Navigability in information networks

Master Thesis
Mircea Grecu
July 26, 2014

Advisors: Prof. Dr. A. Krause, Dr. A. Karbasi
Learning and Adaptive Systems Group
Department of Computer Science, ETH Zürich
Abstract

In this work, we study the problem of navigation in information networks. We propose a model that can describe how humans traverse such a network from one item to another, which is based on embedding the network into a metric space. We construct a distance function that determines this embedding, using examples of navigation traces made by people and other information about the network, and we define a simple navigation algorithm that uses this distance function. This algorithm generates paths between nodes in the network that are similar to the paths traversed by people between the same nodes.

We also present two methods for recommending new links in such a network, which would make human navigation easier.

The first method is based on creating a score function for ordered pairs of nodes, learned from link suggestions given by people. This function determines which of the ordered pairs should be recommended as links first. This method gives recommendations that are similar to unseen link suggestions given by people that traversed the network.

The second method aims to minimize the average length of the paths traversed by our navigation algorithm between any two nodes in the network. It is based on maximizing a monotone submodular function that resembles this objective.

We test these ideas on a dataset that consists of a subset of Wikipedia articles and navigation traces made by people that were trying to get from one article to another, only by following the links in the pages.
Acknowledgements

I would like to thank Dr. Amin Karbasi and Prof. Andreas Krause for their guidance and valuable ideas for this work.

I am grateful to my family and friends, who supported me and encouraged me to do a good job, even though sometimes it wasn’t clear to me what value this thesis brings to the world.
# Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Contributions</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Outline</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Background and related work</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Small-world networks</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Modelling human navigation</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>Link prediction</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Problem statement</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Theoretical setting</td>
<td>11</td>
</tr>
<tr>
<td>3.2</td>
<td>Objectives</td>
<td>12</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Modelling human navigation</td>
<td>12</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Recommending new links</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>Metric embeddings for information networks</td>
<td>15</td>
</tr>
<tr>
<td>4.1</td>
<td>Embeddings using network structure and information content</td>
<td>15</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Graph distance</td>
<td>16</td>
</tr>
<tr>
<td>4.1.2</td>
<td>TF-IDF angle distance</td>
<td>16</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Jaccard distance between sets of inlinks</td>
<td>17</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Distance in the category tree</td>
<td>17</td>
</tr>
<tr>
<td>4.2</td>
<td>Embedding based on counting ordered pairs in navigation traces</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>Learning a linear combination of distance functions</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>Link recommendation</td>
<td>21</td>
</tr>
<tr>
<td>5.1</td>
<td>Matching user suggestions</td>
<td>21</td>
</tr>
<tr>
<td>5.2</td>
<td>Reducing the average path length</td>
<td>22</td>
</tr>
</tbody>
</table>
## Contents

6 Experiments ........................................... 27
   6.1 Dataset ........................................ 27
   6.2 Results ........................................ 27
       6.2.1 Embeddings .............................. 27
       6.2.2 Link recommendation ................... 30

7 Conclusion ............................................ 35
   7.1 Summary ....................................... 35
   7.2 Future work ................................... 36

Bibliography ........................................... 37
1.1 Motivation

Most of the bits of information in the contemporary world are part of more complex structures, made of interconnected items. These structures are called information networks and, as any network, can be modeled using graphs. The web pages, news articles, research papers, dictionaries can be considered nodes and the links or references that connect the items are the edges.

A big problem concerning these networks is being able to find the information you need as fast as possible. This is the subject of the field of information retrieval. But most of the efforts around this problem are related to searching information items using queries. In contrast, we want to look at how people find information by navigating a network from one item to another, using the links between items, until the required item is reached. Understanding human navigation in these types of networks and discovering ways to make it more efficient can give insights for designing better systems for accessing information.

The World Wide Web can be seen as the biggest information network, composed of web pages and hyperlinks between them. And while we think we can find most things we are looking for using online search engines, the reality is that most of the web pages are not accessible by bots that crawl the web, and therefore, are not indexed by the major search engines. It is estimated in [7] that the quantity of information accessible using query forms of searchable databases - the so called ‘deep Web’ - is 500 times larger than the size of the information contained in the ‘surface Web’, which consists of static HTML pages. And even when we get relevant results using a search engine, we often click on a result and then use hyperlinks to navigate to the final information item we are looking for.
1. Introduction

Besides the obvious applications in information retrieval, modelling human navigation in networks can be applied to other types of navigations as well: finding a person in a social network, reaching a destination in a road network, finding a target computer in a computer network.

1.2 Contributions

This study had two main goals:

1. to find a formal model, based on a simple algorithm, that describes how people navigate information networks;

2. to explore methods that can improve the structure of a network, so that it becomes as easy to navigate as possible.

The contribution of this work for the first goal is a method to construct a distance function between items in a network, based on examples of navigation traces made by people. The purpose of this function is to show how far items are from one another in the semantic space in which humans think. It can be used to predict the steps made by people to reach a particular target. At each step in the navigation, people try to get as close as possible to their desired target in this semantic space. Therefore, the distance function can be used to define a simple algorithm that simulates navigating the network by people.

For the second goal, we look at methods for recommending new links in an existing network, which would facilitate navigation. We consider two different objectives for a link recommender:

1. matching the link suggestions given by people while navigating the network;

2. maximizing the reduction in average path length between all pairs of items in the network, when traversed by the navigation algorithm mentioned above.

For the first objective, a method is proposed that associates a score to each ordered pair of nodes for which there is no link. The score function is learned from a sample of link suggestions made by people while traversing the network. The pairs are sorted in descending order of the scores, and recommended in that order.

For the second objective, we propose a recommender that efficiently finds a near-optimal solution for a maximization problem that closely resembles this objective.

The methods are tested on a network that represents a subset of Wikipedia articles and the hyperlinks between them, along with navigation traces from
people trying to get from a starting article to a final one, only by following the links in the documents.

1.3 Outline

In the next chapter, we discuss some of the most important findings related to networks and navigation, and we mention other methods presented in literature that are related to our problem. In Chapter 3, we formalize our problem, specifying the theoretical setting, the input and the objectives. We also present the model we use to describe human navigation in information networks. In Chapter 4, we describe the methods we use to embed an information network into a metric space that would best describe how people navigate the network. In Chapter 5, we present our solutions for recommending new links in an information network, in order to speed up navigation. In Chapter 6, we outline the details of the dataset we used to test our ideas, the setup of our experiments and the results for all criteria we considered. We also analyze our results and put them in perspective. Finally, in Chapter 7, we summarize what we did and present some directions for future work related to this problem.
Chapter 2

Background and related work

In this chapter, we will present some of the most important findings about network navigation and briefly discuss other models and algorithms related to our problem.

