords betydelser genom att koppla ihop dem med fördefinierade objekt eller handlingar. Resultaten visar att det med en tillräcklig mängd data går att lära en dator ord som representerar specifika objekt eller rörelsemönster genom att hitta ord som sägs i samband rörelser i vissa riktningar eller i närheten av vissa objekt.

Data-driven learning of the meaning of route descriptions

Abstract

The interaction between humans and computers is often limited by the large differences of the ways we prefer to communicate, for example when trying to describe a route. In this project, we aim to develop a program that by analyzing visual and verbal human given route descriptions, can accurately guess the meaning of certain words by associating them with predefined objects or actions. The results indicate that with a sufficient amount of data, it is possible to learn the words representing certain objects or movement patterns by finding words said in conjunction with moving in certain directions or in the vicinity of certain objects.

Table of contents

1 Introduction

A large problem in the computer science field is having a computer understand natural human language, both spoken and written. Different humans speak in different accents, speeds, pitches, and have a tendency to not strictly follow grammatical rules. Different words can have the same or similar meanings, and aside from that, a human can easily make unintentional logical errors by saying or writing the wrong word.

Overcoming these problems is a complicated task, but it opens up possibilities for many interesting applications, like having a computer following instructions. Specifically, this project is focused on navigational instructions. A standard approach is to tell the computer the definitions of different words. It is a simple task to program a robot (or any machine) to turn left when it receives the instruction “left”. The computer will then parse the human instructions and will try to identify these words to try to understand the instruction.

In this project, we explore if it is possible for the computer to learn the definitions of words by itself, from a sample of test data. The setup is the following: the computer knows a set of concepts, which are the definitions of some words. However, it does not know what language the instructions are given in, and has to learn the words in that language which correspond to the concepts. The concepts will be both directions (“north”, “west”...) and objects (“tunnel”, “train station”). We will use a technique called tf-idf (see chapter 2.2) and see how well it is suited for this task.

The premise for this project is the project *A Testbed for Examining the Timing of Feedback using a Map Task* by Gabriel Skantze [4]. A test person moves the mouse cursor on a specific map in a specific route, while describing the route (see figure 1.1). This was done by ten different test persons on five different maps. Both the speech and the mouse movements were recorded, and that is the test data used in this project.

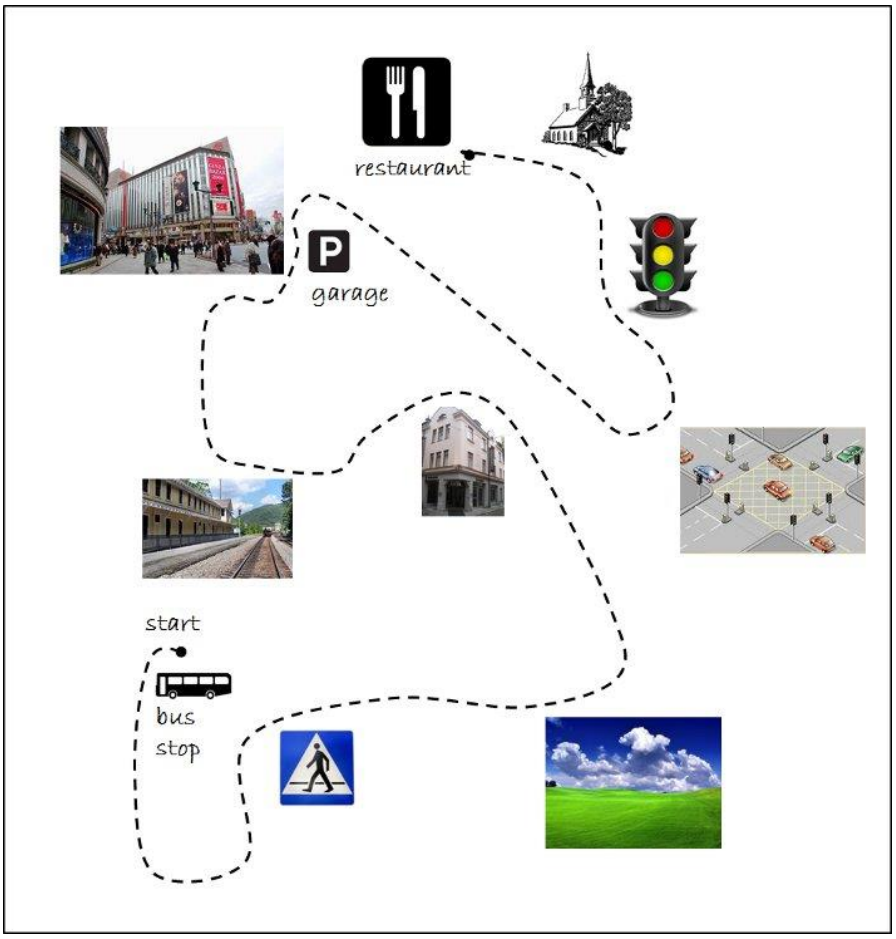

Figure 1.1. One of the maps with the route the test person it supposed to take. The test person moved the mouse along the black trail from the start at the bus stop to the end at the restaurant, while describing the route.

