TU Graz

Fakultät

Institut für Wissenstechnologien

Masterarbeit

Navigation in Informationsnetzwerken mit unvollständigem Wissen

Autor: Stefan Knecht

Stefan.Knecht@student.tugraz.at

1. Prüfer: Helic, Denis, Assoc.Prof. Dipl.-Ing. Dr.techn.

Abgabedatum:

I Kurzfassung

Diese Arbeit beschäftigt sich mit dem Thema der simulierten Navigation durch ein Informationsnetzwerk. Es wird ein neues Verfahren entwickelt und getestet, bei dem die Navigation aufgrund der möglichen Wissensbasis der User simuliert wird. Speziell wird auf die Entwicklung dieses Algorithmus eingegangen und die verschiedene Definitionen solch einer Wissensbasis gezeigt. Resultat, Vergleiche mit anderen Verfahren, Problemstellungen und deren mögliche Lösungen werden im späteren Abschnitt erläutert und diskutiert.

Abstract

This paper investigates the simulated navigation through information networks. A new procedure is developed and testet, where the navigation of users based on their possible knowledge base is simulated. The paper especially concentrates on the development of the algorithm. Additionally the various definitions of the knowledge base are pointed out. Results, comparisons to other procedures, problems and possible solutions are ecemplified and discussed in the following chapters.

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008 Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....
date (signature)

Danksagung

Ich danke meinen Eltern, dafür dass sie mir das Studium erst ermöglicht und mich unterstützt haben.
Des Weiteren danke ich allen Kollegen und Freunden, die während des Studiums bei Fragen immer geholfen, speziell Florian Geigl, der mir in der Masterarbeit ein große Hilfe war. Dazu gehört auch mein Betreuer Herr Assoc.Prof. Dipl.-Ing. Dr.techn. Denis Helic, der immer Zeit für die Besprechung entstandener Probleme hatte.
Weiter möchte ich Patrizia danken, die mir einen Großteil des Studiums zu Seite stand und immer für motiviert hat.
Abschließend danke ich Lara, die mich bei der Masterabreit unterstützt und sehr geholfen hat.

Stefan Knecht

II Inhaltsverzeichnis

III Abbildungsverzeichnis

IV Tabellenverzeichnis

V Abkürzungsverzeichnis

MUN	Modeling User Navigation
SNAP	Stanfort Netzwork Analyse Platform
DKE	Domain Knowledge Elements

1 Einleitung

Hintergrund dieser Arbeit ist es, die Navigation durch ein Informationsnetzwerk zu simulieren. Diese Navigation wird anhand unvollständiger Informationen durchgeführt. Für die Navigation wird hier ein neu entwickeltes Verfahren verwendet.

Simpel gesagt wird versucht, die Navigation eines Menschen durch zum Beispiel das Wikipedia-Netzwerk vorherzusagen bzw. zu simulieren. Um dies möglich zu machen, wird das Verhalten eins Menschen nachgeahmt, indem dessen Auswahlverhalten so gut wie möglich kopiert wird. Die Idee dahinter ist, dass jeder User, der sich durch solch ein Netzwerk arbeitet und bestimmte Artikel sucht bzw. liest, sich meistens in den selben Bereichen oder Kategorien bewegt. Anhand dieser Grundlage kann man nun eine Wissensverteilung über die Kategorien erstellen, die den User repräsentieren soll. Ein Beispiel dafür wäre ein User, der sich hauptsächlich in Bereichen der Informatik bewegt, also Artikel über Programmiersprachen, mathematische Verfahren, Datenbanken oder ähnliches liest. Dieser User hat in seiner Wissensverteilung ebenfalls hohe Werte in Bereichen wie Mathe oder Wissenschaft. Es genügt hier nicht, den Algorithmus nur mit einem Wissensmodell zu testen, also zum Beispiel nur mit dem oben beschriebenem User. Man benötigt hier also unterschiedliche Modelle. Dabei werden zum einen Modelle von Generalisten verwendet, die sich in allen Bereichen gleich gut auskennen, aber auch Modelle von Spezialisten, die sich in bestimmten Kategorien ausgeprägtes Wissen besitzen, aber dafür in anderen weniger. Mit diesen verschiedenen Modellen der User werden nun anhand der, aus den Userdaten kommenden Navigation, eigene erstellt.

Für diese Arbeit werden die Testdaten aus dem sogenannten "Wikigame" herangezogen, bei dem es sich um ein Spiel handelt, bei dem man einen Startartikel innerhalb des "Wikipedia for schools" angegeben bekommt und man in möglichst wenigen Schritten zu einem bestimmten Zielartikel gelangen soll. Es wird nun das Verhalten der Testuser aus diesem Spiel durch die oben beschriebene Theorie simuliert und getestet. Diese Simulationen werden anschließend mit den wirklichen Daten der User verglichen und die Resultate in Grafiken und Statistiken illustriert. Des Weiteren werden die Ergebnisse kritisch diskutiert und die zukünftige Arbeit in diesem Bereich angeschnitten.

Um nun zu testen, ob sich durch dieses Verfahren eine gute Navigation berechnen lässt, die dem des Menschen ähnelt, werden die Pfade der Testuser nacheinander durchgegangen und bei jedem Schritt überprüft, ob der Algorithmus den selben nächsten Schritt gehen würde, wie der User selbst. Führt man diese nun mit allen Pfaden und den unterschiedlichen Wissensmodellen durch, erhält man für jedes Modell eine Trefferquotenverteilung. Anhand dieser lässt sich sehr gut analysieren, wie effektiv und erfolgreich die jeweiligen Modelle sind.

1.1 Einführung in die Thematik

Im heutigen Zeitalter des Internets stellt sich folgenden Fragen: Ist es möglich, das Klickverhalten eines Menschen zu simulieren? Gibt es einen Algorithmus, der dies mit Zuhilfenahme bestimmter Parameter über den User ermöglicht? Um eine Lösung für dieses Problem zu finden, wurden verschiedene Methoden entwickelt. Eine davon wird in dieser Arbeit beschrieben.

1.2 Ziele der Arbeit

Ziel der Arbeit ist es, einen neuen Weg bei der Knotenselektion zu entwickeln, der dem des Menschen so nah wie möglich kommt.

Im Detail wird für diese Variante der Knotenselektion ein Wissensmodell angenommen, dass die Wissensbasis verschiedener Nutzer repräsentieren soll. In solch einem Wissensmodell werden die Bereiche oder Kategorien definiert in denen sich der "virtuelle" User sehr gut bzw. überhaupt nicht auskennt. Anhand dieser Definition wird nun entschieden, welche Knoten bei der Navigation durch ein Netzwerk als nächstes gewählt werden. Man geht davon aus, dass hier Knoten bzw. Artikel ausgewählt werden, die einer Kategorie angehören, in der man sich als User gut auskennt, um dort weiter zu navigieren. Hierfür werden verschiedene Wissensverteilungen der User angenommen und die Ergebnisse dieser miteinander verglichen.

Des weiteren wird gezeigt, wie nahe dieses Verfahren an den wirklichen Daten liegt, und, falls sie nicht repräsentativ sind, wie diese Methode verbessert werden kann und wo angesetzt werden muss.

Gezeigt werden die Resultate an Statistiken und Graphiken, die aus den Berechnungen des eigentlichen Algorithmus im Vergleich zu den Userdaten hervorgehen. Um dies vergleichen zu können, wird hier überprüft, wie hoch die Anzahl der gleichen Entscheidungen zwischen dem hier entwickeltem Algorithmus und wirklichen Testdaten aus dem Wikipedia for schools ist. Diese jeweilige Trefferquote wird für die verschiedenen Wissensmodelle berechnet und mit den restlichen Modellen und Algorithmen verglichen. Diese Ergebnisse werden anschaulich in Statistiken und Grafiken dargestellt.

Zusammenfassend beschreibt diese Arbeit die Idee, den Ablauf und die Ergebnisse des Algorithmus, aber auch mögliche Weiterentwicklungen dieses.

1.3 Struktur der Arbeit

Zu Beginn werden Grundlagen zur Graphentheorie gegeben und im Detail erläutert, welche Komponenten in solch einem Netzwerk enthalten sind und welche Arten es davon gibt. In den darauf folgenden Kapiteln wird die Suche und Navigation in diesen Strukturen dargestellt. Nachdem die Grundlagen wiedergegeben wurden, wird das Verfahren an einem Beispiel gezeigt und detaillierter beschrieben. Dazu gehören auch Ausschnitte aus dem Code und die Entwicklung diese Verfahrens. Darauf folgt die Illustrierung des Frameworks und dessen wichtigste Komponenten. Îm Anschluss folgen die Ergebnisse des Algorithmus, dessen Probleme und Statistiken zu den Resultaten. Den Schluss dieser Arbeit bildet eine Aussicht auf die zukünftigen Möglichkeiten, die man auf diesem Projekt aufbauen kann und verbessern kann.

2 Graphentheorie

In diesem Kapitel werden grundlegende Informationen zur Graphentheorie gegeben, dazu gehören im Detail die Komponenten solcher Graphen und weiterführend, welche Arten es davon gibt.

2.1 Einführung

Es wird hier genauer auf die Graphentheorie eingegangen, da unter anderem mit dem "Wikipedia for schools" Netzwerk gearbeitet wird und die Struktur eines solchen Netzwerkes mathematisch als Graph dargestellt werden kann.

Bei einem Graphen handelt es sich um eine unbestimmte Anzahl an Objekten (Knoten), zwischen denen untereinander Verbindungen bestehen.

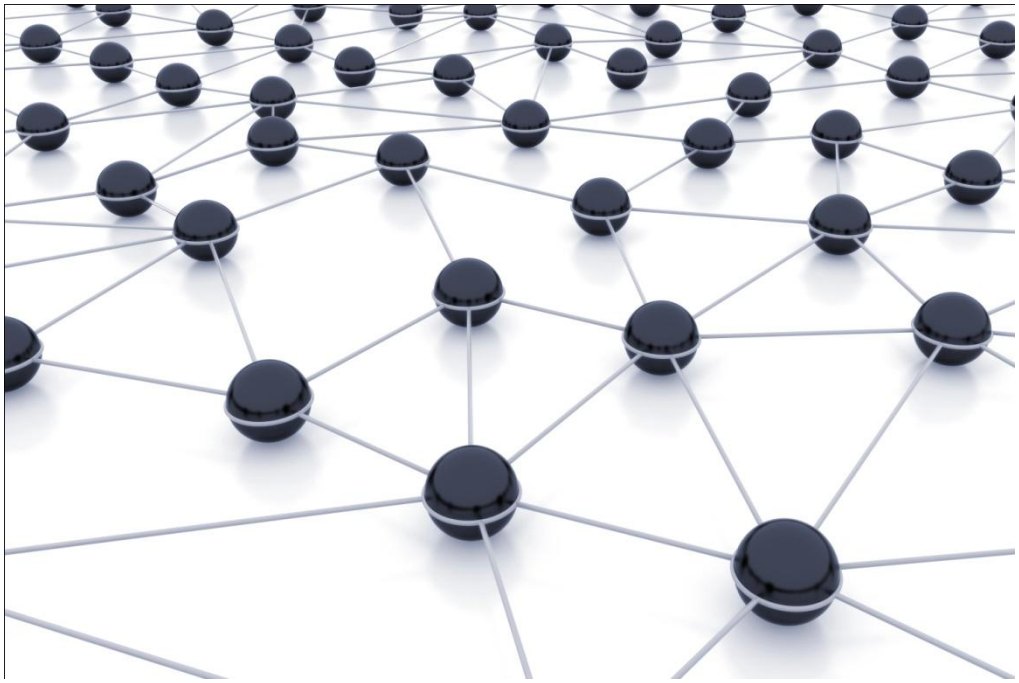


Abbildung 1: Graph [34]

2.2 Komponenten

In den folgenden Abschnitten werden die einzelnen Elemente eines Graphen beschrieben und erklärt.

2.2.1 Knoten und Kanten

Ein Graph besteht aus einem Set von Objekten, die Knoten genannten werden. Jeder dieser Knoten ist mit mindesten einem anderen Knoten verbunden und die jeweiligen Verbindungslinien nennt man Kanten. In Abbildung 2 werden 5 Knoten A, B, C, D, E dargestellt. A und E sind über eine Kante miteinander verbunden. Zwei miteinander in Verbindung stehende Knoten werden auch *Nachbarn* genannt. Abbildung 2 ist eine typische Darstellung eines Graphen.