2.1 Small-world networks

The concept of a ‘small-world’ network was first used in the context of social sciences, in studies related to networks of people and estimating the chance that two individuals know each other or have a mutual acquaintance. One of the first and most important experimental studies made to show that people are connected to each other by short chains of contacts was conducted by Stanley Milgram in the 1960’s [15]. The experiment consisted in the following:

- Different people with various backgrounds were chosen at random from a town in the United States.
- Each of these individuals were given the name and address of another person in a different state and were asked to send a letter to that person by transmitting it through acquaintances that they knew on a first-name basis.
- Every participant was asked to forward the letter to the acquaintance they think is most likely to know the target person.

The main insight from this experiment was that, on average, the number of steps it takes for the letter to reach the target is around six. This is where the expression ‘six degrees of separation’ originated. After that, many networks in the real world, including information networks, have been shown to exhibit the ‘small-world’ phenomenon. Some examples include: the collaboration graph of film actors [22], the power grid in the United States [22], and the hyperlink graph of the World Wide Web [2].
A number of theoretical models have been proposed for random networks that have the ‘small-world’ property. In particular, we mention the Watts-Strogatz model [22] and the Barabasi-Albert model [5].

But besides the fact that there is a short path between any two people in the network, there is another important insight from Milgram’s experiment: most people are able to find these paths without knowing much about the target.

In [11], J. Kleinberg proposes a simple model for a random network that exhibits the ‘small-world’ property, and at the same time describes a simple algorithm for navigating to any target by only using local information of the graph at each step, for which the expected number of steps is exponentially lower than the network size.

We will briefly describe Kleinberg’s model and the main result associated to it. We consider a directed graph that consists of nodes arranged as lattice points in a two-dimensional $n \times n$ grid, and we denote the lattice distance between any two points $(x_1, y_1)$ and $(x_2, y_2)$: $d((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$. From each node $u$, we create edges to its neighbours on the grid that are within lattice distance of $p$, for a universal constant $p$, and call these ‘local-range’ edges. Also, given universal constants $q \geq 0$ and $r \geq 0$, for each node $u$ we add $q$ more edges to random nodes in the grid (so called ‘shortcut’ edges) using independent random trials; the probability that $v$ is chosen as the target in one of the trials is proportional to $[d(u, v)]^{-r}$.

Now let’s consider the following simple algorithm by which a message is transmitted from a node $u$ to another node $v$: at each step in the path, from all the adjacent nodes of the current node, choose the one that is closest to the target in terms of lattice distance. It was proven in [11] that for $p = q = 1$
and \( r = 2 \), the expected number of steps made by the algorithm above to send a message between any two nodes in the grid is at most \( a \cdot (\log n)^2 \), for some constant \( a \). This result generalizes for \( k \)-dimensional grids: the expected number of steps is polynomial in \( \log n \) if \( r = k \).

Part of the reason why the simple navigation algorithm mentioned above works is that the lattice distance is a good measure for comparing nodes with respect to how close they are to the target. The problem is that most graphs in the real world do not lie on a grid and we don’t have any lattice distance. What we could do instead is embed the graph into a metric space, which basically means finding a suitable distance function for the graph’s nodes.

In [9], an algorithm is proposed to add new edges to graphs with certain properties in order to improve navigation, which can be seen as a generalization of Kleinberg’s model. Suppose we have a graph \( G(N, L) \) and a metric embedding \((M,d)\) for the objects in \( N \). Suppose we also have the following property: for every pair of distinct nodes \((x,t)\) \(\in N \times N\), there is a node \( u \), such that \((x,u) \in L \) and \( d(u,t) < d(x,t) \). And let’s consider a similar navigation algorithm: at each step, choose from all the nodes adjacent to the current node the one that is the closest to the target, in terms of the distance in the embedding. Then if for each node \( x \), we add one ‘shortcut’ edge from it, with the probability that the other endpoint is \( y \) being proportional to \( 1/|\{u \in N|d(x,u) \leq d(x,y)\}| \), we have that the expected number of steps taken by the navigation algorithm is polynomial in \( \log n \).

Another algorithm for adding shortcut edges in order to create a small-world network is presented in [18]:

1. For each node, add one random ‘shortcut’ edge.
2. For a number of iterations, do the following:
   2.1 Choose two nodes uniformly at random \( s \) and \( t \).
   2.2 Navigate from \( s \) to \( t \), and save the path
   2.3 For each node in the path, independently with a fixed probability \( p \), replace its current shortcut edge with one that points to \( t \).

### 2.2 Modelling human navigation

We already mentioned navigation algorithms in the previous section. In this section, we are going to look at other ideas in literature. In [24], the authors propose a skeleton for navigation algorithms in information networks, and compare different ideas that fit the skeleton. The general algorithm involves doing a depth-first search for finding the target. At each step, rank all the nodes adjacent to the current node, using a scoring function, and push them
to the stack in that order. Several ideas are considered to rank the nodes, with respect to how close they might be to the target:

- The out-degree of the node
- The TF-IDF cosine similarity [17] with the target
- Expected-value navigation [20]
- Supervised learning of a ranking function, based on examples of navigation traces from humans
- Reinforcement learning

Another idea for describing human navigation in networks is based on the problem called ‘inverse reinforcement learning’ [1]: human navigation in networks is being modeled by a Markov-decision process and a reward function associated to it, which is derived from examples of navigation traces. The same idea, but with a slightly different solution, is exploited in [27] to model real-world navigation preferences of drivers, using GPS data collected from taxi drivers in a city network.

### 2.3 Link prediction

If we have a graph that evolves with time, one interesting problem to study is predicting the new edges that will appear next in the graph. Many networks in the real-world adjust themselves by adding new links to facilitate navigation, in particular social networks.

In [14], this problem is discussed for co-authorship networks: predicting new collaborations between authors in a particular research field. The general idea presented in the paper is ranking pairs of nodes, based on a score function that would express how likely it is for an edge to appear between those nodes. The following functions were among the ones considered:

- Graph distance: \( s(x, y) = \text{negated length of shortest path between } x \text{ and } y \)
- Common neighbors: \( s(x, y) = |\Gamma(x) \cap \Gamma(y)| \), where \( \Gamma(x) \) - the set of nodes adjacent to \( x \).
- Jaccard similarity between sets of adjacent nodes: \( s(x, y) = |\Gamma(x) \cap \Gamma(y)| / |\Gamma(x) \cup \Gamma(y)| \)
- Adamic/Adar score [3]: \( s(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} (1 / \log |\Gamma(z)|) \)
- Preferential attachment: \( s(x, y) = |\Gamma(x)| \cdot |\Gamma(y)| \)
- Katz \( \beta \) score [10]: \( s(x, y) = \sum_{l=1}^{\infty} \beta^l \cdot |\text{paths}_{x,y}^{(l)}| \), where \( \text{paths}_{x,y}^{(l)} \) is the set of all paths of length \( l \) from \( x \) to \( y \)
Predicting new collaborations in a co-authorship network has also been studied in [4] as a classification problem and different supervised learning methods have been evaluated, such as: SVM, multi-layer perceptron, Naive Bayes, K-nearest neighbors, RBF networks.