2 Background

This chapter explains the background to this project, including some related work. It mentions the work which made this project possible, as well as a very detailed explanation of the test data available. It also covers the tf-idf method which is used by the program. Many important terms are defined in section 2.2.

2.1 Similar work

This project is within the field of natural language processing (NLP), which is quite broad. There are many articles and papers in the field of NLP, far too many to cover in this section. To narrow it down, we have found some research being done in the specific area of route descriptions:

The paper *Learning to Follow Navigational Route Instructions* by Nobuyuki Shimizu & Andrew Haas [1] is about guiding a robot to the correct destination given a route description in an unconstrained natural language. However, they assume that the natural language is English, and they also focus on actually following the instructions.

An article which is more similar but also more extensive is *Learning to Follow Navigational Directions* by Adam Vogel & Dan Jurafsky [2]. They work with very similar test data, but their methodology is too complex to take much from in this small project.

Learning to Interpret Natural Language Navigation Instructions from Observations by David L. Chen and Raymond J. Mooney [3] is a project with a very similar premise, but is broader. They also have a simulated robot following the instructions given. However, the maps they work with are much simpler, and are comparable to graphs, where one can only move from nodes to other nodes that have an edge between them. In our maps, the user is free to move in any direction at any time.

2.2 Test data available and definitions of terms

The foundation for this project is a certain amount of data collected from a previous experiment - *A Testbed for Examining the Timing of Feedback using a Map Task* - by Gabriel Skantze [4]. In the experiment, test persons are describing a route on a map (see figure 2.1) to the computer, while manually tracing the path themselves with the mouse cursor. This was done by ten different test persons on 5 different maps. Each map has a route drawn on it, passing various pictures of landmarks. The idea is that the test person describes the route using directional words as well as references to the landmarks on the map, e.g. “after you pass the traffic light, go south”.

The data as a whole entity will from now on be referred to as the *test data*. The test data consists of 48 *sessions*. Each session includes one test person describing the route on one map. Each session has three key elements: the *map*, the *mouse data file* and the *dialogue file*. The map is the one the test person traced the path on, the mouse data file is a log of the test person’s mouse movement and the dialogue file is a transcription of the user’s spoken route description.

The mouse data file consists of a long list of *data points*. All data points together define the *walk*, the route the test person actually traced with the mouse. One data point contains the x and

y coordinates of the mouse position, and the timestamp for when the cursor was at that position. The timestamp is a relative measure, with 0 referring to the time when the recording began. Each data point also has an *index*, starting at 0, giving each data point a unique ID. Thus, a *section* can be defined by a *start index* and an *end index*, constituting a series of consecutive data points. So a section represents an array of points, and can be considered to be such, but the internal representation is two indices.

The recording of the test persons' route descriptions have been transcribed into text manually, so the dialogue files contain the route description in text, with near-perfect transcription. The full route description will be referred to as the *dialogue*. The dialogue is divided into *segments*. A segment corresponds to a sentence and is on average around 10 words long. Each segment also has a start time and end time, and is given in relative times, meaning that 0 is the timestamp of when the recording started. Because of this, a timestamp of for instance "10.000" somewhere in the dialogue file refers to exactly the same time as a timestamp of "10.000" somewhere in the mouse data file.

The sound files containing the actual recordings are not available, but they are not relevant to this project either.