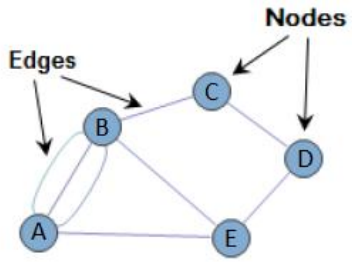


Abbildung 2: Knoten und Kanten

2.2.2 Pfade

Bei einem Pfad handelt es sich um eine Sequenz von Knoten, die durch Kanten miteinander verbunden sind. Alle Elemente innerhalb einer Reihe unterscheiden sich aber voneinander. Hier spricht man von einem *simplen* Pfad [1]. Sind einzelne Knoten mehrfach vorhanden, spricht man von einem *nicht simplen* Pfad [1]. Bei dem in Abbildung 3 gezeigten validierenden Graph wäre ein gültiger Pfad B -> E -> D -> C oder A -> E -> B, aber A -> E -> B -> A hingegen eine spezielle Art eines Pfades. Um was sich hier handelt, wird in einem späteren Abschnitt beschrieben.

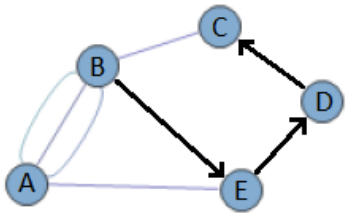


Abbildung 3: Pfade

Soviel zur grundlegenden Erklärung von Pfaden. Pfade sind ein sehr mächtiges Werkzeug innerhalb von Graphen und finden in vielen verschieden Bereichen ihren Einsatz. Unter anderem findet man ihre Verwendung in jeglichen Arten von Netzwerken, die später noch erläutert werden. Dazu gehören Informations- und Kommunikationsnetzwerke oder biologische und soziale Netzwerke. Des Weiteren sind sie in vielen Bereichen der Mathematik von großer Bedeutung.

2.2.3 Kreise

Ein besonders wichtiger Fall eines *nicht simplen* Pfades ist der sogenannte *Kreis*. Bei diesem speziellen Fall sind die Knoten und Kanten in einer Ringform angeordnet, wie es in Abbildung 4 zu sehen ist. Hier bilden B, E, D, C die eben erklärte Struktur.

Für einen Kreis benötig man mindestens drei Knoten, die miteinander verbunden sind, in welchem der erste und letzte Knoten dem gleichen entsprechen [1]. ´

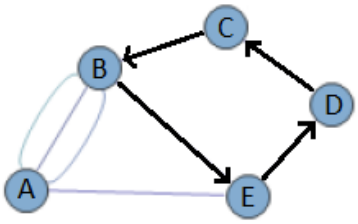


Abbildung 4: Kreis

Solche Kreise finden sich nicht nur in Graphen und den darauf aufbauenden Netzwerken wieder, sondern auch in Bereichen des wirklichen Lebens. Lernt man zum Beispiel eine neue

Person im Freundeskreis kennen, bei der sich im Verlauf des Gesprächs herausstellt, dass deren bester Freund ein früherer Schulkollege von einem ist, kann man hier auch von einem Kreis sprechen, der bei einem selbst geschlossen wird.

2.2.4 Konnektivität

Konnektivität ist ein weiterer wichtiger Aspekt in der Graphentheorie. Normalerweise ist bei einem gegebenen Graphen jeder Knoten über einen bestimmten Pfad mit jedem andern Knoten erreichbar. Ist diese Prämisse erfüllt, spricht man davon, dass dieser Graph *connected* oder *verbunden* ist. Im Beispiel aus Abbildung 5 ist dies nicht der Fall. Daher kann man hier auch nicht von einem *connected graph* sprechen [1].

Allgemein kann man davon ausgehen, dass Graphen zum Großteil zusammenhängend sind. Als Beispiel betrachte man ein Transport- oder Kommunikationsnetzwerk, bei dem zum Großteil das "connected" Kriterium erfüllt ist oder dieses zumindest angestrebt wird, da in solchen Bereichen Pakete oder Verbindungen zwischen allen Knoten möglich sein sollten. Außerdem kann es möglich sein, dass es in sozialen Netzwerken User gibt, für die es keinen Pfad zu jeder anderen Person innerhalb dieses Netzwerkes gibt.

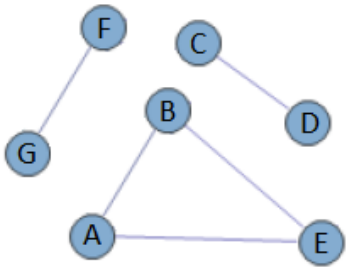


Abbildung 5: Konnektivität

Ein etwas komplexeres Beispiel ist in Abbildung 6 zu sehen. Es handelt sich hier um einen "Collaboration" Graph von einem biologischen Forschungszentrum, in dem einzelne Knoten Forscher darstellen. Zwischen diesen Knoten/Forschern besteht eine Kante, wenn beide gemeinsam an einem Paper gearbeitet haben, welches veröffentlicht wurde.[3] An diesem Graphen erkennt man sehr gut, dass es sich um einen nicht verbundenen Graphen handelt.

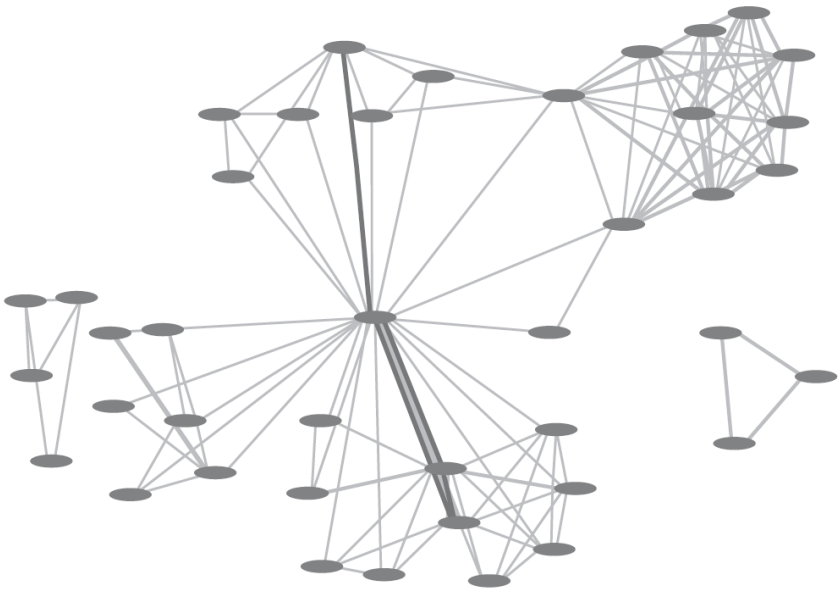


Abbildung 6: Graph mit enthaltenen Subgraphen

Ist ein Graph also nicht vollständig zusammenhängend, kann er in mehrere kleinere *connected* Graphen aufgeteilt sein. Also besitzt der in Abbildung 6 gezeigte Graph drei Subgraphen. Das Beispiel aus Abbildung 5 enthält ebenfalls drei Subgraphen (Im Detail bilden G und F, C und D sowie A, B und E einen solchen).

Diese werden auch als *connected component* bezeichnet. Solche Komponenten werden durch folgende Punkte definiert:

1. Jeder Knoten erreicht jeden anderen Knoten über einen Pfad.
2. Diese Untermenge darf nicht zu einer anderen größeren Menge gehören, in der jeder Knoten jeden anderen Knoten über einen Pfad erreichen kann.

Zusammenfassend ist zu sagen, dass solch ein *component* (Kurzform) intern mit dem Graphen verbunden ist und als solch einer definiert ist (1), aber keinerlei Kanten zu dieser Menge führen(2) [1].

2.2.5 Distanzen

Bei der Distanz spricht man von der Anzahl der Schritte, die benötigt werden, um von einem Startknoten zu einem Endknoten zu gelangen.

Stellt man sich jetzt nicht nur die Frage, welche Knoten denn über einen Pfad miteinander verbunden sind, sondern wie **lang** dieser Pfad denn ist, kommt man zu einer interessanten Problemstellung, welche in vielen Bereichen eine Rolle spielt. Daraus resultiert eine weitere Frage, die eigentlich die Wichtigere ist: Gibt es mehrere Pfade von dem selben Start- und Endknoten und welcher dieser Pfade ist der kürzeste Weg?

Diese Frage ist ein wichtiger Punkt im Transport-, Internet- und Kommunikationsbereich, da hier durch die kürzeren Distanzen Ressourcen gespart werden. Angenommen, ein Transportunternehmen würde die Strecke aus Abbildung 7 fahren, könnte stattdessen aber einfach direkt von Knoten B zu C „fahren“. Natürlich ist das ein sehr simples Beispiel, das aber zeigt, wie sich Distanzen auf die Pfadauswahl innerhalb eines Netzwerkes oder Graphen auswirken. In diesem Beispiel wäre das eine Ersparnis von 66% der benötigten Ressourcen.

Distanzen spielen im verwendeten Verfahren dieser Arbeit auch ein Rolle und werden später noch benötigt.

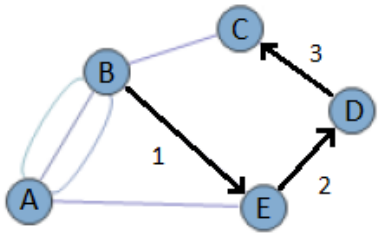


Abbildung 7: Distanzen

Ein weiterer Begriff innerhalb der Graphentheorie ist die *Gewichtung*, welche bei der Navigation in Netzwerken ein gutes Werkzeug ist, um reale Systeme bestmöglich als Graph darzustellen. Nimmt man wie oben an, dass man versucht, ein Transportunternehmen und dessen Routen als ein Netzwerk darzustellen, reicht es nicht aus, jede Strecke zwischen einem Knoten nur als "1" Weg anzusehen. Zusätzlich muss die Entfernung von jedem Knoten zu einem anderen berücksichtigt werden. So lässt sich ein solcher Graph sehr gut simulieren und für die Berechnungen der kürzesten Routen verwenden. Um dies an einem Beispiel zu illustrieren, nehmen wir an, dass man von dem, in Abbildung 8 gezeigten Knoten A, mit dem geringsten

Aufwand zu dem Knoten D gelangen möchte. Würde man hier die *Gewichtung* außen vor lassen, wäre der beste Weg von Knoten A über Knoten E zu Knoten D. Doch in Abhängigkeit der *Gewichtung* würde man auf dieser Strecke sieben, nennen wir es "Ressourcen", verbrauchen. Um den wirklich effizientesten Pfad zu nehmen, müsste die Route A, B, E, und D lauten, weil hier der "Ressourcenverbrauch" bei sechs liegt.

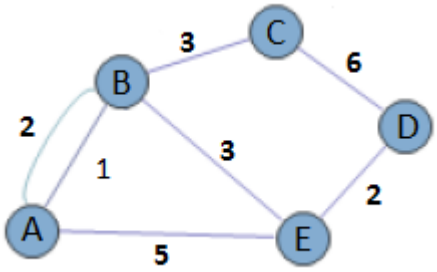


Abbildung 8: Gewichtungen

Natürlich spielt die *Gewichtung* nicht nur bei ungerichteten Graphen eine Rolle, sondern auch bei gerichteten. Diese werden im nächsten Kapitel erläutert.

2.2.6 Grad

Der Grad (oder die *Valenz*) eines Knotens bezeichnet die Anzahl der Nachbarknoten oder anders gesagt, die Kanten die der Knoten besitzt. Hat eine Ecke hingegen einen Grad von 0, also keine Nachbarn, so nennt man diesen Zustand „*isoliert*“.

Des weiteren werden *Minimal-* und *Maximalgrad* laut Diestel [5] folgendermaßen definiert:

"*Die Zahl $\delta(G) := min \{ d(v) \mid v \in V \}$ heißt Minimalgrad von G, und $\Delta(G) := max\{ d(v) \mid v \in V \}$ ist sein Maximalgrad.*"

Außerdem nennt man einen Graphen *regulär* oder *k-regulär*, wenn jeder Knoten der Graphen den gleichen Grad k besitzt [5].

2.3 Gerichtete und ungerichtete Graphen

Da nun die grundlegenden Elemente eines Graphen erklärt wurden, werden nun noch die Begriffe des gerichteten und ungerichteten Graphen beschrieben. Ebenfalls kann hier auch der Begriff der symmetrischen und unsymmetrischen Verbindung verwendet werden [1].