Another approach to the problem of link prediction has been proposed in [19], where the notion of probabilistic link prediction based on Markov chains is introduced.
Chapter 3

Problem statement

In this chapter, we will state our problem, by describing the setup and by defining the goals we wish to achieve.

3.1 Theoretical setting

We consider a set of documents. Each document has text content associated with it and also has links to other documents. As an example, we can think of a set of web pages and the hyperlinks between them. We only consider the case where the graph determined by the documents and the links is strongly connected - there exists a directed path between any two nodes. We will use the notation $G(N,E)$ to refer to the graph determined by the documents.

Furthermore, there is a set of categories, organized in a tree by how specific they are: the root of the tree is the most general category, and the leaves are the most specific categories. Each document belongs to one category. We can say that each document belongs to one node in the tree of categories.

We also have a set of navigation traces made by people who were trying to get from a starting document to a target document only by following the links in the documents. In some examples the person reaches the target, while in others the person gives up before reaching it and the path is not complete. At each step in the navigation, people were allowed to go back to the previous document and choose another link.

Moreover, we have a list of pairs of documents from the set, which represent link suggestions made by people, while navigating the document graph.
3. Problem statement

3.2 Objectives

We have two main goals. The first is to understand human navigation in such networks and to be able to describe it using a formal algorithm. The second goal is to find algorithms that would recommend new links between documents, which, if added to the network, would make human navigation easier - people would be able to reach their targets faster. We will discuss these two goals in detail.

3.2.1 Modelling human navigation

First of all, we assume that human navigation in an information network is based on how people decide which documents are similar to each other and which are not. A natural assumption people make is that similar documents are connected to each other. So at each step in the navigation, out of all the links in the current document, they follow the one corresponding to a document that is more similar to the target document than any other link, assuming that there is a higher chance to find a link to the target from a more similar document. So the first thing we need to do is to formalize this notion of similarity between documents.

We assume there is a metric space \((M, d)\), where \(d(x, y)\) represents the distance between two objects \(x, y \in M\), such that each document is represented by an object in \(M\). We call the representation of \(G(N, E)\) on \((M, d)\) a metric embedding. In other words, we have a distance function \(d\) associated to the documents. This function associates a non-negative real value to each pair of documents, which expresses how dissimilar those documents are.

Given this measure of dissimilarity between documents, we can formalize an algorithm to navigate the network. At every step, we try to follow the link that would get us as close to the target as possible in the chosen embedding. We allow the algorithm to go back to the previous node, if certain conditions are met, just as people would click the back button on a browser if they navigated a set of web pages. More specifically, if the best link we can follow doesn’t get us closer than the current document, and is even further than the previous document in the path, then we go back. We also limit the number of documents that the agent can see when looking for the target. We formalize these ideas into Algorithm 1.

Our goal is to find an appropriate metric embedding for the set of documents, which is as close as possible to the dissimilarity measure used by people when navigating the information network. In addition, the paths between any two nodes in the graph, computed using Algorithm 1 and the embedding, should be similar to the paths chosen by people.
Algorithm 1: Navigation using a metric embedding

**Input:**
- \( G(N, E) \) - graph
- \( d \) - distance function
- \( s \) - start node
- \( t \) - target node
- \( k \) - max number of nodes that can be visited

**Output:**
- \( P \) - path

**begin**

Initialize:
- \( P \) - empty list of nodes
- \( S \) - empty stack of nodes
- \( V \) - empty set of nodes \( // \) set of visited nodes

\( S.push(s) \)
\( P.add(s) \)
\( current \leftarrow s \)
\( previous \leftarrow null \)
\( i \leftarrow 1 \)

**while** \( i < k \) **do**

\( V \leftarrow V \cup \{current\} \)
\( next \leftarrow \arg \min_{n \in \Gamma(current) \setminus V} d(n, t) \)

**if** \( next = t \) **then**

\( // we have reached the target \)
\( P.add(t) \)
\( return P \)

**else if** \( (next \neq null) \land (d(next, t) < d(current, t)) \lor previous = null \land d(next, t) < d(previous, t) \) **then**

\( // go forward to the next node \)
\( i \leftarrow i + 1 \)
\( S.push(next) \)
\( P.add(next) \)
\( previous \leftarrow current \)
\( current \leftarrow next \)

**else**

\( // go back \)
\( S.pop() \)
\( current \leftarrow S.pop() \)
\( previous \leftarrow S.peek() \)
\( S.push(current) \)

**end**

**end**

return \( P \)

**end**
3. Problem statement

3.2.2 Recommending new links

After we find an appropriate embedding that we can use to model human navigation in the network, our next goal is to recommend new links that would make navigation easier. When people navigate without knowing the full structure of the network, they assume that certain connections exist and make their decisions based on those assumptions. If some of those assumptions are wrong, the time it takes to reach the target is longer than expected. We are looking for methods that recommend the best links to add, such that the average number of steps a person has to make to navigate between two arbitrary documents gets smaller.

These link recommenders should receive as input the set of documents, the graph, the chosen embedding and should just implement the simple interface described in Figure 3.1. At the first call, a recommender should return the best link to add, given the existing graph. At the next call, it should return the new best link to add, given that the previous recommended link was added to the graph. And so on.

There are two criteria that we can use to see if a link recommender gives good suggestions.

The first criterion is based on comparing the link recommendations with the ones given by people. If many of the first links given by the recommender were also suggested by people, this means we can use the recommender to add links that people would expect.

The second criterion involves computing the reduction in the average length of the path traversed by Algorithm 1 to navigate between two arbitrary documents, after adding the recommended links. We consider the length of a path as the number of distinct documents visited along the way, including the source and the target (if it was reached). The reason is that examining a document is usually a much more time consuming operation than following a link or going back to the previous document. If this average path length decreases substantially, it means that the link recommendations are suitable for improving navigation.
Chapter 4

Metric embeddings for information networks

In this chapter, we propose and analyse several ideas for metric embeddings of an information network. We will first discuss embeddings that exploit the network structure and the content of the documents. Then, we will look at how we can use examples of navigation traces to define similarity scores between documents, which can be used to define a distance function. In the last section, we propose a method to create an embedding by using a non-negative linear combination of distance functions and learning the weights using examples of human navigation traces.

We consider $N$ to be the set of documents. Throughout the paper, we will abuse notation and write $N \subset M$, identifying the documents with their representations in the embedding.

In order to obtain a metric embedding, we just need to find a distance function between objects. As a reminder, a distance function $d : M \times M \rightarrow \mathbb{R}$ should have the following properties:

D1) $d(x, y) \geq 0$, $\forall x, y \in M$
D2) $d(x, y) = 0 \Leftrightarrow x = y$
D3) $d(x, y) = d(y, x)$, $\forall x, y \in M$
D4) $d(x, y) + d(y, z) \geq d(x, z)$, $\forall x, y, z \in M$ (triangle inequality)

4.1 Embeddings using network structure and information content

In this section, we will define several distance functions based on the content of the documents, the topics they belong to, and the links between them.
These ideas are not new. We will use these functions both as baselines in our experiments presented in Chapter 6 and as part of our proposed distance function, which we will present in Section 4.3.