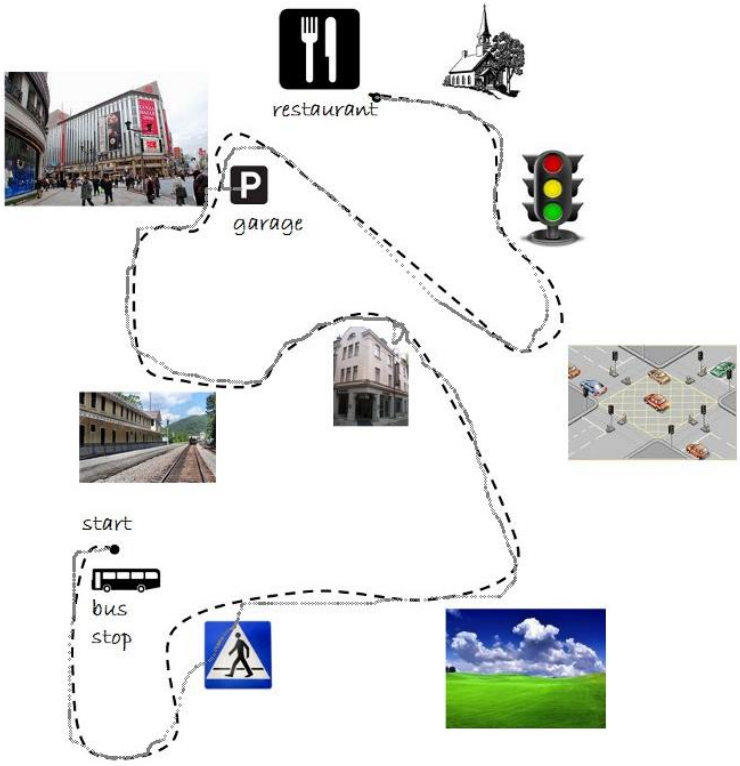

Figure 2.1: A walk (in grey) plotted out on a map. The black dashed line is the route the test person is supposed to describe.

2.3 Tools

2.3.1 Tf-idf

Term Frequency - Inverse Document Frequency, or tf-idf in short, is a numerical statistic often used in the field of information retrieval. Tf-idf indicates how characteristic or important a word is to a certain *document* in a document collection. A document in this context has no requirements about being well-structured; it can essentially be just any piece of text.

Tf-idf consists of two parts: *tf* and *idf*. The term frequency, $tf_{d,w}$, tells us how common a certain word *w* is within a certain document *d*. The inverse document frequency, idf_w, tells us how rare that word *w* is in the whole document collection. Note that the *idf* values depend on the words and the whole dialogue input data, with each word having a fixed idf-score regardless of the document being worked on. The *tf* value, in contrast, is a function of not only a word, but also a specific document.

The *tf* value is usually defined as the raw amount of occurrences of the word within the document. The *idf* value for a word *w* is calculated as follows:

$$idf_w = log(N/df_w) \qquad \textbf{(Formula 2.1)}$$

where N is the total amount of documents in the whole text collection, and dfw, the *document frequency*, is the number of documents in the whole collection that contain the word *w*. The final tf-idf score for a word *w* in the document *d* is calculated with the formula

$$tf\text{-}idf_{d,w} = tf_{d,w} * idf_d \qquad \textbf{(Formula 2.2)}$$

There are alternative definitions of the tf-idf score. For example, it is common to attenuate the *tf* value using a logarithm, a root function or even a boolean function (if the word occurs at least once in the text, tf is 1, otherwise 0). The base of any logarithm functions used, including the logarithm function in formula 2.1, is not relevant to the final result, as long as the same base is used consistently for all words. The *tf* and *idf* components can be weighted differently depending on what is considered most important to the context.

A word that is highly characteristic to a certain document is a word which is common within that document but rare in the rest of the text collection. Such a word is labeled with a high tf-idf score in that document.

3 Procedure

In this section the problem is defined more precisely, and we explain the procedure of how to get from "Test data available" to the solution to the problem.

3.1 Limiting and specifying the problem

We start by defining a limited set of *concepts*. A concept is not a word, but rather the semantic implication of it. For example, "running" is a word, but its meaning - the action of running - is what we refer to as a concept. The concept is of course not affected by the language used to describe it. The goal is to have a computer program learn the words for different concepts from a natural language, based on the test data available (chapter 2.2). No assumptions are made about the natural language from the program's point of view.

For a small-scale project like this, the type and amount of concepts that the program will try to learn the words for has to be limited. Some key concepts in route descriptions are *landmarks* and *directions*, which is what will be the focus in this project.

A simple route description from the landmark "Train Station" to the landmark "Church" in natural language may look as follows:

"Go north until you reach the crossroad. You should turn west and move in that direction until you see a hotel. After the hotel, go north and you should soon be at the church."

With access only to words representing directions and landmarks, one could still construct a sentence like:

"*Train Station north Crossroad west Hotel north Church*"

It may not be as detailed, but it might be possible to follow that route description and still reach the destination.