Man spricht von einem ungerichteten Graphen, wenn man innerhalb eines Graphen die Kanten in beide Richtungen "begehen" kann. Dies ist in allen vorherigen Beispielen (Abbildung 2, 3, 4, 5 und 7) der Fall, da hier alle Kanten keine bestimmte Richtung haben [5]. Verwendet man die oben erwähnte Ausdrucksweise, spricht man von "symmetrischen" Verbindungen.

Um das Ganze noch zu verdeutlichen, sieht man in Abbildung 8 einen gerichteten Graphen, was bedeutet, dass man nicht in jede Richtung von einem zum anderen Knoten gelangen kann [4]. Also könnte man zum Beispiel nur von Knoten A zu Knoten B gelangen, aber nicht umgekehrt. Dies gilt natürlich auch für Pfade und wirkt sich somit auch auf die Distanzen dieser aus, da hier nicht einfach der kürzeste Weg angenommen werden kann, sondern anhand der Übergangseinschränkung neu gesucht werden muss. Will man nun von Knoten A zu Knoten C gelangen, kann man nicht einfach über Knoten D navigieren, sondern muss sich hier über Knoten B bewegen. Hier kann der Begriff "unsymmetrisch" verwendet werden.

Wird es nicht explizit erwähnt, handelt es sich bei einem Graphen normalerweise um einen ungerichteten Graphen ohne Einschränkungen bei der Navigation durch die Knoten [1].

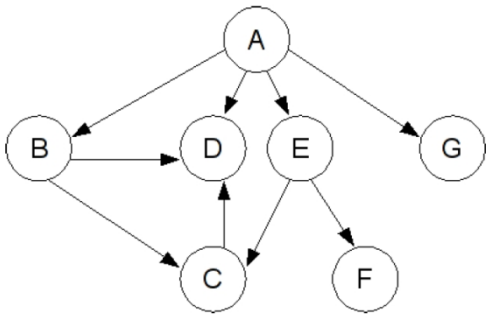


Abbildung 9: gerichteter Graph

Ein Punkt, der jetzt noch in diesem Abschnitt diskutiert wird, nämlich die *Gewichtung* von Kanten, wurde im vorherigen Kapitel "2.2.5 Distanzen" schon grundlegend für ungerichtete Graphen definiert und erklärt. Dies muss jetzt noch für gerichtete Graphen geschehen.

Bei gewichteten Graphen kann ebenfalls eine Gewichtung vorliegen, die in Kombination mit den gerichteten Kanten die Navigation innerhalb von Netzwerken erschwert. Am besten kann

man diese Problemstellung an einer Abbildung zeigen. Nehmen wird Abbildung 10 als Beispielgraphen und versuchen den kürzesten Pfad von Knoten A nach Knoten D zu finden. Der direkte Pfad nach D hätte eine Gewichtung von 7, was dem "längsten" Weg entspricht, wenn man sich alle anderen Pfade im Detail anschaut. Der Weg über B zu D hat zusammengerechnet eine Gewichtung von 6, was einer geringeren Anzahl entspricht. Doch es gibt noch zwei andere Pfade, einmal über B und C nach D oder über E und C nach D. Ersteres würde einem Aufwand von 5 (2 (A->B) + 2 (B->C) + 1(C->D)) entsprechen, Letzteres nur 4 (2 (A->E) + 1(E->C) + 1(C->D)).

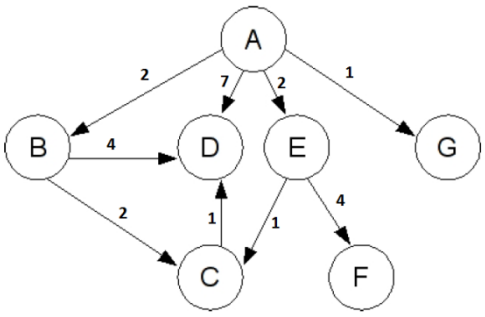


Abbildung 10: Gerichtete Graphen mit Gewichtung

Also ist abschließend zu sagen, dass es meist mehrere Pfade von einem Start- zu einem Endknoten gibt, man hier aber die Richtung und Gewichtung in solch einem gerichteten Graphen berücksichtigen muss.

2.4 "Small-World-Phänomen“

In diesem Abschnitt geht es um das sogenannte "Small-World-Phänomen“, welches ein Begriff aus einem von Stanley Milgram resultierenden Experiment ist. Es wird anhand des "Small-World-Phänomen" die verblüffend geringe Anzahl an Menschen beschrieben, über die jeder Mensch mit irgend einem anderen Menschen auf der Erde verbunden ist.

Wie in dem Kapitel "Kreise" erläutert, ist es jedem schon einmal passiert, dass man mit einer Person zum ersten Mal spricht und feststellt, dass man doch irgendwie über ein paar Ecken miteinander bekannt ist und man sich denkt: "Wie klein doch die Welt ist". Genau dieses Phänomen wurde zum ersten Mal in den 1960ern anhand von Studien versucht zu erklären und zu definieren [7].

Es gibt hier verschiedene Strategien, die dieses Problem der zufälligen Begegnung erklären. Man kann sich jetzt die Frage stellen: "Mit welchem *Grad* kennt jedes Mitglied einer Gruppe jedes andere Mitglied?". Das Wort *Grad* in dieser Frage spiegelt die durchschnittliche Anzahl der Personen wieder, über wie viele andere Menschen man die gesuchte Zielperson kennt. Es ist klar, dass nicht jede Person innerhalb dieser Gruppe jedes weitere Mitglied direkt kennt. Somit bilden sich Verbindungen über verschiedene Personen, die es ermöglichen, mit den anderen Mitgliedern auf eine andere Art und Weise verknüpft zu sein [6].

Nehmen wir an, es existiert für eine zufällige Person A zu jeder anderen zufälligen Person Z solch eine Verbindung, so ergibt sich: A-B-C-D...W-X-Y-Z. Daraus resultiert die Fragestellung: „Wie hoch ist die Wahrscheinlich unter der oben genannten Prämisse, dass 0,1,2,3,..,k die mindeste Anzahl von Verknüpfungsknoten ist, die man braucht um einen Pfad zwischen diesen beiden Individuen zu finden?“ Um diese Frage zu beantworten, benötigt man fundierte Daten, die diese Aussage belegen. An dieser Stelle kommt Stanley Milgram ins Spiel, der einer der Ersten war (1967), der sich mit diesem Thema beschäftig hat und angefangen hat, eine Studie über dieses Thema an der Harvard University durchzuführen. Diese Studie wurde zwei Mal durchgeführt. Einmal mit 60 Personen und beim zweiten Mal mit 296. Es wird hier Letzteres diskutiert.

Diese Studie hat er zusammen mit Jeffrey Travers entwickelt und geleitet. Bei dieser Studie ging es um eine Art Dokument, das von einer Gruppe aus 296 Startpersonen an eine bestimmte Zielperson in Boston geschickt werden sollte. Dieses Dokument, das versendet werden sollte, enthielt Informationen über den Ablauf der Studie, die Zielperson und eine Bitte, dieses Dokument an eine weitere Person weiterzusenden und somit an der Studie teilzunehmen. Des Weiteren sollte man sich in einer Tabelle (Alter, Geschlecht und Beruf) auf dem Dokument eintragen und eine Postkarte an den Professor der Universität schicken, damit der Verlauf der

einzelnen Dokumente leichter nachzuvollziehen ist. Voraussetzung für das Ganze war, dass das Dokument nicht direkt an die Zielperson gesendet werden durfte (dies durften nur Personen tun, die die Zielperson direkt kannten), sondern es nur an Menschen geschickt wurde, die sie persönlich kannten und bei denen die Wahrscheinlichkeit höher war, dass sie mit der Zielperson wiederum vertraut waren.

Wurde das Verfahren von jeder Person eingehalten, so hat das Dokument einen Pfad mit unbekannter Anzahl aus Bekanntschaften zur Zielperson erzeugt und hinter sich gebracht, falls es bei der Zielperson angekommen ist und nicht auf dem Weg von einer Person, die nicht an der Studie teilnehmen wollte, unterbrochen wurde [7].

Um herauszufinden, ob die Anzahl des Grades (also die Anzahl der Verbindungspersonen) etwas mit der Entfernung zur Zielperson zu tun hat oder ob man geschäftliche Verbindungen in dessen Richtung besitzt, wurde die Gruppe der 296 Startpersonen aus den folgenden Bevölkerungsgruppen zusammengestellt:

- eine zufällige Auswahl der Bewohner von Boston (n = 100)
- eine zufällige Auswahl der Bewohner von Nebraska (n = 96)
- eine zufällige Auswahl der Bewohner, die in Verbindung mit Nebraskas stehen (n = 100)

Von den 296 Personen haben tatsächlich 217 ihr Dokument weitergeschickt und es sind letztendlich 64 davon bei dem Professor angekommen. Die restlichen Dokumente, die nicht angekommen sind, wurden als "incomplete chains" eingeordnet, da sie wahrscheinlich irgendwo auf ihrem Weg nicht weiter geschickt oder einfach vergessen wurden [8].

In Abbildung 11 sieht man genau die 64 abgeschlossenen Pfade, deren Anzahl an Verbindungspersonen und welche Anzahl am häufigsten aufgetreten ist. Analysiert man diesen Graphen weiter, erkennt man, dass die häufigste Kettenlänge 6 beträgt. Berechnet man hier den Durchschnitt der Studie, resultiert daraus eine Pfadlänge von 5,2. Milgram und Travers zeigten nun mit ihrer Studie, dass man im Durchschnitt jeden Menschen innerhalb von Amerika über 5 Menschen erreicht und mit ihm verbunden ist.

Außerdem ist über das Resultat zu sagen, dass es sich um eine bimodale Verteilung handelt. Diese wurde noch im Detail untersucht und es zeigte sich, dass der zweite Hochpunkt, bei 6,1 dadurch entstand, dass die Dokumente mit Hilfe der Adresse zur Zielperson gelangten und sich die erste Spitze, bei 4,6 über den Beruf der Zielperson bildete.

Des Weiteren erreichten die Dokumente, die über die Adresse zur Zielperson gelangen sollten, Boston sehr schnell, doch verlief sich dort die Suche und sie verzögerte sich etwas. Bei

der joborientieren Suche hingegen gelangte das Dokument sehr schnell zur Zielperson, da hier noch das Wissen innerhalb der Branche herangezogen werden konnte.

Zusammenfassend ist zu sagen, dass Milgram's Studie ein Durchbruch in dem sogenannten "Small-World-Phänomen“ war und sie gezeigt hat, dass man über 5 Schritte von einem Menschen zu jedem anderen in Amerika gelangen kann.

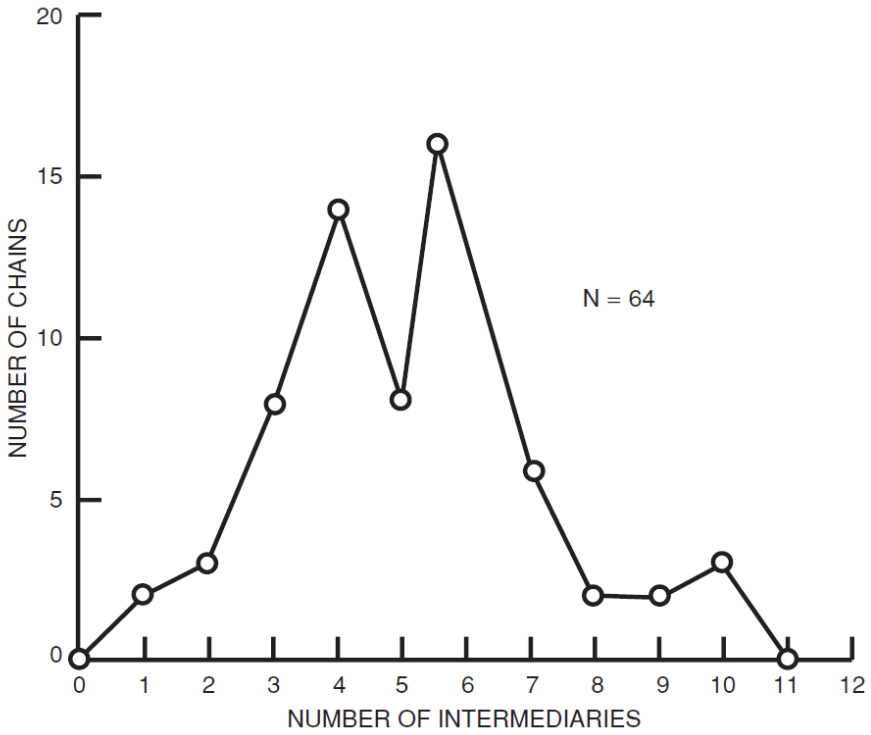


Abbildung 11: Zeigt die Länge des kompletten Kommunikationspfades (referenziert von *Travers and Milgram*, 1969)

2.5 Soziale Netzwerke

Soziale Netzwerke sind mit die bekanntesten Arten von Netzwerken, da alleine "Facebook" über eine Milliarden angemeldete Nutzer hat, was eine beeindruckende Zahl ist, da anders ausgedrückt jeder 7. der Weltbevölkerung bei "Facebook" angemeldet ist [6]. Andere soziale Netzwerke wären zum Beispiel "Twitter", "XING", "Google+", "LinkedIn". Zu den deutschen Vertretern gehören hier "StudiVZ", "SchülerVZ", "MeinVz" oder "werkenntwen".