4.1.1 Graph distance

Our set of documents and the links between them determine a directed graph. Let’s consider the function $g : N \times N \rightarrow \mathbb{R}$, where $g(x, y)$ is the number of edges in the shortest path from $x$ to $y$. Then we can define the following distance function:

$$d_G(x, y) = g(x, y) + g(y, x) \quad (4.1)$$

It can be easily seen that $d_G$ satisfies the properties D1, D2 and D3. To prove D4, we first notice that: $g(x, y) + g(y, z) \geq g(x, z), \forall x, y, z \in N$. Otherwise, there would be a shorter path from $x$ to $z$ that goes through $y$, whose length is less than $g(x, z)$. The property D4 follows from:

$$d(x, y) + d(y, z) = g(x, y) + g(y, x) + g(y, z) + g(z, y) =$$

$$= (g(x, y) + g(y, z)) + (g(z, y) + g(y, x)) \geq g(x, z) + g(z, x) = d(x, z)$$

This is the simplest idea when it comes to metric embeddings for a graph, but the function above might prove useful when it is combined with other functions.

4.1.2 TF-IDF angle distance

Term frequency - inverse document frequency (tf-idf) [17] is a statistic used widely in information retrieval that expresses the importance of a word (term) in a given document in a collection. It is usually defined as: $\text{tf-idf}_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log(n/df_t)$, where $\text{tf}_{t,d}$ is the term frequency of term $t$ in document $d$, $df_t$ is the number of documents that contain the term $t$, and $n$ is the number of documents in the collection.

We consider each document $x$ to be represented by a real-valued vector of tf-idf weights $\vec{t_x}$. Then the following function is a metric:

$$d_T(x, y) = \arccos(\cos(\vec{t_x}, \vec{t_y})) = \arccos \frac{\vec{t_x} \cdot \vec{t_y}}{\|\vec{t_x}\| \cdot \|\vec{t_y}\|} \quad (4.2)$$

The cosine of the angle between the tf-idf vectors is used in information retrieval as a similarity measure between documents. Therefore, the angle between the vectors can be seen as a dissimilarity measure between documents and fits our framework.
4.1. Embeddings using network structure and information content

It can be easily seen that $d_T$ satisfies D1, D2 and D3. In order to prove D4, we note that the angle between the vectors $\overrightarrow{t_x}$ and $\overrightarrow{t_y}$ represents the geodesic distance between the corresponding normalized vectors $\overrightarrow{t_x} / \| \overrightarrow{t_x} \|$ and $\overrightarrow{t_y} / \| \overrightarrow{t_y} \|$ on the unit hypersphere. The geodesic distance on a hypersphere satisfies the triangle inequality. Therefore, $d_T$ also satisfies it.

4.1.3 Jaccard distance between sets of inlinks

Let’s consider the following function: $I : N \to 2^N, I(x) = \{ y \in N | x \in \Gamma(y) \}$. For each node $x$, it gives the set of all nodes that have a link to $x$. It makes sense to think that two documents that have similar sets of documents that point to them are themselves similar as well. One way to measure dissimilarity between sets is the Jaccard distance: $J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$.

This is known to be a metric for a set of sets [13]. One problem is that we might have two documents with exactly the same set of inlinks. Taking this into account, we can define the following function:

$$ d_J(x, y) = \begin{cases} 0 & \text{if } x = y \\ J(I(x), I(y)) + \epsilon & \text{otherwise} \end{cases} \quad (4.3) $$

where $\epsilon$ is a constant, with $0 < \epsilon \ll 1/|N|$. It is easy to see that $d_J$ satisfies D1, D2, D3. Given that $J$ satisfies D4, it can be easily proven that $d_J$ also satisfies D4.

This distance function is inspired from the notion of co-citation [21], which is a semantic similarity measure used for documents, especially for research papers.

4.1.4 Distance in the category tree

As we mentioned in Section 3.1, our documents belong to categories, which are organized in a tree. We can use this information to define another distance function. Let’s consider $g_C(x, y)$ the distance in the category tree between the category associated to $x$ and that associated to $y$. We might have two different documents in the same category, so, taking that into account, we can define the following distance function:

$$ d_C(x, y) = \begin{cases} 0 & \text{if } x = y \\ g_C(x, y) + 1 & \text{otherwise} \end{cases} \quad (4.4) $$

It can be easily proven that $d_C$ is a metric. This function measures how far two documents are from each other in the topic space. It was observed in
4. Metric embeddings for information networks

[25] that people navigate information networks by first going to high-degree hubs, which usually belong to more general categories, and are then guided by content similarity to reach their targets, which usually involves articles from the same category as the target. So the distance in the category tree might be a good measure of how far two documents are from each other in the semantic space in which humans think.

The idea of a category tree distance was also used in [25] to analyze human navigation.

4.2 Embedding based on counting ordered pairs in navigation traces

Let’s consider a path $p$ traversed by a person when navigating between two documents, $s$ and $t$. The intermediary documents visited were the following, in this order: $(p_1, p_2, \ldots, p_m)$. We consider all ordered pairs of nodes $(p_i, p_j)$, for which $i < j$ and the person had to go through $p_i$ in order to reach $p_j$. In other words, if we consider the search tree made by the person when looking for the target, $p_i$ is part of the shortest path between $s$ and $p_j$ in the search tree.

**Figure 4.1:** An example of a search tree corresponding to a path
4.3 Learning a linear combination of distance functions

For example, in Figure 4.1, we have the following ordered pairs that satisfy this property: \((p_1, p_2), (p_1, p_3), (p_1, p_4), (p_1, p_5), (p_4, p_5)\). We will use the following notation: \(p_i \prec_p p_j\). We also consider by convention that \(p_i \prec_p t, \forall i \in \{1, \ldots, m\}\), regardless whether the person reached the target \(t\) or not.

We define the following similarity measure between documents:

\[
s(x, y) = |\{p \text{-path}| x \prec_p y\}| \tag{4.5}
\]

In other words, a document is similar to another if there were many cases where people passed through \(x\) to get to \(y\). A high value for this measure means that most people assume it’s easy to get from document \(x\) to document \(y\). Note that this measure is not symmetric. Given that for most pairs we will have a score of 0, we can also use the information about the existing edges to improve the similarity score:

\[
s'(x, y) = s(x, y) + 1_{y \in \Gamma(x)} \tag{4.6}
\]

We can define a distance function based on this measure:

\[
d_p(x, y) = \begin{cases} 
0 & \text{if } x = y \\
1 + \frac{1}{s'(x, y) + s'(y, x) + 1} & \text{otherwise}
\end{cases} \tag{4.7}
\]

It can be easily seen that \(d_p\) satisfies D1, D2 and D3. For D4, we have:

\[d(x, y) + d(y, z) \geq 2 \geq d(x, z), \forall x, y, z \in N\]

4.3 Learning a linear combination of distance functions

We consider a non-negative linear combination of distance functions:

\[
d(x, y) = w_1 \cdot d_1(x, y) + w_2 \cdot d_2(x, y) + \cdots + w_k \cdot d_k(x, y) \tag{4.8}
\]

where \(w_i \geq 0, \forall i \in \{1, \cdots, k\}\). It’s easy to prove that if \(d_1, d_2, \cdots, d_k\) are metrics, then \(d\) is also a metric. The problem is in choosing the right weight vector \(w\). To solve this, we need to remember what our actual goal is and what data we have available.