We define concepts for the four cardinal directions: north, east, south and west, later referred to as *movement patterns*, or *MPs* for short. We also define concepts for 12 different landmarks featured on the maps in the test data (chapter 2.2): bus stop, crossing, hotel, station, department store, garage, junction, traffic lights, church, field, restaurant, tunnel.

To be able to interpret the results, we decide ourselves what words we consider to be the "*correct*"[1] representation of each concept. We use English words since the test data is in English.

[1] "Correct" is written with quotation marks because in a natural language, there are often many correct ways to describe specific concepts. This is discussed further in chapter 5.1.

Concept	"Correct" word(s)
(moving) North	"north"
(moving) South	"south"
(moving) East	"east"
(moving) West	"west"
Bus Stop	"bus stop"
Crossing	"crossing"
Hotel	"hotel"
Station	"station"
Department Store	"department store"
Garage	"garage"
Junction	"junction"
Traffic Lights	"traffic lights"
Church	"church"
Field	"field"
Restaurant	"restaurant"
Tunnel	"tunnel"

Table 3.1. The "*correct*" words for each concept.

Thus, a good result can be defined as many concepts that the program maps to their respective correct words. This is an indicator of how well the program could learn the words for the defined concepts. The final definition of the problem can be formulated as follows:

Given the test data available, how well can the program learn the correct words for each of the 16 predefined concepts using the tf-idf method?

3.2 Overview of method

This task is performed by constructing a Java program that takes the test data as input. The plan is to analyze the test data to determine the time for when the test person moved the cursor close to certain landmarks or in certain patterns, thus for each concept creating a mapping to a set of time intervals (chapters 3.3 and 3.4). Each concept is then associated with a collection of segments which all took place during these time intervals (chapter 3.5). The key idea is that when moving according to an MP or near a landmark, the test person is likely to have described the concept using an appropriate word. For each concept, tf-idf scores are computed for all words in the associated collection of segments (chapter 3.6). By sorting these words in descending order with respect to their tf-idf score, we will have a ranking order for which words are considered the most likely to represent the given concept.

3.3 Recognizing movement patterns

We have developed an algorithm used to identify movement patterns in the mouse data.

The algorithm looks for specific MPs one at a time, starting by only looking for movements going south. It begins by selecting 50 mouse data points at the beginning of the walk. It then checks whether the movement within that section can be qualified as a movement in a southerly direction. It is qualified if the ending point is placed south of the starting point and the horizontal distance between the two points is no longer than a certain fraction of the vertical distance. This fraction is set to 0.1.

If the movement does not qualify, the interval is shifted forward by one data point. A check is performed to see if the new section qualifies, and the process is repeated until the movement is qualified or the end of the mouse data file is reached.

To make sure that this was not just an error on the test person's part (for example, the test person flinched with the mouse), the algorithm continues to further extend the section by one point at a time until a certain number of extra points are added. This number is here set to 50. If during this time, the section still does not qualify, the instance is finalized and added to the result list. The algorithm then keeps looking for more of the same MP in the rest of the walk, starting with a new section beginning where the last MP ended.

The process is then repeated for the other MPs, using the same procedure. The qualification requirements for the other MPs are similar, but of course adapted to a different direction. For example, a movement to the east needs the horizontal distance to be at least ten times as large as the vertical distance, while of course also requiring the ending point to further out east than the starting point.

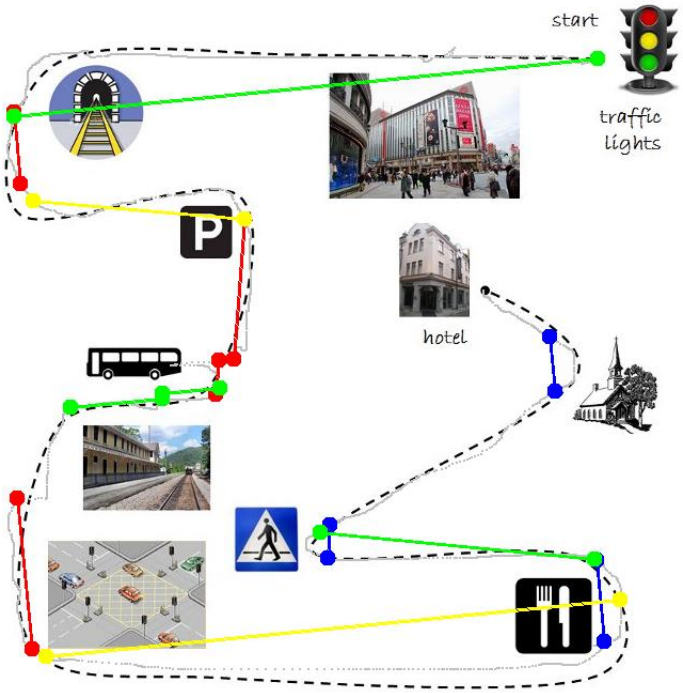

Figure 3.2. The results of the MP recognition algorithm drawn on a map, represented with colored lines. Each color represents a cardinal direction: Green = west, red = south, yellow = east and blue = north. The grey line of dots represents the walk made by the test person