Bei sozialen Netzwerken handelt es sich um Plattformen, bei denen man sich ein Netzwerk aus Freunden, Familie und sonstigen Bekannten aufbauen und mit diesen über eine Plattform kommunizieren kann. Des Weiteren dient solch eine Plattform als Repräsentation der eigenen Person im Internet, sowohl für private als auch für berufliche Zwecke. Ebenfalls kann man verschiedenen Gruppen beitreten (Informationsaustausch, Organisation, Gemeinsamkeiten) oder man teilt mit Freunden ausgewählte Informationen (Videos, Nachrichten, Bilder, etc.) [7].

Eben solch eine Plattform kann auch in einem Graphen dargestellt werden. Hier stellen die einzelnen Knoten die jeweiligen User dar. Diese User sind über Kanten mit anderen Usern verbunden, wobei eine Kante hier eine bestimmte Beziehung zwischen den Knoten darstellt. Diese Verbindung kann für unterschiedliche Sachen stehen, was meist von dem gewählten Netzwerk abhängt, zum Beispiel für eine freundschaftlich, beruflich oder Interessen basierende Verbindung. Zusätzlich kann man aus solchen Graphen weitere Informationen extrahieren, wenn man zum Beispiel die Position der Knoten, deren Knotengrad oder Kanten genauer betrachtet. So erhält man unter anderem Informationen über Größe des Freundeskreises oder Interessen der Personen [8].

In Abbildung 12 sieht man ein soziales Netzwerk eines Freundeskreises bestehend aus 34 Personen, die alle im selben Karate-Club sind. Analysiert man, wie oben erwähnt, die Höhe der jeweiligen Knotengrade, so lässt sich daraus schließen, dass wahrscheinlich der Knoten 1 und Knoten 34 die zentralen Punkte des Freundeskreises sind. Diese Knoten könnten möglicherweise auch die zwei Trainer des Clubs sein. Außerdem lässt sich sagen, dass der isolierte Knoten 12 zu den "Außenseitern" der Gruppe gehört, da er lediglich eine Verbindung zu der zentralen Person besitzt.

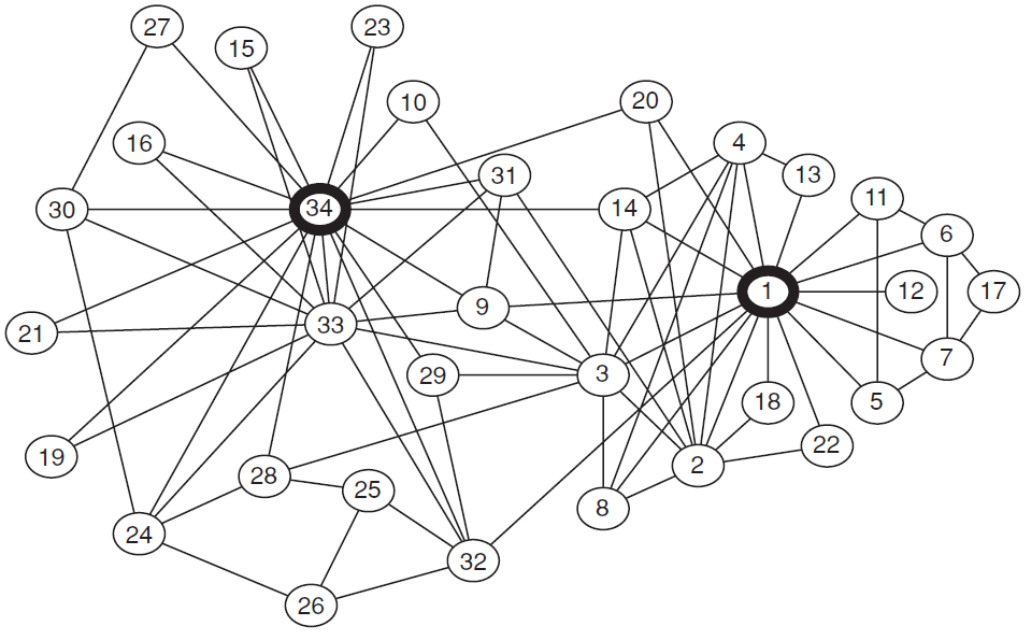


Abbildung 12: Ein soziales Netzwerk, das einen Freundeskreis eines Karate-Clubs enthält. Dieser Karate-Club besteht aus 34 Mitgliedern. (Grafik referenziert von *Journal of Anthropological Research* [9])

2.6 Informationsnetzwerk

Kommen wir nun zu den Informationsnetzwerken, bei denen es sich um eine weitere Art von Graphen handelt. Dieses Netzwerk wurde auch in dieser Arbeit als Datensatz verwendet: Das "Wikipedia for Schools" Netzwerk.

Bei einem Informationsnetzwerk handelt es sich um eine Menge von Internetseiten, Dokumenten oder den oben schon erwähnten Wikipedia-Artikeln, die miteinander in Verbindung stehen. Diese sind zum Beispiel über Hyperlinks, Referenzen, Zitate, Verweise oder Weiterleitungen miteinander verbunden. Wikipedia-Artikel stellen hier die Knoten in solch einem Graphen dar und die eben aufgezählten Verbindungen sind im Gegensatz zu den Knoten die Kanten [1].

Die Verbindungen können helfen zu verstehen, wie die Seiten/Artikel zueinander stehen, wie sie gruppiert sind oder welche von ihnen wichtig und beliebt sind. Ein Teil davon wird in Abbildung 13 gezeigt [1].

Bei solch einem Netzwerk treten meist, im Vergleich zu anderen Netzwerkarten, signifikant mehr Kanten auf, da die Daten untereinander viel stärker miteinander vernetzt sind. Je größer der Graph bzw. das Netzwerk ist, desto schwieriger wird es, solch einen Graphen zu illustrieren, da einfach zu viele Kanten vorhanden sind. Dies kann man sehr deutlich an dem im Ab-

bildung 13 gezeigtem polnischen Plog sehen. Versucht man nun, im Vergleich zum relativ kleinen Netzwerk des Plogs, ein Netzwerk wie die Wikipedia darzustellen, welches exponentiell größer ist, wird es nochmal wesentlich schwerer, eine übersichtliche Methode zu finden und diese zu repräsentieren [1].

Wenn man nun in Abbildung 13 noch einmal ins Detail geht, wird deutlich, dass es trotz dem oben beschriebenen Problem der Visualisierung möglich ist, diverse Aussagen über das Netzwerk zu treffen. Zum einen handelt es sich nicht um einen zusammenhängenden Graphen, da sich auf der linken Seite eine kleine Gruppe abgespalten hat und zum anderen ist sehr deutlich zu erkennen, dass es sich um zwei große Gruppierungen handelt. Dies ist an den zwei Ansammlungen der vielen Knoten links und rechts des Graphen zu sehen.

Es gibt zahlreiche Anwendungsmöglichkeiten für Informationsnetzwerke. Eine der bekanntesten Möglichkeiten, wenn nicht sogar die bekannteste ist Google. Google nutzt intensiv die Struktur eines solchen Netzwerkes, um die Relevanz und Qualität von Webseiten zu analysieren und zu bewerten.

Damit Google innerhalb von einer Sekunde die gewünschten Resultate liefern kann, werden sogenannte "Spiders" durch das Web geschickt, die sich über Hyperlinks von Seite zu Seite bewegen und so ein Netzwerk aus Internetseiten aufbauen. Dieses Netzwerk wird dann mit dem eingegebenen Suchwort gefiltert und jede Seite beispielsweise anhand von Prominenz, wie oft das Suchwort vorkommt, wo kommt es im Inhalt vor, wie oft wird diese Seite besucht, handelt es sich um eine seriöse Seite, u.v.m. durchsucht. Daraus resultiert ein Ranking für jede einzelne Seite, die dann abhängig von diesem Ranking in Google aufgelistet wird [10]. Dieses Thema wird jedoch in einem späteren Kapitel noch detaillierter betrachtet.

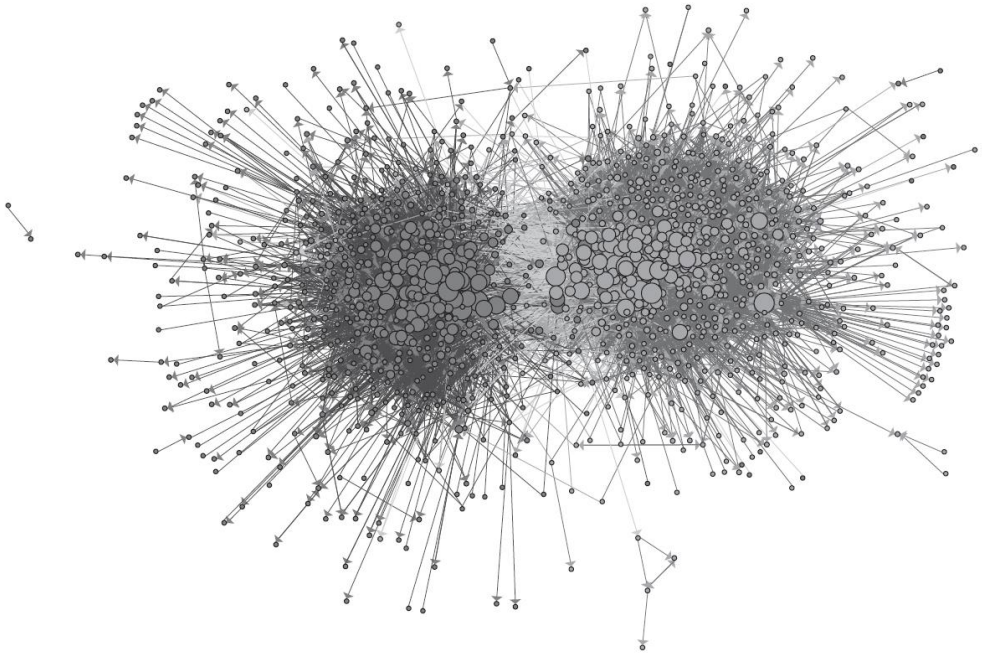


Abbildung 13: : Durch die Konstellation der Links zu den Internetseiten können bestimmte Gruppen und beliebte Seiten herausgelesen werden. In dieser Netzwerkstruktur eines politischen Blogs vor der U.S. Präsidentschaftswahl 2004 erkennt man deutlich die Separierung zweier Gruppen. (Grafik referenziert von *In Proceedings of the 3rd International Workshop on Link Discovery* [11]

2.7 Kommunikationsnetzwerk

Bei einem Kommunikationsnetzwerk handelt es sich um einen Graphen, in dem Computer oder Ähnliches durch Knoten repräsentiert werden. Zwischen diesen Knoten findet eine Kommunikation durch Nachrichten statt, die übertragen werden. Diese Kommunikation zwischen den einzelnen Computern wird durch die Kanten wiedergespiegelt.

Solch ein Netzwerk basiert auf Telefonie und ist kreisorientiert. Diese Netzwerke sind für die Sprachkommunikation entwickelt worden und verwenden Modems zur Umwandlung von digitalen Daten in für den Menschen hörbares Material [12].

In der Abbildung 14 sieht man die Netzwerkstruktur des sogenannten "Advanced Research Projects Agency Network", kurz geschrieben ARPANET aus den 70ern. Die einzelnen Knoten stellen die Städte dar, in denen die Großrechner stehen. Die Kanten hingegen zeigen die Kommunikationsmöglichkeiten zwischen den Großrechnern, also den Städten. Hier konnten somit Nachrichten zum Beispiel von Stanford nach Harvard geschickt werden.