We look at the forward steps in the navigation traces made by people. If at a certain point in a path, out of all the links in the current document, a person chose one particular link to get closer to the target, then our distance function should reflect that. More precisely, if a person is at a document \(x\),
wants to reach target \( t \), and chooses to follow a link from \( x \) that leads to \( y \), then our distance function should satisfy the following:

\[
d(y, t) < d(z, t), \ \forall z \in \Gamma(x) \setminus \{y\} \tag{4.9}
\]

We can write the inequalities above in the following way:

\[
w^T \cdot (d_1(y, t), d_2(y, t), \ldots, d_k(y, t)) > w^T \cdot (d_1(z, t), d_2(z, t), \ldots, d_k(z, t)) \Rightarrow
\]

\[
w^T \cdot [(d_1(y, t), \ldots, d_k(y, t)) - (d_1(z, t), \ldots, d_k(z, t))] > 0
\]

We can use the Ranking SVM method [8] to find the weight vector \( w \), given these inequalities. Each pair of nodes can be seen as an object that has a feature vector \( x_i \) associated to it, which consists of the values of all distance functions for that pair of nodes. If we have a set \( S \) of ordered pairs of indices \((i, j)\), for which \( w^T (x_i - x_j) > 0 \), then we can use them as training examples for a learning to rank problem. We can formulate this as a convex optimization problem:

\[
\min_{w, \xi} \frac{1}{2} ||w||^2 + C \sum_{i,j} \xi^{(ij)} \tag{4.10}
\]

subject to:

\[
w^T (x_i - x_j) \geq 1 - \xi^{(ij)} \text{ and } \xi^{(ij)} \geq 0, \ \forall (i, j) \in S
\]

Additionally, we have the constraint:

\[
w_l \geq 0, \ \forall l \in \{1, \ldots, k\}
\]

If we have a large training set, we can use stochastic gradient descent [6] to solve the problem. After each update of the weight vector, we project \( w \) to the non-negative subspace, by just zeroing the negative components.

Basically, this means we can combine all the distance functions we defined above into a non-negative linear combination, which might describe the semantic space in which people think better than the individual distance functions.

When people navigate such a network, their decisions are based on a variety of factors. Therefore, as the different functions exploit different features from the network structure, the content and the navigation traces, combining these functions makes sense for the problem of modelling human navigation.
In this chapter, we will propose two methods to recommend links in an information network, in order to facilitate human navigation. Each method is optimized for one of the two criteria we discussed in Section 3.2.2.

5.1 Matching user suggestions

We want to choose, out of all the possible new links that can be added, the ones that are most likely to be suggested by people. Basically, we want to rank all these links by a score that would correspond to how likely it is for them to be suggested. We are looking for a function $f : N \times N \rightarrow \mathbb{R}$, that would give a score to each pair of documents. Using such a function, we can sort these pairs in descending order of the score, remove the pairs that are already links in the graph, and recommend the remaining pairs one by one, in that order.

We look at the following type of functions:

$$f(x, y) = w_1 \cdot f_1(x, y) + w_2 \cdot f_2(x, y) + \cdots + w_k \cdot f_k(x, y),$$

(5.1)

where $f_1, \cdots, f_k$ represent features associated to the pairs of documents. We can find the appropriate weights in the following way: we take a sample of the pairs of documents that represent link suggestions made by people and a sample of the rest of the pairs. Consider $S$ the set with the first group of samples and $T$ the set with the second group of samples. Then for each ordered pair $((x, y), (z, t))$ with $(x, y) \in S$ and $(z, t) \in T$, we should have $f(x, y) > f(z, t)$. We can write this as a learning to rank problem and solve it using Ranking SVM, as shown in Section 4.3.

As features, we considered the following:

- $d(x, y)$ - the distance in the chosen embedding between $x$ and $y$. If we choose a good embedding, then this value should indicate how dissim-
5. Link recommendation

5.1 Similar x and y are from each other, and people would expect to see links between similar documents. And if we use a linear combination of distance functions, as described in Section 4.3, then this value should already include all kinds of features related to the pair of documents.

- PageRank(x), PageRank(y) - the PageRank score [16] indicates the probability that at a certain point in time one would be at the given node, when doing a random walk in the graph. These features might be helpful, as a low PageRank for y might mean that there are not enough links to y, and a high PageRank for x might mean that people were more likely to stop their search at x and might have suggested more links from x.

- A similarity measure based on the following: We consider a simpler navigation algorithm than Algorithm 1 presented before: at each step, choose the link that gets you as close as possible to the target, in the embedding, but only if it gets you closer than the current node, otherwise just stop. This means that if we are at node x, and we want to get to t, and we have \( d(y, t) \geq d(x, t) \), \( \forall y \in \Gamma(x) \), then we stop the navigation. We can consider as similarity measure for the pair \((x, y)\) the value equal to the number of times this navigation algorithm stopped at x, when trying to get to y from all the other nodes in the graph. The motivation is that if the navigation from different starting points stops at a certain document, this means there should probably be a link between that document and the target.

5.2 Reducing the average path length

For \( N \) - the set of nodes, \( E \) - the set of edges, and \( k \) - the maximum number of documents that can be visited in a navigation, we define the length of a path between \( x \) and \( y \) traversed while navigating using Algorithm 1 as the number of distinct documents visited during navigation, including the starting node and the target (if it was reached). We will use the notation \( \ell_{(N,E,k)}(x,y) \).

Let’s say we want to add at most \( m \) new links to our network, such that we maximize the decrease in the sum of all path lengths. We consider the following function:

\[
f: 2^{N \times N} \to \mathbb{R}, \quad f(S) = \sum_{x,y \in N} \left[ \ell_{(N,E,k)}(x,y) - \ell_{(N,E\cup S,k)}(x,y) \right]
\]  

(5.2)

Given that, we can formulate the following optimization problem that matches our objective:

\[
\arg \max_{S \subseteq (N \times N) \setminus E, |S| \leq m} f(S)
\]  

(5.3)
5.2. Reducing the average path length

As the number of sets $S$ that fit the requirement is exponential in $m$, we cannot try all possibilities. We will use the concept of submodularity to construct a method that targets this problem. For a detailed explanation of submodularity and maximizing submodular functions you can consult the survey [12]. Here we will briefly present the basic facts we need in order to explain our method.