3.4 Recognizing landmarks

The algorithm used to detect landmarks is very simple. It iterates over the list of mouse data points. For every data point, the algorithm calculates which landmarks are located within a certain number of pixels, in this case set to 70, of the data point coordinates. The distance function is defined as the euclidian distance:

$$Distance(Point\ p1,\ Point\ p2) = \sqrt{(p1.x - p2.x)^2 + (p1.y - p2.y)^2} \qquad \textbf{(Formula 3.1)}$$

A consecutive series of data points that are within distance of a specific landmark are considered to constitute a time interval. The algorithm also keeps track of the previous landmark that the walk passed, which will be of use later on.

3.5 Mapping intervals to dialogue segments

We now have a set of time intervals, from different sessions, mapped to each concept, and the next step is to translate these time intervals into collections of dialogue segments. However, a theory is that the test person will also say the relevant words slightly earlier or later than what is expected. It may thus be a good idea to extend the time intervals in order to cover more dialogue segments.

In the case of landmarks, the test person is likely to have said the correct word for a landmark a long time before actually arriving to that landmark. For example "*are you by the hotel? Go north until you get to the field"*, might have been said around the hotel, but would also be relevant to the field. Thus, for landmarks, we extend each interval backward to the endpoint of the closest time interval, information that is provided by the landmark recognition algorithm.

Regarding the movement patterns, a phrase like *"... until you get to the garage and then go west."* might be said before actually starting the cursor movement to the west. Therefore, these intervals are extended both backward and forward by two different constant number of data points. We are here using the values 60 and 30 respectively, of course making sure that the endpoints do not go out of bounds.

Each time interval will be mapped to a collection of segments. For a segment to be included in this collection, at least a certain fraction of the segment has to appear within that interval. In the algorithm the fraction 0.2 is used.

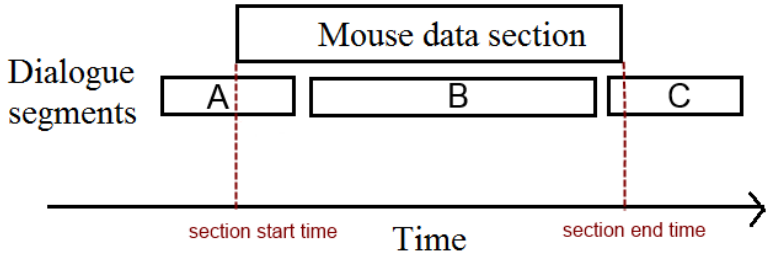

Figure 3.3. The time interval is mapped to a collection of dialogue segments, including segment A and B, but not C, because the overlapping time interval is too small.

3.6 Computing tf-idf

We now have a set of time intervals, from different sessions, mapped to each concept, and the next step is to translate these time intervals into collections of dialogue segments. However, a theory is that the test person will also say the relevant words slightly earlier or later than what is expected. It may thus be a good idea to extend the time intervals in order to cover more dialogue segments.

Given a word, we define the *document frequency*, *df*, as the total number of segments in which the word occurs at least once, meaning that a word appearing more than once inside a single segment is not weighted higher within that segment than a word occurring just once in it.

To compute the *inverted document frequency*, *idf*, for each word, formula 2.1 is used. N is the total number of dialogue segments, and the logarithm is of base 10.

Tf is defined as the amount of segments in a given segment collection that contain a certain word. This means that multiple occurrences within a dialogue segment do not increase the weight further. This is different from the traditional *tf* definition, but similar to the *df* one. Specifically, the collection of segments will be the segments mapped from time intervals as described in chapter 3.6.

The tf-idf score is then calculated for each word using the slightly modified formula

$$tf\text{-}idf = tf * idf^2 \qquad \textbf{(Formula 3.2)}$$

Putting a higher weight on the idf value by squaring it further benefits less common words at the expense of more common ones.