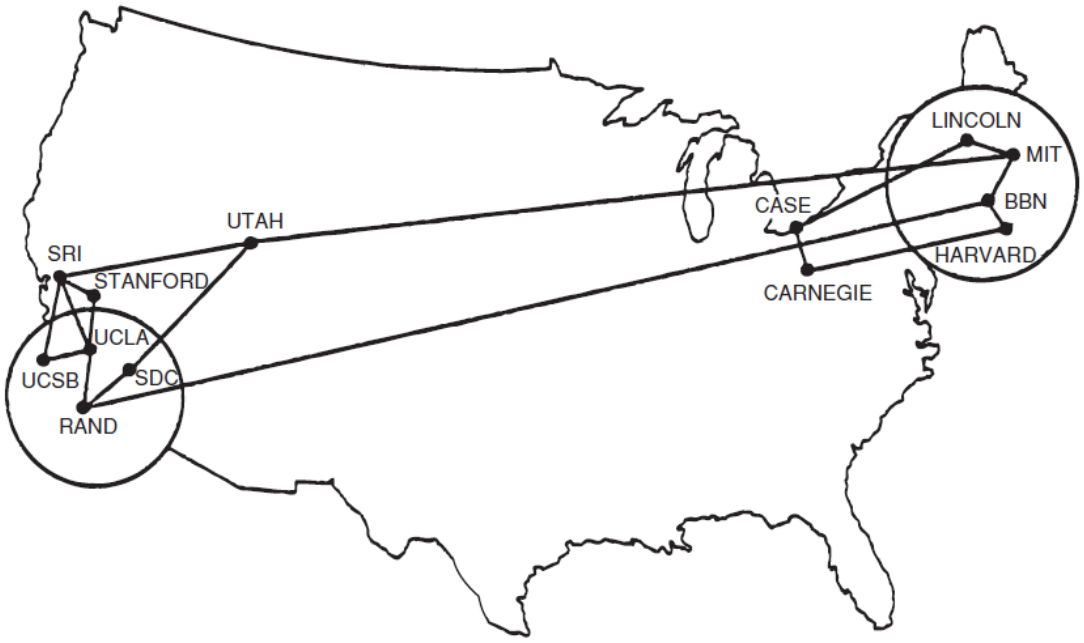


Abbildung 14: Darstellung eines Netzwerkes der Seiten im Internet vom Dezember 1970, welches als ARPANET bekannt war. (Grafik referenziert von ARPANET Completion Report [13])

D. Easley [1] zeigt in *Networks, Crowds and Markets: Reasoning About a Highly Connected World* eine wesentlich übersichtlichere und einem Graphen ähnlichere Darstellung, die in Abbildung 15 zu sehen ist. Er nimmt hier nur die 13 Knoten und 17 Kanten für die Grafik und stellt diese in einer objektiv anschaulichen Positionierung dar.

Der Graph ist hier also nicht von seiner Darstellung abhängig. Anhand dieser Illustration können die wichtigen Informationen, die bei solch einem Graphen eine Rolle spielen, deutlicher dargestellt werden. Dazu gehört zum einen, wie die Knoten miteinander verbunden sind und somit, welcher Server mit welchen Städten kommunizieren kann, zum anderen kann man so die Eigenschaften eines Graphen besser analysieren. In Abbildung 15 sieht man zum Beispiel sehr deutlich, dass hier mindestens zwei Kreise vorhanden sind.

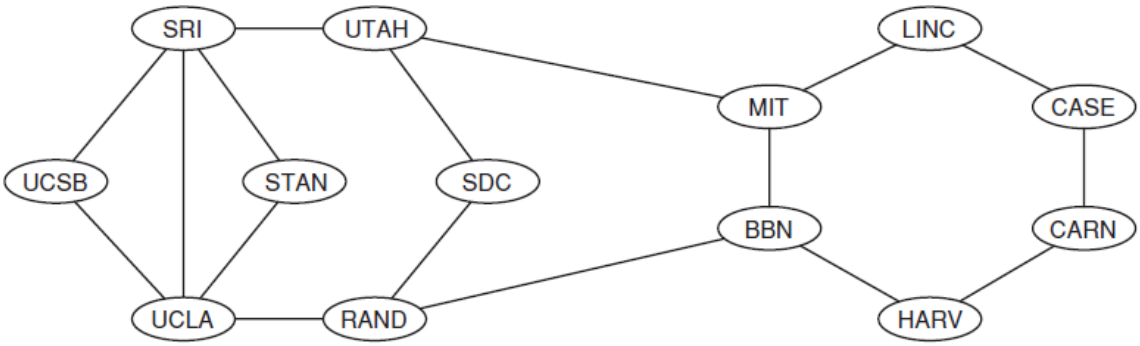


Abbildung 15: Alternative Darstellung der Internetgraphen aus dem Jahre 1970 mit den enthaltenen 13 Knoten [1]

2.8 Biologisches Netzwerk

Bei biologischen Netzwerken gibt es eine Vielzahl von Arten, die Knoten und Kanten zu definieren. In der Biologie ist die Netzwerkdarstellung ein mächtiges Werkzeug, mit dem unter anderem Metabolite, Gene oder Proteine illustriert werden können. Diese können nicht nur direkt gezeigt, sondern auch in Graphen eingebettet werden.

Das soll heißen, dass es bei biologischen Netzwerken nicht nur eine Ebene gibt, sondern man in den einzelnen Knoten noch weitere Netzwerke integrieren kann. Die Abbildung 16 zeigt dies sehr anschaulich anhand der drei Ebenen, die jeweils eine andere Art von biologischem Netzwerk symbolisiert. Das oberstes Level ist hier ein sogenanntes evolutionäres Netzwerk, in dem "*die evolutionären Abhängigkeiten zwischen Organismen dargestellt werden*" [14]. Die zweite Stufe beschreibt "*interzelluläre Signal-Netzwerke, bei denen Interaktionen zwischen Zellen beschrieben werden*" [14]. Bei der tiefsten und detailliertesten Ebene wird eine Kombination aus Genen, Transkripen, Proteinen und Metaboliten dargestellt, die untereinander über Kanten verbunden sind. Diese Kanten stellen nicht alle die gleiche Interaktionsart dar, sondern sind unterschiedlich definiert. So können zum Beispiel "*Proteine miteinander interagieren, Proteine die Aktivität von Genen regulieren, Metabolte die Aktivität von Proteinen beeinflussen usw.*" [14]

Dieses Beispiel zeigt sehr deutlich, was Graphen zu einem sehr mächtigen Werkzeug macht und wie vielfältig die Anwendungsmöglichkeiten dieser sind. Man kann nicht nur Systeme in vielen Bereichen darüber darstellen, sondern diese auch ineinander verschachteln.

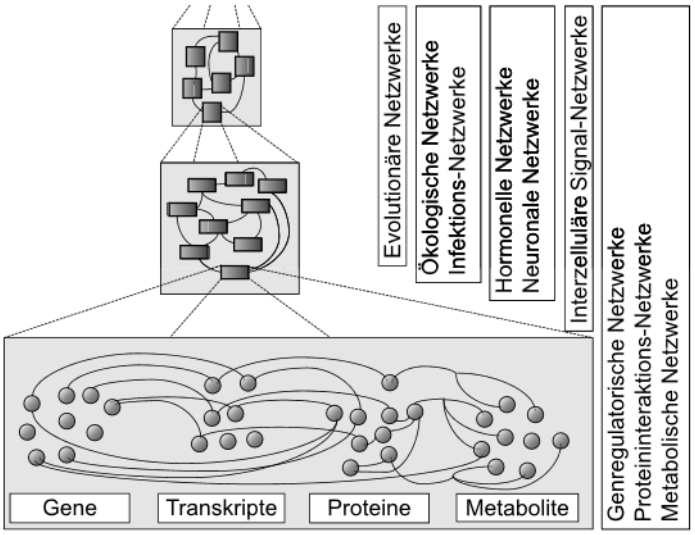


Abbildung 16: Verschiedene Ebenen eines biologischen Netzwerkes. (Grafik referenziert von *Analyse und Visualisierung biologischer Netzwerke* [14])

Bei den vorherigen Netzwerkarten handelte es sich immer um einen ungerichteten Graphen, da es nie eine Einschränkung bezüglich der Übertragungsrichtung von Nachrichten oder Verbindungen zwischen Onlineartikeln gab.

Bei biologischen Netzwerken kann es hingegen Graphen geben, bei denen es sich um gerichtete Systeme handelt. Dazu gehören zum Beispiel Proteininteraktions-Netzwerke und Strukturgraphen von Molekülen [14]. Eine weitere Variante sieht man in Abbildung 17, bei der es sich um die genetische Verwandtschaft eines Kaktusses handelt. Hier wird allerdings nur ein geringfügiger Teil der gesamten Genetik dargestellt.

Die Darstellung spielt hier ebenfalls eine große Rolle und führt auch zu diversen Problemstellungen, da sie zum einen ebenfalls sehr groß und komplex werden können, aber es auch keine einheitliche Darstellung gibt, sondern unzählig viele. Dadurch wird es schwierig sich in jede einzelne Variante neu hineinzuversetzen und diese richtig zu interpretieren. So können leicht Fehlanalysen resultieren [14]. Seit Kurzem gibt es hier in "*Systems biology graphical notation: Process diagram*" eine erste Spezifikation einer genormten Darstellung für solch eine allgemeine Illustrierung [15].

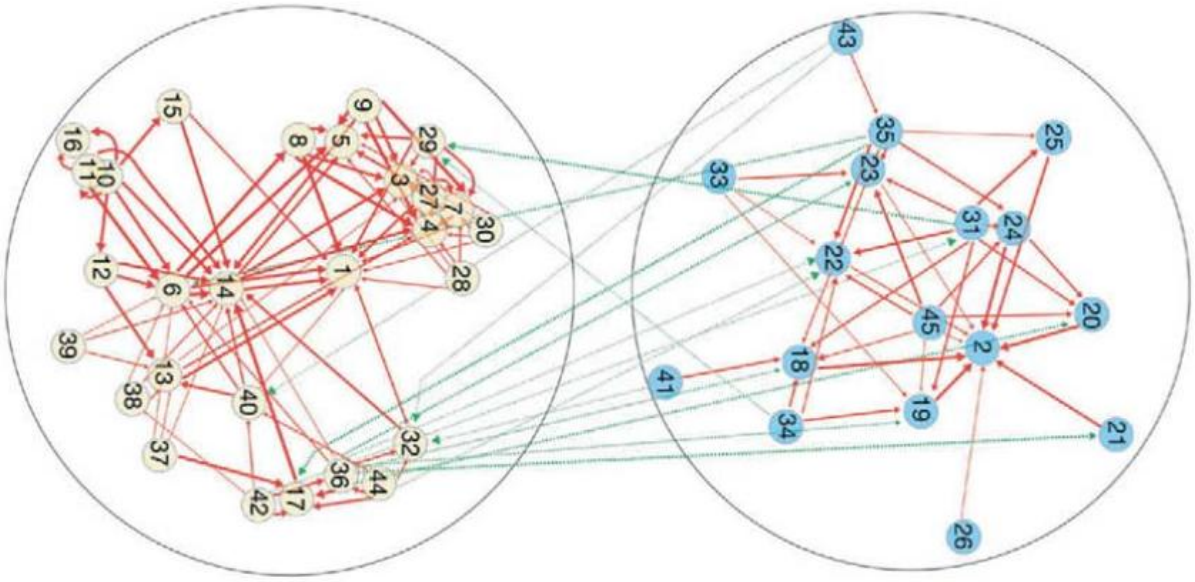


Abbildung 17: Die genetische Verwandtschaft der Population eines "cactus Lophocereus schottii". In diesem Diagramm repräsentieren die Kanten einen Bruchteil der gesamten genetischen Variation (Grafik referenziert von *Population graphs: the graph theoretic shapeof genetic structure* [16] .

3 Suche, Navigation und Erweiterungen in Netzwerken

In diese Kapitel geht es um die Suche in Netzwerken, Navigation im Internet und dortige Erweiterungen, wie bestimmte Hintergrundinformationen gesammelt und gefiltert werden. Es werden Algorithmen und bestimmte Verfahren erläutert, die essenziell für eine effektive Suche in solchen Graphen sind. Dazu gehören zum einen der "Breadth-First Search" und zum anderen das Pageranking Verfahren für die Suche im Internet.