A set function $f : 2^V \to \mathbb{R}$ is said to be monotone submodular iff $\forall S, T \subseteq V$, with $S \subseteq T$, and $\forall e \in V$, we have: $f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$.

The intuition behind this property is that the marginal gain of any new element added to a set $S$ gets smaller as $S$ gets bigger.

Let’s consider the problem of maximizing a monotone submodular function:

$$\max_{S \subseteq V} f(S), \text{ where } |S| \leq m$$

This problem was proven to be NP-hard for certain classes of submodular functions [12]. But there is a simple greedy algorithm that gives a result, which is at least $1 - \frac{1}{e}$ of the optimum [12]. Algorithm 2 returns a set $S$ that gives this guarantee by finding at each step the element that would give the highest marginal benefit and adding it to $S$.

**Algorithm 2: Greedy submodular function maximization**

```plaintext
begin
    $S \leftarrow \emptyset$
    while $|S| < m$ do
        $e^* \leftarrow \arg \max_{e \in V} f(S \cup \{e\}) - f(S)$
        $S \leftarrow S \cup \{e^*\}$
    end
    return $S$
end
```

Unfortunately, the function $f$ we defined above is not monotone submodular. In Figure 5.1 we show a counterexample. We consider two new possible edges $e_1$ and $e_2$ for the existing graph. If we denote $S = \emptyset$ and $T = \{e_1\}$, then $f(S \cup \{e_2\}) - f(S) = 2$ (the length from $y$ to $z$ reduces from 4 to 2), and $f(T \cup \{e_2\}) - f(T) = 2 + 2 = 4$ (the length from $y$ to $z$ reduces from 4 to 2, and the length from $x$ to $z$ reduces from 5 to 3).

We consider a simpler problem. We have a fixed target $t$, and we want to recommend links to $t$, such that we maximize the reduction in the sum of
Figure 5.1: Example which shows that $f$ is not submodular

path lengths from all nodes to $t$. We define the following function:

$$f_t : 2^{N \times \{t\}} \rightarrow \mathbb{R}, \quad f_t(S) = \sum_{x \in N} \ell_{(N,E,k)}(x,t) - \ell_{(N,E \cup S,k)}(x,t) \quad (5.5)$$

Then we have the following result:

**Theorem 5.1** Function $f_t$ is monotone submodular, $\forall t \in N$.

**Proof** We define the following function:

$$f_{s,t} : 2^{N \times \{t\}} \rightarrow \mathbb{R}, \quad f_{s,t}(S) = \ell_{(N,E,k)}(s,t) - \ell_{(N,E \cup S,k)}(s,t) \quad (5.6)$$

Then we can write:

$$f_t(S) = \sum_{s \in N} f_{s,t}(S) \quad (5.7)$$

We will prove that $f_{s,t}$ is monotone submodular for any fixed nodes $s$ and $t$. Let's consider the navigation from $s$ to $t$. We describe the path from $s$ to $t$ by the list of documents visited along the path, in the order they were first seen: $(s = p_1, p_2, \cdots, p_n)$. If we add an edge from $p_i$ to $t$, then the length of $p$ reduces from $n$ to $i + 1$, as the navigation algorithm will immediately follow a link to the target when it sees one. We consider two arbitrary sets $S, T \in N \times \{t\}$, with $S \subseteq T$, and an arbitrary node $x \in N$.

We examine the following cases:

1. $p_i \neq x, \forall i \in \{1, \cdots, n - 1\}$.
   Then the marginal gain is 0 for both $S$ and $T$:
   $$f_{s,t}(S \cup \{(x,t)\}) - f_{s,t}(S) = f_{s,t}(T \cup \{(x,t)\}) - f_{s,t}(T) = 0$$

2. $\exists j \in \{1, \cdots, n - 1\}$, such that $x = p_j$.
   Then we have several cases:
5.2. Reducing the average path length

2.1 \((p_i, t) \notin T, \forall i \in \{1, \ldots, n - 1\}\).

Then:
\[
fs,t(S \cup \{(x, t)\}) - fs,t(S) = fs,t(T \cup \{(x, t)\}) - fs,t(T) = n - (j + 1)
\]

2.2 \(i_T = \min_{(p_i, t) \in T} i, \) and \((p_i, t) \notin S, \forall i \in \{1, \ldots, n - 1\}\).

Then:
\[
fs,t(S \cup \{(x, t)\}) - fs,t(S) = n - (j + 1)
\]
\[
fs,t(T \cup \{(x, t)\}) - fs,t(T) = \max\{(i_T + 1) - (j + 1), 0\}
\]
\[
n - (j + 1) \geq 0 \quad \text{or} \quad n \geq i_T + 1 \Rightarrow
\]
\[
fs,t(S \cup \{(x, t)\}) - fs,t(S) \geq fs,t(T \cup \{(x, t)\}) - fs,t(T)
\]

2.3 \(i_T = \min_{(p_i, t) \in T} i, \) and \(i_S = \min_{(p_i, t) \in S} i.\)

Then:
\[
fs,t(S \cup \{(x, t)\}) - fs,t(S) = \max\{(i_S + 1) - (j + 1), 0\}
\]
\[
fs,t(T \cup \{(x, t)\}) - fs,t(T) = \max\{(i_T + 1) - (j + 1), 0\}
\]
\[
S \subseteq T \Rightarrow i_S \geq i_T \Rightarrow
\]
\[
\max\{(i_S + 1) - (j + 1), 0\} \geq \max\{(i_T + 1) - (j + 1), 0\}
\]

It can be easily proven that the sum of monotone submodular functions is also monotone submodular. Therefore, given that the function \(f_{s,t}\) is monotone submodular, \(\forall s \in N,\) then the function \(f_t\) is also monotone submodular.

\(\Box\)

This result means that we can use the greedy algorithm to minimize the average path length from any node to \(t,\) given that we can only add \(m\) links to \(t.\)

Now we get back to the global problem. Let’s consider the following function:
\[
f'_t : N \times N \rightarrow \mathbb{R}, \quad f'_t(S) = f_t(S \cap (N \times \{t\})),
\]
for a fixed node \(t \in N.\)

**Theorem 5.2** Function \(f'_t\) is monotone submodular, \(\forall t \in N.\)
5. Link recommendation

**Proof** We consider two arbitrary sets $S, T \subseteq N \times N$, with $S \subseteq T$, and an arbitrary pair of nodes $(x, y) \in N \times N$. Let $S' = S \cap (N \times \{t\})$ and $T' = T \cap (N \times \{t\})$. We have: $S' \subseteq T'$.