4 Results

In this chapter the results are presented. For each concept, the words are ranked by their tf-idf score, i.e. by their likelihood of being the correct representant of the given concept. The magnitude of the scores is not relevant - the only things that matter are the relative scores, and thus also the ranking order.

4.1 Final results

This is the results of the exact program described in chapter 3. Out of all different variations of the algorithms we tried, these are the results we consider to be the “best”.

Table 4.1 MP south

Word	Tf-idf score
south	98.00
traffic	92.69
lights	87.63
south	73.38
again	65.42

Table 4.2 MP north

Word	Tf-idf score
north	108.86
traffic	65.50
lights	65.13
south	64.29
again	61.40

Table 4.3 MP west

Word	Tf-idf score
west	86.27
station	62.13
field	61.40
train	55.67
at	55.47

Table 4.4 MP east

Word	Tf-idf score
east	92.62
crossing	64.36
after	58.84
continue	58.22
turn	56.44

Table 4.5 Landmark bus stop

Word	Tf-idf score
bus	79.92
stop	77.82
down	22.77
south	21.35
again	18.39

Table 4.6 Landmark crossing

Word	Tf-idf score
crossing	74.02
sign	41.99
pedestrian	32.80
east	22.12
should	19.44

Table 4.7 Landmark hotel

Word	Tf-idf score
hotel	70.31
white	22.45
thats	21.33
building	20.97
pass	17.76

Table 4.8 Landmark station

Word	Tf-idf score
station	63.96
train	43.02
railway	25.66
-tation#	19.76
s-	18.28

Table 4.9 Landmark department store

Word	Tf-idf score
store	49.41
department	48.01
garage	24.85
-tore	18.24
t_BREATH_IN	17.87

Table 4.10 Landmark garage

Word	Tf-idf score
garage	39.76
parking	30.38
place	24.32
south-	18.24
east	16.59

Table 4.11 Landmark junction

Word	Tf-idf score
junction	77.81
intersection	31.46
crossroads	20.14
stop	16.79
bus	16.28

Table 4.12 Landmark traffic lights

Word	Tf-idf score
traffic	86.71
light	82.87
lights	69.59
red	42.76
south	24.26

Table 4.13 Landmark church

Word	Tf-idf score
church	62.89
hotel	18.50
your	16.25
yourself	15.31
on	15.27

Table 4.14 Landmark field

Word	Tf-idf score
field	74.10
green	60.94
fields	41.06
meadow	34.58
north	21.22

Table 4.15 Landmark restaurant

Word	Tf-idf score
restaurant	93.43
northwest	35.79
parking	27.01
north	26.76
are	23.76

Table 4.16 Landmark tunnel

Word	Tf-idf score
tunnel	77.92
railroad	22.45
all	16.83
right	16.73
way	15.73

4.2 Comparing idf weighting schemes

These are the results of using formula 3.2 compared to formula 2.2 when computing the tf-idf scores. The difference is the weighting of the *idf* used in the computation of the tf-idf score. The tables show the MPs North and West with the two weighting variants. Note that the variant with formula 3.2 is what is used to get the results in chapter 4.1, so those tables are identical.

Table 4.17 MP north, using idf^2

Word	Tf-idf score
north	108.86
traffic	65.50
lights	65.13
south	64.29
again	61.40

Table 4.18 MP north, using idf

Word	Tf-idf score
north	113.35
and	95.19
then	92.63
go	88.47
to	84.70

Table 4.19 MP west, using idf^2

Word	Tf-idf score
west	86.27
station	62.13
field	61.40
train	55.67
at	55.47

Table 4.20 MP north, using idf

Word	Tf-idf score
and	74.73
the	74.63
west	73.72
you	71.92
go	71.67

As can be seen, increasing the *idf* weighting by squaring vastly improves the results. However, further increasing the *idf* weighting does not further improve the results, but rather makes uncommon words stand out too much.

4.3 Comparing section vs extended section

These are the results of extending the sections, as described in chapter 3.5, versus not doing that. The tables show landmarks Hotel and Restaurant. Note that the results in chapter 4.1 were achieved with extension of intervals extending, so the tables are identical.