3.1 Einleitung

Suche in Netzwerken ist heutzutage ein großes und wichtiges Thema in der Informationswelt, in der man lebt. Alles muss immer schneller gehen und Informationen müssen sofort erreichbar oder Pakete und Daten schnellstmöglich vor Ort sein.

Deshalb ist die effektive Suche in Netzwerken so wichtig. Ob es sich nun um ein Suchergebnis bei Google oder andere Suchen in Graphen handelt, es muss so schnell wie möglich ein Ergebnis geliefert werden. Gerade bei der Suchmaschine Google ist es faszinierend, wie schnell das Netzwerk "Internet" komplett durchsucht wird und innerhalb von Millisekunden ein passendes und gewünschtes Ergebnis geliefert wird.

Um so interessanter ist es, welches Verfahren und genaue Struktur hinter diesem effizienten und präzisen Resultat der Suchmaschinen steckt, welche Algorithmen, Analyseverfahren und Vorarbeit.

Jede dieser Suchmaschinen verwendet für ihre Resultate unterschiedliche Algorithmen und bestimmte Verfahren, die im Detail nicht nach außen dringen, aber man das grundlegende Verfahren erläutern kann.

3.2 Breadth-First Search

Es wurde in dem Kapitel "Distanzen" schon beschrieben, dass die Entfernung zwischen zwei Knoten innerhalb eines Netzwerkes sehr wichtig ist. Für einen kleinen Graphen ist die Berechnung der Länge eines Pfades sehr trivial und kann für den Menschen durch reines "hinschauen" berechnet werden, doch gilt das für größere und komplexere Graphen nicht und es muss ein systematisches Verfahren verwendet werden.

Dies nennt man "Breadth-First Search" und funktioniert ähnlich wie das Verfahren, das man wahrscheinlich auch verwenden würde, wenn man selbst in solch einem Netzwerk suchen würde. In Abbildung 18 ist dieses System sehr anschaulich dargestellt. Dabei handelt es sich um ein Soziales Netzwerk, in dem die Pfadlänge von einem selbst zu anderen Personen berechnet wird [1]:

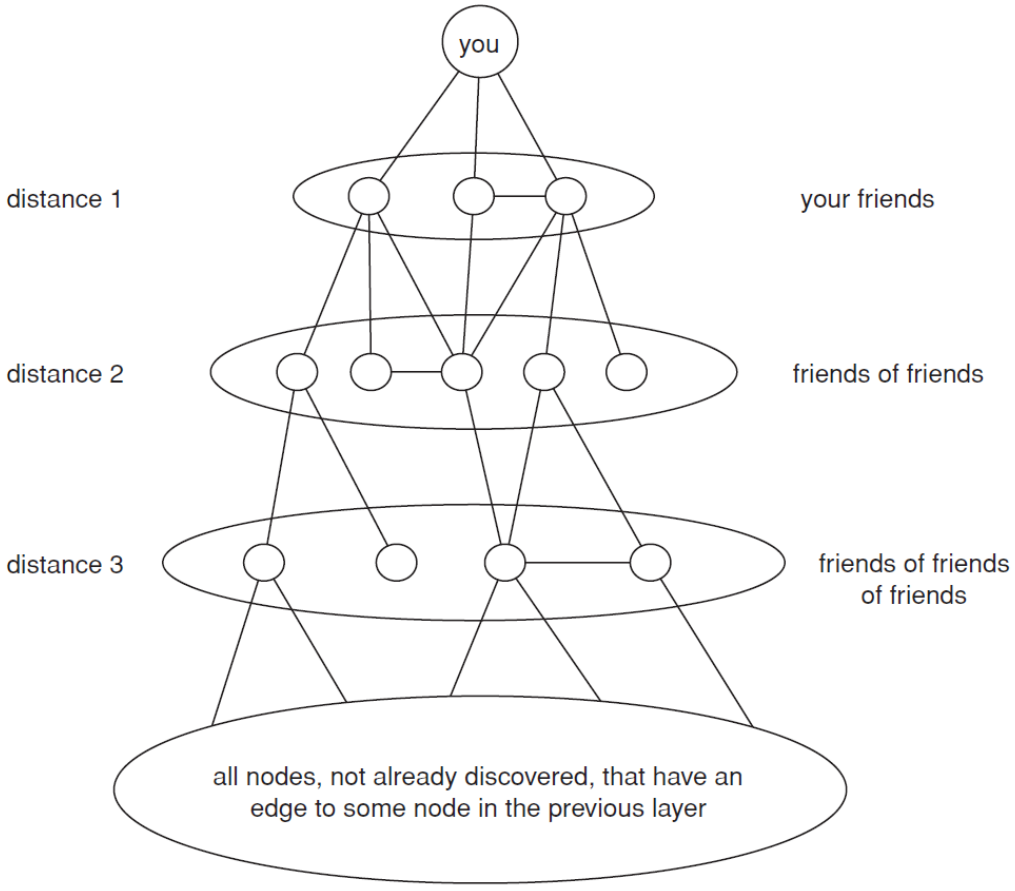


Abbildung 18: Zeigt das die "Breadth-First" Suche, welche die Entfernungen von dem Startknoten zu jeder weiteren Ebene darstellt. Jede Ebene besteht aus der Ansammlung der Knoten, die die Distanz der Ebene besitzen. (Referenziert aus *Networks, Crowds, and Markets: Reasoning About a Highly Connected World* [1])

1. Als erstes werden alle Nachbarknoten, in diesem Fall alle tatsächlichen Freunde, mit der Distanz *1* markiert.
2. Finde alle Freunde deiner Freunde (natürlich beinhaltet diese Liste nicht die Freunde, die du selbst schon in deiner direkten Freundesliste hast) und markiere diese Knoten mit der Distanz *2*.
3. Im dritten Schritt werden alle Freunde der Freunde deiner Freunde gesucht (hier gilt wieder das selbe Kriterium wie bei Schritt 2 und zwar werden bereits existierende Knoten nicht mehr hinzugefügt) und diese mit der Entfernung *3* markiert. (Dies entspricht der 3. Ebene in der Abbildung 18)
4. (4,...,k). Die Suche wird auf diese Weise weiter durchgeführt und für jede weitere neue "Ebene" werden neue Knoten gefunden, denen dann die Distanz der Ebene zugeteilt wird, *4*,...,*k*. Jede dieser neuen Ebenen enthält alle Knoten, die folgende Kriterien erfüllen:
    - Wurde bis jetzt noch nicht als Freund entdeckt und in die Liste hinzugefügt
    - Besitzt eine Verbindung zu einem Knoten in der vorherigen Ebene

Dieser Algorithmus wurde hier in dem Beispiel für ein soziales Netzwerk verwendet. Das heißt aber nicht, dass es nur hier verwendet werden kann, sondern es natürlich auf jegliche Art von Graphen anwendbar ist und man von einem festgelegten Startknoten aus, die „Schicht für Schicht“ Methode durchlaufen lassen kann, wodurch sich jede neue Ebene aus noch nicht erforschten Nachbarknoten zusammenstellt [1].

Um dies noch etwas zu verdeutlichen, nehmen wir uns das schon aus den vorherigen Kapiteln bekannte ARPNET zu Hilfe (Abbildung 14) und zeigen, wie sich die "Breadth-First" Suche auf dieses Informationsnetzwerk anwenden lässt.

Der Startkonten ist hier das MIT in Massachusetts, von wo aus alle Nachbarknoten mit der Distanz *1* definiert werden und in der nächsten "Ebene" gespeichert werden. Diese Ebene enthält nun die Großrechner von UTAH, BBN und LING. Ausgehend von dieser Ebene werden nun wieder all deren Nachbarn zu einer weiteren Schicht hinzugefügt. Diese Ebene nennt man nun vereinfacht "Ebene *2*". Hinzugefügt wurden SRI, SDC, RAND, HARV und CASE. Das MIT gehört hier nicht dazu, da dieser Knoten bereits besucht wurde und deshalb nicht erneut mit in eine andere Ebene hinzugefügt werden darf. Ebene *2* entspricht der Distanz von *2*.

Betrachtet man nun als letzten Schritt alle Nachbarknoten von Ebene *2* (UTAH, BBN, LINC, UCSB, STAN, UCLA und CARN), werden zur neuen Schicht mit der Distanz *3* folgende Knoten hinzugefügt: UCSB, STAN, UCLA und CARN. Die fehlenden Nachbarknoten wer-

den, wie in der vorherigen Ebene das MIT, nicht hinzugefügt, da sie bereits verwendet wurden und sie somit schon in diesem Spannbaum vorhanden sind.

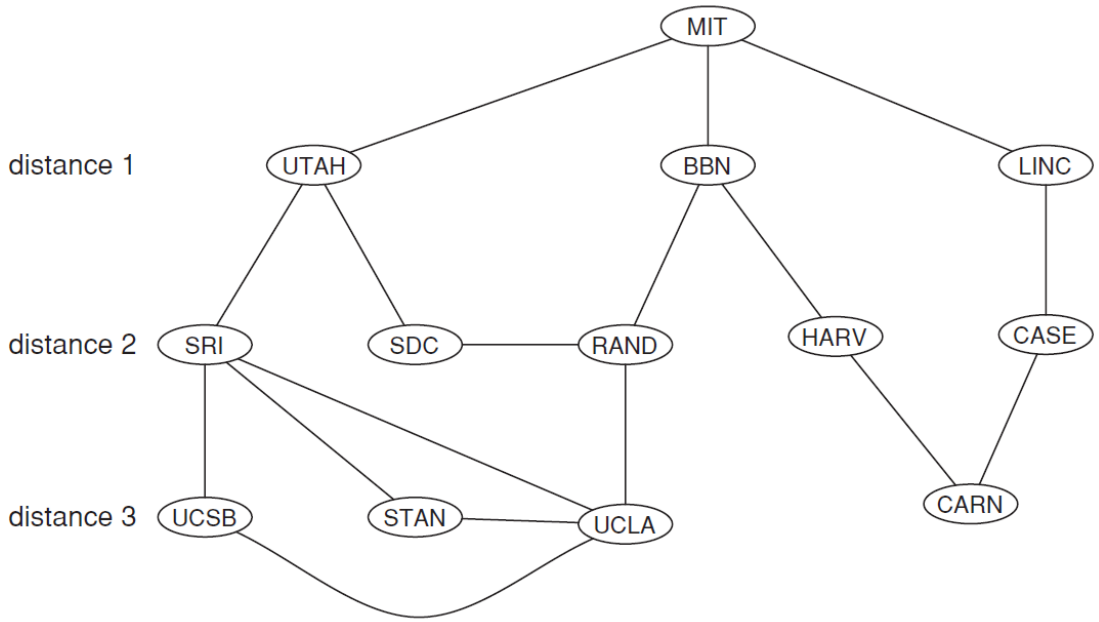


Abbildung 19: Ergebnis der " Breadth-First" Suche innerhalb des ARPANET Netzwerkes von 1970. Startknoten ist hier das MIT in Massachusetts (Referenziert aus *Networks, Crowds and Markets: Reasoning About a Highly Connected World* [1])

Abschließend ist zu sagen, dass dieser Algorithmus so benannt wurde, da er ausgehend von einem Startknoten die nahesten Knoten zuerst erreicht. Es kann damit nicht nur die Distanz zu bestimmten Knoten berechnet werden, sondern auch die Struktur eines Graphen erfasst werden (ausgehend von einem bestimmten Startknoten).

3.3 Suche im Web

In diesem Kapitel geht es um die Suche im Web, welche Techniken es hierbei gibt, um das beste Ergebnis innerhalb von Sekunden zu erhalten und über Probleme, die dabei auftreten. Dieses Thema wurde in einem vorherigen Abschnitt kurz angesprochen und wird hier jetzt ausführlich erläutert.

Egal, ob man bei Google einen Begriff mit passender Erklärung sucht oder einen Onlineshop für eine bestimmte Ware, die Suchmaschine findet immer eine zutreffende, weiterhelfende Seite. Doch woher weiß eine Internetseite, was ich genau möchte und was dafür die beste Seite im ganzen World-Wide-Web ist? Nimmt man als Beispiel den Suchbegriff "Karl Franzen", erhält man als Resultat die Homepage der Karl-Franzens-Universität Graz (http://www.uni-graz.at/). Um auf solch ein Ergebnis zu kommen, kann es nicht nur reichen, Internetseiten nur nach ihrem Namen zu filtern. Es müssen weitere Informationen gesammelt werden, die eine fundierte Auflistung der zur Verfügung stehenden Resultate der Suche zur Folge hat. Diese Informationen werden innerhalb der nächsten Kapitel nicht nur definiert und erklärt, sondern wird auch erläutert, wie man diese analysiert und verarbeitet.