If $y \neq t$, the marginal gain for both $S$ and $T$ is zero.
If $y = t$, we have:

\[
(S \cup \{(x, y)\}) \cap (N \times \{t\}) = (S \cap (N \times \{t\})) \cup \{(x, y)\} = S' \cup \{(x, y)\} \Rightarrow \\
f_i(S \cup \{(x, y)\}) = f_i(S' \cup \{(x, y)\}) \Rightarrow \\
f_i(S \cup \{(x, y)\}) - f_i(S) = f_i(S' \cup \{(x, y)\}) - f_i(S')
\]

Similarly:

\[
f_i(T \cup \{(x, y)\}) - f_i(T) = f_i(T' \cup \{(x, y)\}) - f_i(T')
\]

From the above and from the fact that $f_i$ is monotone submodular, we have that $f_i'$ is also monotone submodular.

\[\square\]

Given this, we can define the function $f' : N \times N \to R$, as:

\[
f'(S) = \sum_t f'_i(S) = \sum_{s,t} \ell_{(N,E,k)}(s,t) - \ell_{(N,E,(S \cap (N \times \{t\})) \cup (N \times \{t\}))}(s,t) \quad (5.9)
\]

This function is monotone submodular and resembles the function $f$ that we defined in equation (5.2) for the global objective. We believe that maximizing $f'$ gives in practice a good approximation to the optimum value for the function $f$. The greedy algorithm can be used to maximize $f'$ for a near-optimal solution.

We note that the same idea can be applied to other definitions for the length of a path, like: the number of forward steps or the total number of steps (including backward steps).
In this chapter, we present the experiments we made using the methods described in the previous two chapters. The dataset we used corresponds to the problem statement defined in Chapter 3.

6.1 Dataset

We used the data collected from the Wikispeedia game [23, 26]. The idea of the game is that you start from a Wikipedia article and you need to get to another article, only by following the links in the articles encountered along the way. The documents represent a subset of English Wikipedia articles from 2007, selected for schools. In our experiments, we consider the biggest strongly connected component from this collection, which consists of about 4000 articles. There are approximately 110000 links between these articles in total.

The examples of navigation traces were collected from people playing the game on the website. We use a total of about 50000 finished paths and 16000 unfinished paths traversed by users while playing the game.

There is also a category tree as the one described in the problem statement. In Figure 6.1, you can see some of the top nodes in the category tree. We also use a set of approximately 1000 link suggestions made by users while playing the game.

6.2 Results

6.2.1 Embeddings

We used two criteria to compare different embeddings on how good they describe human navigation in the network. Both of them are based on
analysing each path traversed by people, giving a score to the embedding for that path, and taking the average over all paths considered in the test set. Both scores are between 0 and 1, with 1 being the best score.

The first score is computed in the following steps for each path:

1. Compute the path taken by Algorithm 1, between the same starting point and target.
2. Return the Jaccard similarity between the set of nodes in the path traversed by the person and the set of nodes in the path traversed by the algorithm.

A high score means that the navigation algorithm uses the same intermediary nodes as people do, which is what we want. We will refer to this criterion as *Path Set Similarity*.

The second criterion is based on computing a score for each human path in the following way:

1. For each forward step in the path (from node $x$ to node $y$, in order to reach target $t$):
   1.1 Order all the nodes adjacent to $x$, with respect to how close they are to the target in the embedding,
6.2. Results

1.2 Compute the percentile rank of \( y \) in this ordering:

\[
r_y = \frac{g_y + 0.5 \cdot e_y}{|\Gamma(x)|},
\]

where

\[
g_y = |\{ z \in \Gamma(x) | d(z, t) > d(y, t) \}|,
\]

\[
e_y = |\{ z \in \Gamma(x) | d(z, t) = d(y, t) \}|.
\]

2. Return the average percentile rank for all forward steps.

This score shows how close the distances in the embedding match the way people navigated the network. We will refer to this score as Human Step Rank.

We tested the four distance functions described in Section 4.1, the function described in Section 4.2, and two non-negative linear combinations of functions, optimized using the idea from Section 4.3. The first combines all the five functions we defined, and the second combines only the four functions that are not based on navigation traces. The motivation for also considering the second combination is that the resulting embedding can also be used for other similar information networks, with the same weights as the ones obtained for this dataset, without the need for collecting navigation traces for those networks.

From the set of human navigation traces, for each test, a sample of 20% was taken to train the learned embeddings, and the rest was used as a test set.

<table>
<thead>
<tr>
<th>Distance function</th>
<th>Human Step Rank</th>
<th>Path Set Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph distance ( d_G )</td>
<td>0.695</td>
<td>0.251</td>
</tr>
<tr>
<td>TF-IDF distance ( d_T )</td>
<td>0.817</td>
<td>0.328</td>
</tr>
<tr>
<td>Inlinks set distance ( d_I )</td>
<td>0.761</td>
<td>0.239</td>
</tr>
<tr>
<td>Category tree distance ( d_C )</td>
<td>0.68</td>
<td>0.131</td>
</tr>
<tr>
<td>Ordered pair counts distance ( d_P )</td>
<td>0.788</td>
<td>0.372</td>
</tr>
<tr>
<td>Linear combination ( [d_G, d_T, d_I, d_C] )</td>
<td>0.821</td>
<td>0.348</td>
</tr>
<tr>
<td>Linear combination ( [d_G, d_T, d_I, d_C, d_P] )</td>
<td>0.837</td>
<td>0.386</td>
</tr>
</tbody>
</table>

Table 6.1: Results for embeddings

In Table 6.1 we present the results given by all distance functions for both criteria we discussed above.

Out of the predefined functions defined in Section 4.1, the TF-IDF distance has the best performance. This might be explained by the fact that this
function uses the biggest quantity of information and the most features out of all the other predefined functions.

The function based on counting ordered pairs in navigation examples gives relatively good results for both criteria. It beats all the predefined functions, except for the TF-IDF distance for Human Step Rank. The fact that it performs well means that people have common strategies for navigating the network, and we can model human navigation by just using the data from a sample of the paths traversed by users.

As expected, the best results were obtained by the linear combination of all the distance functions. As each of them exploits different features, combining them means using all sorts of features about the network to create a suitable distance function between documents.

After scaling the values of all distance functions to the range \([0, 1]\), learning linear combinations of these functions gave the weights in Table 6.2. These weights can be interpreted as how important each function is in modelling human navigation.

<table>
<thead>
<tr>
<th>Distance function</th>
<th>Weight vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear combination ([d_G, d_T, d_J, d_C, d_P])</td>
<td>([0.00, 0.26, 0.14, 0.12, 0.95])</td>
</tr>
<tr>
<td>Linear combination ([d_G, d_T, d_J, d_C])</td>
<td>([0.27, 0.85, 0.44, 0.08])</td>
</tr>
</tbody>
</table>

Table 6.2: Weights obtained after learning linear combinations of distance functions

For both combinations it can be noted that the learned weights are higher for the functions that have higher individual scores for the two criteria.

The linear combination of all distance functions will be used for the experiments related to link recommendation, which are presented next.