Table 4.21 landmark hotel, extending

Word	Tf-idf score
hotel	70.31
white	22.45
thats	21.33
building	20.97
pass	17.76

Table 4.22 landmark hotel, without extending

Word	Tf-idf score
hotel	55.51
white	22.45
building	20.97
pass	16.28
left	15.45

Table 4.23 landmark hotel, extending

Word	Tf-idf score
restaurant	93.43
northwest	35.79
parking	27.01
north	26.76
are	23.76

Table 4.24 landmark hotel, without extending

Word	Tf-idf score
restaurant	72.53
parking	23.63
lot	23.43
left	17.82
north	16.61

As can be seen, the correct words for the landmarks, “hotel” and “restaurant” respectively, have higher scores when the intervals are extended. That is natural - since more words are said during a longer interval, that word will also appear more times, getting a higher *tf* value and thus also a higher score. However, other irrelevant words also appear more times, which can lead to the program being less confident about the what the correct word is.

5 Discussion

In this chapter the results are presented. For each concept, the words are ranked by their tf-idf score, i.e. by their likelihood of being the correct representant of the given concept. The magnitude of the scores is not relevant - the only things that matter are the relative scores, and thus also the ranking order.

5.1 Result analysis

Looking at the results, we can tell that the program successfully mapped each concept to the word we intended, except for the three landmarks whose english language representation consisted of more than a single word. Our program does not support considerations of these representations, but instead treats each word as an independent entity. We can still note, though, that the top candidates for these concepts were in fact parts of the multi-word representations we had in mind, which means that the program should be able to get the words right with only a minor modification.

We can also see that when it comes to landmarks, the program generally had an easier time choosing a winner compared to the situation with movement patterns. For example, the word "south" was only deemed slightly more likely than "traffic" to represent the action of moving south, whereas the word "tunnel" had a score nearly four times as high as "railroad", the second most likely candidate, when trying to find the english word for the tunnel landmark.

The results in chapter 4.2 showed that the standard weighting on the idf does give high scores to the correct words, but also gives very high scores to common words like "and" and "go". By modifying the weight on the idf score by squaring it, the common words' scores becomes significantly lower, and it really improved the results. Some experimenting also showed that taking the weighting further, to a cubic weight, made some uncommon words receive too much score, so using formula 3.2 to calculate the tf-idf score is the optimal.

The results in chapter 4.3 showed that extending the sections did not have a very large impact on the results. While the correct words got a higher absolute score, so did the other words, and the benefit extending the sections is questionable.

5.2 Improving results by improving test data

It is important to remember that there is usually not a single true unambiguous representation for a given concept, and the test persons can not always know whether their own description is sufficiently accurate. For example, the "field" landmark was mapped to the word "field" as its most likely representation. However, when you look at the candidates below it, you find words such as "green", "fields" and "meadow". Many people consider "meadow" and "field" to be synonyms, and the field is indeed green, which means that a lot of people likely referred to it as the "green field". Are these representations wrong? No, most certainly not. They are in fact more descriptive than "field", and would actually probably be prefered in a route description.

Another interesting example is the "hotel" landmark. By just looking at the picture, one can tell that it is a white building but it is quite hard to see that it is in fact a hotel. The words "white" and "building" are among the top candidates, which indicates that many test persons probably

referred to it simply as the “white building”. And that is most definitely not an incorrect description, because the hotel is indeed a white building! Yet another example is “department store”, where one test person kept referring to it as “some sort of commercial building”.

You can also apply a similar reasoning to the MPs. They don’t have to be described in terms of cardinal directions - the words “up”, “down”, “right” and “left” might be preferred depending on the context, but neither of them can be considered wrong.

One solution to problem with the ambiguity of the landmarks on the maps is to add captions under all landmarks. Some maps had captions under certain landmarks, and the consistency in the dialogue increased for those landmarks with captions, so it would most certainly improve the results. However, is that something desirable? In the real world, people do not always describe an object consistently using the same words. A hypothetical commercial system needs to be able to understand synonyms. One way to accomplish this is to do the experiment with many more test persons, so that all synonyms are used often enough to get significantly higher scores than any “incorrect” words. In that case it might not be easy to handle multi-word representations like “bus stop”, without further modifications to the program.

5.3 Accuracy of tf-idf

The way the test persons described the landmarks was not always consistent. As concluded in the previous chapter, it can be improved by using modified versions of maps and giving clearer instructions to the test persons. However, that is not the only inconsistency. Figure 5.1 highlights how inconsistent a walk can be.