3.3.1 Probleme bei der Websuche

Bei solch einer Suche treten jede Menge Probleme auf, die die Suche erschweren. Eines der Probleme wurde schon im sogenannten *Information Retrieval* in Zeiten vor dem Internet definiert und versucht dieses bestmöglich zu lösen. Es handelt sich hierbei um das effektive Durchsuchen von bereits vorhandenen, größeren Datenmengen. Es ist hier sehr schwer, anhand von nur wenigen bis einzelnen Suchwörtern eine hochwertige Antwort innerhalb von diesen Daten zu finden, da einfach nicht ausreichend Suchmaterial angegeben wird. Sucht man zum Beispiel nach "Puma", gibt es mehrere Möglichkeiten dies zu interpretieren. Sucht der User hier nach dem Tier, will er Schuhe oder ähnliches kaufen oder sucht er eventuell nach Informationen über die Firma. Als wäre das Durchsuchen solcher Mengen nicht schon problematisch genug, treten hier noch weitere Komplikationen auf, nämlich durch *Synonyme* oder der *Polysemie*. Bei Synonymen handelt es sich um die unterschiedlichen Wortmöglichkeiten für ein Wort (zum Bespiel kann man für Auto auch Fahrzeug, Wagen oder Automobil sagen). Gibt es für ein Wort mehrere Bedeutungen (Ente: Das Tier oder das Auto) spricht man von Polysemie.

Es sollte auch die Qualität des Inhaltes von resultierenden Berichten oder Homepages bestimmt werden, was wiederum zu der Frage führt, wie man nun unterscheidet, ob der Text oder gar die Homepage professionell geschrieben oder entwickelt und erstellt wurde. Selbst wenn der Inhalt verständlich und in Hochdeutsch geschrieben ist, heißt das noch lange nicht, dass der Inhalt genau das widerspiegelt, was man sucht, und ob der Inhalt qualitativ hochwertig ist [1].

3.3.2 Hubs und Authorities

Nachdem die grundlegenden Schwierigkeiten erläutert wurden, werden nun wichtige Punkte erläutert, die es möglich machen, die einzelnen Homepages zu bewerten und somit die Resultate der Suche wesentlich zu verbessern.

- *Bewertung durch "In-Links":* Bei dem oben genannten Beispiel des Wortes "Karl Franzen" gibt es sicherlich eine große Anzahl von Internetseiten, in denen dieses Wort vorkommt, aber das Topresultat sicher nicht die höchste Anzahl des gesuchten Wortes beinhaltet. Im Gegensatz dazu gibt es jedoch viele Verlinkungen von anderen Seiten auf diese. Also kann man davon ausgehen, dass die sogenannten "In-Links" eine Rolle bei der Bewertung spielen. Als Beispiel nimmt man eine Menge von Internetseiten her, die alle das Suchwort enthalten. Nun überprüft man, welche dieser Seiten die meisten *In-Links* durch andere Seiten dieser Menge erhalten und hat einen ersten Indikator für eine "Top-Ranked" Seite [1].

- *"List-Finding" Technik:* Analysiert man das oben erklärte Bewertungsverfahren, lässt sich daraus schließen, dass die Internetseiten, die auf Homepages verlinken, sich in diesem Bereich gut "auskennen" und man dies als Informationsquelle ausnutzen kann. Man weist also diesen zentralen Seiten eine Liste zu, in denen alle verlinkten Seiten enthalten sind. Es handelt sich hier natürlich nur um zum Thema relevante Links. Diese Liste geht man nur durch und liest bei jeder verlinkten Seite die dazugehörigen "Votes" (*In-Links)* aus und summiert diese Werte. So erhält man für jede dieser zentralen Seiten einen Wert, der aussagt, wie hochwertig die Verlinkungen dieser im Allgemeinen sind.

Um dieses Verfahren noch etwas zu verdeutlichen, sieht man in Abbildung 20 die Anwendung von beiden oben erklärten Verfahren mit dem Suchwort "Newspaper" in den USA. In den größeren Blasen sieht man die möglichen Resultate der Suche. Beschriftet sind diese mit der Anzahl an *In-Links* von anderen Seiten, die das "Voting" darstellen. Die New York Times besitzt auf anderen Seiten zum Beispiel insgesamt 4 *In-Links*. So erhält jede dieser Ergebnisse

ein gewisses Voting. Kombiniert man dies nun mit dem *List-Finding*, welches durch Zahlen in den kleineren Blasen dargestellt wird, erhält man ein noch besseres Verhältnis [1].

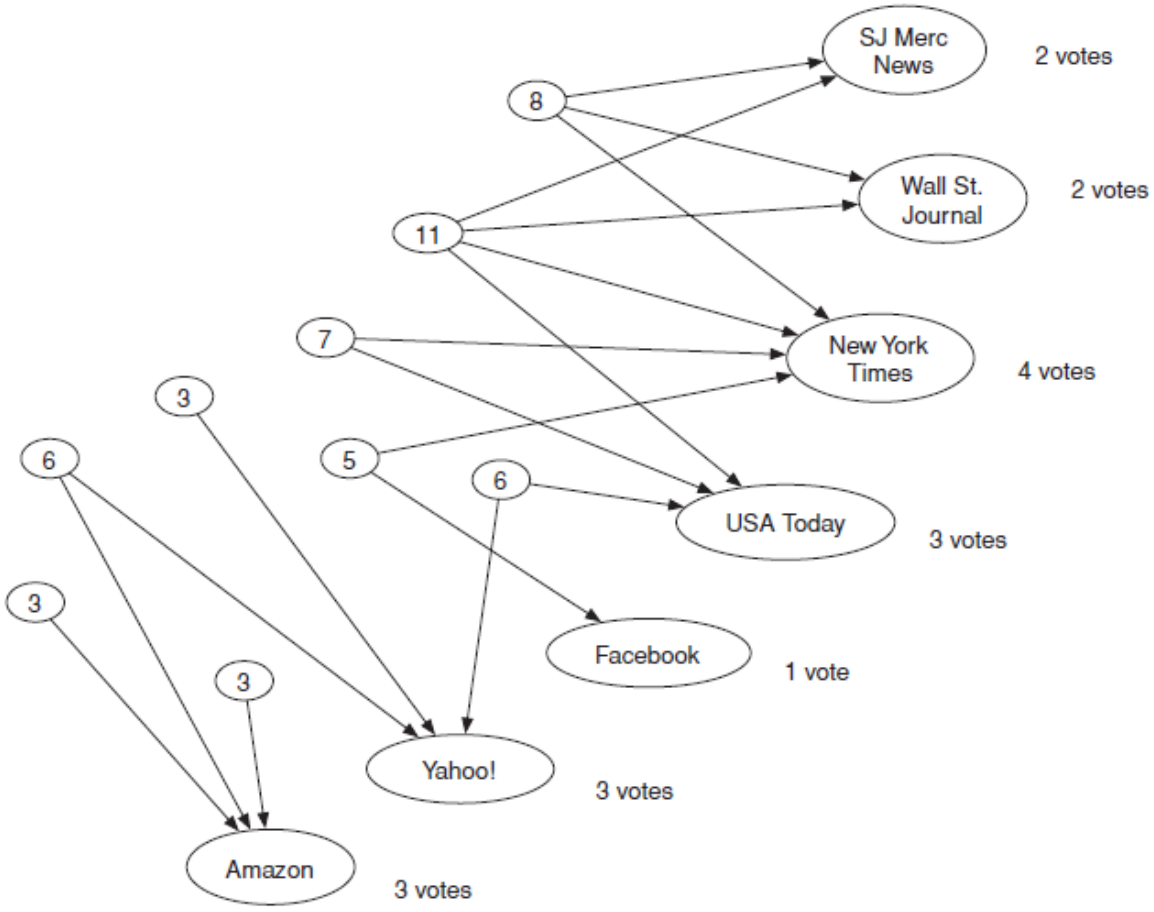


Abbildung 20: Votes und Listing-Value der Suchanfrage "Zeitung" (Referenziert aus *Networks, Crowds and Markets: Reasoning About a Highly Connected World* [1])

Dieser Wert berechnet sich aus der Summierung der Votes der verlinkten Seiten. Zum Beispiel ergibt sich also bei der obersten zentralen Seite ein Wert von 8, welcher aus SJ Merc News (2 Votes), Wall St. Journal (2 Votes) und New York Times (4 Votes) zusammengesetzt wird. Nun lassen sich durch Kombination der beiden Werte die hochwertigeren *In-Links* herauslesen (im Beispiel wären die *In-Links* der kleineren Blase mit dem Wert 11 am qualitativsten). Daraus lassen sich dann die besseren Suchergebnisse filtern. Dieses Beispiel ist natürlich nur eine Miniaturdarstellung, zeigt aber das grundlegende Verfahren.

Diese zwei Techniken lassen sich nun noch verfeinern. Dafür definieren wir zum einen die Art von Internetseiten (Hohes Ranking), die wir eigentlich suchen, als sogenannte *Authorities*. Des Weiteren werden die hochwertigen Listen (*List-Finding*) als *hub* definiert (Werte aus den

kleineren Blasen). Jede der Seiten, die analysiert werden, erhalten einen *Authority* und einen *Hub* Wert. Der Sinn hinter diesen Werten liegt darin, dass man diese Werte nun nacheinander updated und sich die besseren Seiten noch mehr herauskristallisieren.

Im Detail funktioniert das folgendermaßen [17]:

- Update von *Authorities*: Summiere alle *hub* Werte, die auf diese Seite zeigen
- Update von *Hubs*: Summiere alle *Authority* Werte von den Seiten, auf die diese Seite zeigt
- Wiederhole diese Schritte *k* mal (*k* ist frei wählbar)
- Normalisieren der Ergebnisse (da die Resultate sehr groß werden können)

Wendet man dieses erweiterte Verfahren auf das Beispiel von Abbildung 20 an, resultieren daraus die Werte in Abbildung 21. Es lassen sich nun genaue Aussagen über das Ranking der Seiten machen. Seiten wie zum Beispiel Amazon, Yahoo! und Facebook fallen komplett aus der Suche heraus und wichtige Ergebnisse wie die New York Times werden hoch gewertet.

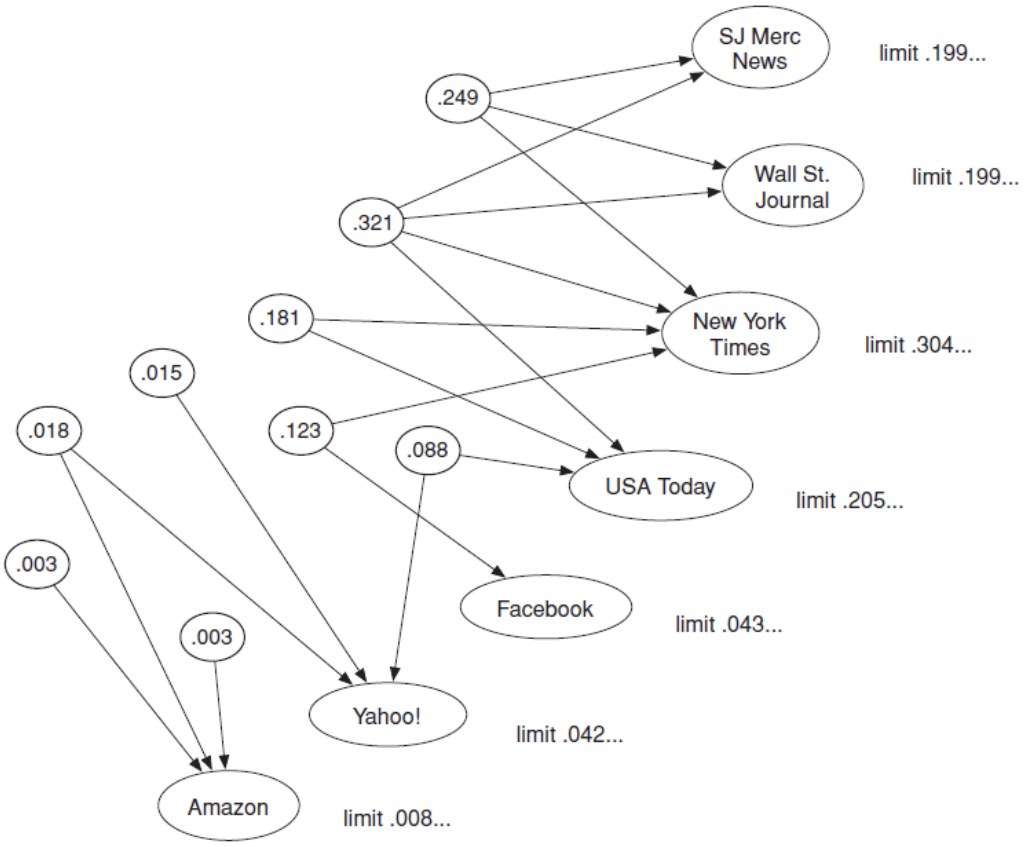


Abbildung 21: *Authorities* und *Hub* Werte für die Suchanfrage "Newspaper" berechnet und normalisiert Referenziert aus *Networks, Crowds and Markets: Reasoning About a Highly Connected World* [1])

Lässt man das ausgewählt *k* gegen unendlich laufen, erreichen die normalisierten Werte irgendwann eine bestimmten Grenze, an der die sich die Ergebnisse nicht mehr signifikante verändern. [1].