### 6.2.2 Link recommendation

As mentioned in Section 3.2.2, we have two criteria for link recommenders: matching the recommendations against the ones given by users, and computing the reduction in the average length of the paths made by Algorithm 1 between any two nodes. For the first criterion, we will use precision-recall curves. For the second criterion, we measure the average path length after adding a constant number of links.

Besides the two methods we proposed in Chapter 5 (we will refer to them as LinkRankLearning and SubmodularSum), we also tested some other ideas from literature on the same dataset. These ideas include: the edge sampling algorithm described in [9] (we will call it ShortcutEdgeSampling), the
6.2. Results

The rewiring algorithm presented in [18] (EdgeRewiring), and the ranking algorithm described in [14] with the best scoring function in their results - the Adamic-Adar score [3] (AdamicRanking).

For the LinkRankLearning method, out of the link suggestions given by users, we took 20% of them and used them as training set for RankingSVM, and the rest were used for testing.

In Figure 6.2 we show the precision-recall curves of all methods, with respect to the link suggestions made by users. The LinkRankLearning method gives the best results by far. Considering that there is a set of 16 million possible new links that can be added to the graph, and only 800 user suggestions in our test set, a precision of more than 10% is actually a lot. The Submodular-Sum method also gives fair results, even though it is not optimized for this criterion.

In Table 6.3 we present the titles of the articles involved in the first new link recommendations by LinkRankLearning. All the links seem appropriate, given the content of the articles and the structure of the network.
One of the reasons why the other methods we tried don’t have good results is that they are not really made for this exact problem. The methods \textit{ShortcutEdgeSampling} and \textit{EdgeRewiring} are used to create small-world networks in existing graphs, by sampling new edges from each node. Their goal is not to find the best new links to add. And the link prediction methods are in general used to predict new links that might appear in the network and are not directly related to improving navigation.
Next, we look at the second goal - reducing the average length of the paths traversed by Algorithm 1 between any two nodes. As a reminder, we consider the length of a path as the number of distinct documents visited along the way. In Algorithm 1, if after traversing \( k \) distinct documents the algorithm didn’t reach the target, then we consider the path length to be \( k \). In our experiments, we chose \( k = 30 \), as more than 99.5% of the paths traversed by people included at most 30 distinct documents. For each method, we compute the average path length after adding 1000, 2000 and 5000 new links. The method that has the lowest value for each case is the best.

<table>
<thead>
<tr>
<th>Link recommender</th>
<th>Average path length after adding ( m ) new links</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( m = 0 )</td>
</tr>
<tr>
<td>ShortcutEdgeSampling</td>
<td>7.32</td>
</tr>
<tr>
<td>EdgeRewiring</td>
<td>6.85</td>
</tr>
<tr>
<td>AdamicRanking</td>
<td>7.23</td>
</tr>
<tr>
<td>LinkRankLearning</td>
<td>6.40</td>
</tr>
<tr>
<td>SubmodularSum</td>
<td>4.90</td>
</tr>
</tbody>
</table>

Table 6.4: Average path length after adding new links

In Table 6.4, we present the results for all recommenders. The average path length before adding any new links is 7.32, which is close to the average length of human paths in our dataset: 6.47. One can see that *SubmodularSum* gives much better results than all the other methods. It is not surprising, as this method is tailored for this objective. Even though the submodular function it is maximizing is not exactly the same as the real objective, we can see that it gives good results in practice. By just adding 1000 new links, we reduce the average number of documents traversed in all paths by 33%.

We show the first links recommended by *SubmodularSum* in Table 6.5. As you can see, there are many link recommendations from hubs (like *United States* and *English language*, which are among the articles in the network that contain the most links) to articles that have the fewest inlinks. This can be explained by the fact that if an article has very few references, the best option to connect it to the rest of the network is just to create a link to it from a node that is easily reachable by all the other nodes. One can notice that the hubs chosen to be connected to articles that are hard to reach are in the same topic as those articles (e.g. *India* → *Geography of India*, *Human* → *Homo floresiensis*, *English language* → *Stuttering*). Nevertheless, the link recommendations are appropriate for the state of the network, as there are many nodes that have
very few inlinks in this small and old Wikipedia subset. If we consider a more dense information network, with fewer isolated nodes, we might get much better link suggestions using \textit{SubmodularSum} than in this case.
7.1 Summary

In this work, we studied two related problems about navigation in information networks: modelling human navigation and improving it by adding new links in the network.

We presented a model that describes how people navigate information networks, based on a distance measure between documents and a navigation algorithm that uses this measure. We create this distance function as a linear combination of other functions, which use data about the network structure, the content of the documents, and examples of navigation traces made by people.

We also proposed two solutions for recommending new links in a network, which would make human navigation easier.

The first method is based on learning a ranking function for pairs of nodes from examples of link suggestions made by people. This method gives link recommendations that are relevant to the network and similar to unseen link suggestions from humans.

The second method has the goal of reducing the average length of the path made by the navigation algorithm we defined that models human navigation between any two nodes in the network. It is based on maximizing a monotone submodular function that is closely related to this goal. In our experiments, the method performed very well. By adding a relatively small number of new links to the network, it reduced the average path length by more than 30%.
7. Conclusion

7.2 Future work

There are plenty of directions one can take to obtain new insights about this problem and to improve the methods we proposed.

First of all, one should try to test these ideas on larger datasets. Even without examples of navigation traces, one can still try to use the linear combination of predefined distance functions, with the same weights as the ones presented in Table 6.2, in order to get link recommendations using the two methods we proposed and see if they are relevant to the network.

If we are talking about modelling human navigation, there are many things that can be changed or improved. One might try to define other distance functions that are suitable for an information network and include them in the linear combination. Also, there might be better criteria to compare different embeddings on how good they are for modelling human navigation.

Furthermore, one might think of a different model than the one we chose, which is based on a metric embedding. There are probably better models that involve something more complex than just a distance function. And the navigation algorithm we used might not be the best option either.

For link recommendation, a direction for improvement can be adding new features for the method LinkRankLearning. Also, one can try to define other submodular functions related to the objective of reducing the average path length in the network. The function proposed in this paper might not be the best one to use for optimizing this objective.

Another direction for future work would be to study the possibility of combining the two ideas we discussed for link recommendation. If we want a recommender that performs reasonably well for both criteria we discussed, then we can combine the two recommenders into one: at each call for a new link we would call one of the two recommenders, based on a probability distribution or on more complex rules.

We focused on modelling and improving navigation in information networks. But similar ideas can be applied to all types of networks where navigation is involved, like: road networks, computer networks, social networks. One can define similar distance functions for these networks and implement the same ideas for link recommendation that we described here.
Bibliography


Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor’s thesis, Master’s thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

NAVIGABILITY IN INFORMATION NETWORKS

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):
GRECU

First name(s):
MIRCEA

With my signature I confirm that
− I have committed none of the forms of plagiarism described in the ‘Citation etiquette’ information sheet.
− I have documented all methods, data and processes truthfully.
− I have not manipulated any data.
− I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date
ZÜRICH, 25.07.2014

Signature(s)