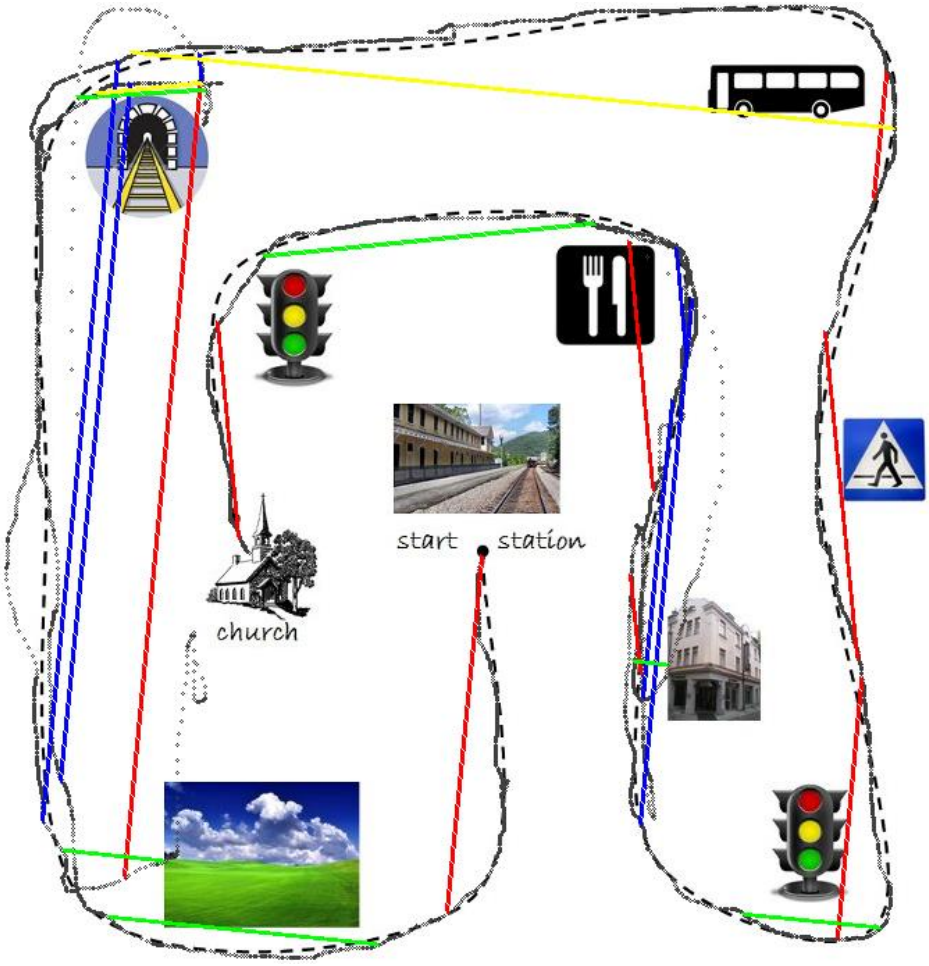

Figure 5.1. The walk of one of the test persons on map number 4, with recognized instances of MPs. The lines seem very random and highlight how confused the program is.

As can be seen in figure 5.1, the program has recognized a lot of MPs where there should not be any, and showcases that humans may have trouble following a given route with the mouse. This is quite an extreme case though, and most walks are quite careful. The exact impact on the final results of walks like this is unknown, but it is most definitely not positive.

The tf-idf method has shown to be quite robust, though. With all the errors in the test data, it still ranked the "correct" word(s) highest, even though it was close in a few cases (MP "south" as the closest one). Tweaking various parameters did not affect the results significantly either. And of course, the more test data, the higher accuracy in the results.

6 Conclusion

Tf-idf is a viable and accurate method to identify the correct words for predefined concepts. Despite inconsistent test data, it still managed to find the correct word(s) for all 16 concepts, and slightly tweaking some of the parameters did not have a large impact on the results. However, it is not that potent for cases where there are multiple ways to describe one concept. In a real-world environment, the tf-idf technique is probably not optimal. In a more controlled and simulated environment, like the situation in this project, tf-idf can be very effective.

7 References

1. *Learning to Follow Navigational Route Instructions* - N Shimizu, A Haas (2009) http://ijcai.org/papers09/Papers/IJCAI09-249.pdf
2. *Learning to Follow Navigational Directions* - A Vogel, D Jurafsky (2010) http://nlp.stanford.edu/pubs/spatial-acl2010.pdf
3. *Learning to Interpret Natural Language Navigation Instructions from Observations* - David L. Chen and Raymond J. Mooney (2011): http://www.cs.utexas.edu/~ml/papers/chen.aaai11.pdf
4. *A Testbed for Examining the Timing of Feedback using a Map Task* - Gabriel Skantze (2012) http://www.speech.kth.se/prod/publications/files/3761.pdf
5. C. D. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008