Dies ist nur ein grundlegendes Verfahren, was eine Art Einleitung in das PageRanking ist und im nächsten Kapitel noch detaillierter auf das Thema eingegangen wird. Es wurde gezeigt, dass nicht nur die möglichen Suchergebnisse ein Rolle bei der Bewertung spielen, sondern auch andere Knoten in dem Netzwerk, die durch ihre *In-Links* auf andere Homepages wichtiges Wissen besitzen, was verwendet werden muss.

3.3.3 PageRank

Bei *Hubs* und *Authorities* geht es grundlegend um die Idee, dass Internetseiten innerhalb des Netzwerkes mehrere Rollen haben, das heißt, dass wie vorher im Kapitel erklärt wurde, jede Seite eine wichtige Aussage über andere Seiten machen kann, obwohl sie selbst keine große Rolle spielt.

Es gibt aber noch andere Netzwerkstrukturen mit anderen Charakteristiken, für die es ein Bewertungsschema der einzelnen Knoten gibt. Betrachtet man ein solches Netzwerk bestehend aus Texten und Berichten, ist eine Seite wichtig, wenn sie von anderen wichtigen Seiten zitiert wird. Diese Verbindungsart findet man bei Netzwerken, die wissenschaftliche Literatur enthalten oder zum Beispiel bei akademischen oder behördlichen Homepages [17].

Diese Charakteristik bildet also die Grundlage für das *PageRanking* Verfahren und wird in folgenden Schritten durchgeführt:

- Ein Netzwerk hat *n* Knoten, man definiert nun für jeden Knoten den durchschnittlichen PageRank $1/n$
- Anzahl der Schritte *k* definieren
- Die folgende Updateregel *k*-mal durchführen:

  PageRankingUpdateRegel: Jeder Knoten teilt seinen PageRank mit der Anzahl an nach außen gehenden Verbindungen und erhält einen neuen PageRank, bestehend aus der Summe aus den zu ihm führenden Verbindungen. Hat ein Knoten keine von ihm weggehenden Verbindungen, wird der aktuelle Wert zu dem neuen PageRank addiert [1].

An dem in Abbildung 22 gezeigten Graphen wird es nochmal im Detail erklärt.

Schritt	1	2	3	4	5	6
1	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
2	$\frac{3}{12}$	$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{2}$
3	$\frac{13}{24}$	$\frac{3}{24}$	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{1}{24}$	$\frac{1}{3}$

Tabelle 1: PageRank Verteilung für *n=3* Schritte verteilt auf die sechs Knoten des Graphen

Der Graph besteht aus sechs Knoten. Daraus resultiert der durchschnittliche PageRank von jedem Knoten im ersten Schritt: $^1/_6$ ($^1/_k$). Im nächsten Schritt erhält der Knoten 1 ein PageRank von $^3/_{12}$, der sich aus den auf ihn gerichteten Kanten berechnet. Hierfür addiert man den kompletten Wert des PageRanks von Knoten 6 ($^1/_6$) und der Hälfte von Knoten 4 ($^1/_{12}$). Des Weiteren lassen sich Knoten 2, 4 und 5 auf die gleiche Weise berechnen, da sie jeweils die Hälfte ihres vorherigen Knotens erhalten. Knoten 6 berechnet sich aus dem kompletten fünften Knoten, addiert mit der Hälfte von 2 und 3.

Führt man dieses Verfahren nun noch *k*-Schritte durch, kristallisieren sich bestimmte Knoten heraus, die eine wichtige und zentrale Rolle innerhalb solcher Netzwerke spielen und höher geranked sind. In dem oben gezeigten Beispiel (k = 3) zeigt sich der signifikante Werteunterschied bei Knoten 1 ($^{13}/_{24}$) und darauf folgenden Knoten 6 ($^8/_{24}$).

Es kann hier nun passieren, dass das Netzwerk nicht so optimal aufgebaut ist, wie im Beispiel aus Abbildung 22, und in einem eher schlecht strukturierten Graphen die falschen Knoten einen hohen PageRank erhalten. Nun nimmt man an, dass es in Abbildung 22 keine Kante von Knoten 6 nach Knoten 1 gibt, sondern diese zu Knoten 5 übergeht. Führt man jetzt den oben erläuterten Algorithmus aus, entsteht das Problem, dass die PageRankwerte alle in Richtung des Knoten 5 und 6 „fließen“ und sich dort in den „Endknoten“ sammeln. Diese Problematik tritt in sehr vielen Netzwerken auf und wird durch die Einführung eines *Skalierungsfaktors* behoben [1].

Man führt hier standardgemäß die *PageRankUpdate*-Regel durch und den Skalierungswert *s* ein, der zwischen 0 und 1 liegt. Anhand dieses Parameters werden die eigentlichen Werte der Knoten runter skaliert. Daraus folgt, dass sich die Summe der PageRanks von 1 auf s ändert. Jetzt nimmt man den restlichen Wert 1 – s und verteilt diesen auf die restlichen Knoten im Graphen ((1 - s) / *n* pro Knoten). Abschließend ist noch zu sagen, dass das Problem für das Beispiel in Abbildung 22 hier nicht gelöst wird, da dieser Graph mit seinen sechs Knoten einfach zu klein ist, aber die Einführung des Skalierungsparameters die Lösung für komplexe Graphen bildet.

Dieses Verfahren hat zur Folge, dass nicht alles in die Richtung von eins, zwei Knoten „fließt“, sondern sich ein Teil der verschobenen PageRankwerte auf alle Knoten neu verteilt. Stellt man sich das bildlich vor, so regnet ein Teil des angesammelten Wassers in dem „Endknoten“ wieder von oben auf alle anderen Knoten herab. In der Praxis wird für den Parameter *s* ein Wert zwischen 0.8 und 0.9 gewählt. Des Weiteren führt dieses Verfahren zur Desensibilisierung des Graphen gegenüber dem Löschen oder Hinzufügen einer geringen Anzahl von Knoten und Kanten. [1,19]

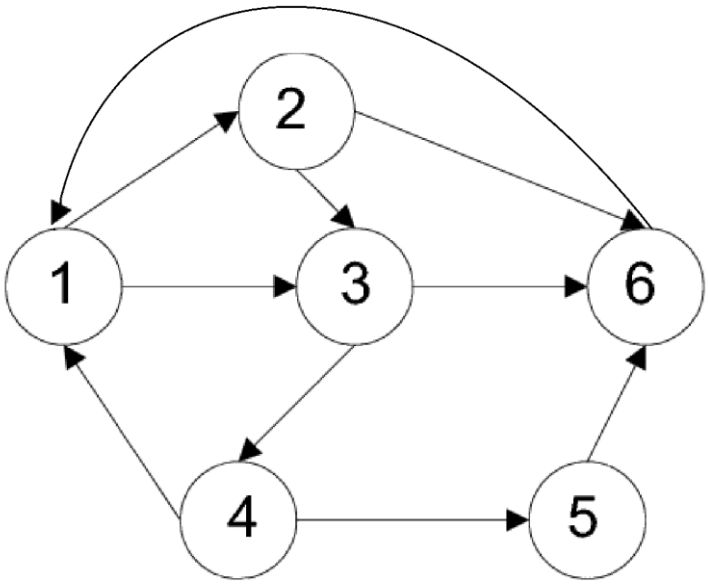


Abbildung 22: PageRanking

Die in diesem Kapitel erklärten Funktionen und Algorithmen sind natürlich nicht allein für die Resultate, die bei Internetsuchmaschinen wie "Google", "Yahoo!", "Bing" oder "Ask" resultieren, verantwortlich. Es werden noch viel komplexere und kompliziertere Wege zur Berechnung der Suchresultate herangezogen und verwendet, die hier nicht diskutiert und analysiert werden, da hier nur Grundlagen erklärt werden. Jede dieser Suchmaschinen verwendet seine eigens entwickelten Algorithmen, um möglichst präzise und effektive Ergebnisse zu liefern.

3.4 Navigation in Netzwerken

Nachdem die Grundlagen der Suche im Internet oder Netzwerken und deren Rankingsystem detailliert erklärt wurden, wird in diesem Abschnitt noch auf die Navigation in sehr großen Netzwerken eingegangen.

Die Navigation in sehr großen Graphen spielt heutzutage eine wichtige Rolle und wird zum Beispiel bei neuen Anwendungen im Bereich von *Wireless Networks*, allgemeine Sicherheit im Internet oder im Militärsektor benötigt [23].

In solch einem Graphen soll eine Nachricht möglichst schnell von einem Knoten zu einem anderen Knoten gelangen, bei dem Hubs (zentrale Knoten mit einem hohen Grad) verwendet werden. Benutzt man bei der Generierung eines Pfades Sprünge von Hub zu Hub, erreicht man im Durchschnitt die kürzesten Wege, da hier die Dichte und der Grad der Hubs stark zusammenhängen. Betrachtet man den "*Max-Degree Search*" Algorithmus, erkennt man den Zusammenhang zwischen der Länge eines Pfades und dem Grad der Hubs, welcher wiederum zu einer schnellen Navigation führt [24, 25].

Bei diesem Algorithmus navigiert man anhand der Höhe des Knotengrades der Hubs durch das Netzwerk. Es werden Knoten mit höherem Grad den anderen Knoten gegenüber bevorzugt. Man startet also bei einem Knoten *u1* und sucht sich den Nachbarknoten mit dem höchsten Grad, zu welchem dann weiter gegangen wird. Dies wiederholt man, bis man am Zielknoten angelangt ist. Dies ist natürlich ein sehr simpler Algorithmus, aber er beschreibt den ungefähren Ablauf bei der Navigation innerhalb von großen, unbegrenzten Netzwerken. Hier wird auch noch nicht berücksichtigt, dass eine bestimmte Typologie, wie zum Beispiel der Kreis, auftreten kann und diese dann noch erkannt werden muss. Dazu gehören auch noch die Endknoten, bei denen auf vorherige Knoten zurück gesprungen werden muss. Im Worst Case durchsucht dieser Algorithmus das komplette Netzwerk.

Abschließend ist zu sagen, dass hier nur kurz erläutert wurde, wie Navigation in solchen Graphen funktioniert. Es gibt eine Vielzahl von Verfahren und Algorithmen, die bestimmen, wie man sich durch solch ein Netzwerk effizient bewegen kann. Einige davon sind auch in dem in dieser Arbeit verwendetem Framework implementiert.

3.5 Domain Model

In diesem Kapitel wird das Domain Model erläutert, das im Kontext dieser Arbeit als Grundlage für das Wissensmodell verwendet wurde. Das Domain Model wird verwendet, um benötigte Hintergrundinformationen bereitzustellen und daraus die automatisierte Navigation durch das Netzwerk zu ermöglichen.

3.5.1 Grundlagen

Bei einem Domain Model handelt es sich um ein Objektmodell, das einen bestimmten Problembereich eines Systems beschreibt. Elemente eines solchen Models sind die einzelnen Klassen und deren Beziehungen (Verbindungen) zueinander.

Allgemein kann man sagen, dass das Domain Model ein Konzept ist, das ein System und deren wichtigste Problembereiche darstellen soll. Dazu gehören nicht nur die Problembereiche, die hier analysiert werden, sondern auch die einzelnen Beziehungen zwischen den Objekten und deren Attribute. Solche Modelle werden häufig und richtiger Weise mit den objektorientieren Modellen in Verbindung gebracht, da ähnlich wie bei diesen hier eine strukturelle Ansicht der Domains durch Erweiterungen, wie zum Beispiel Use Cases, erzeugt wird.

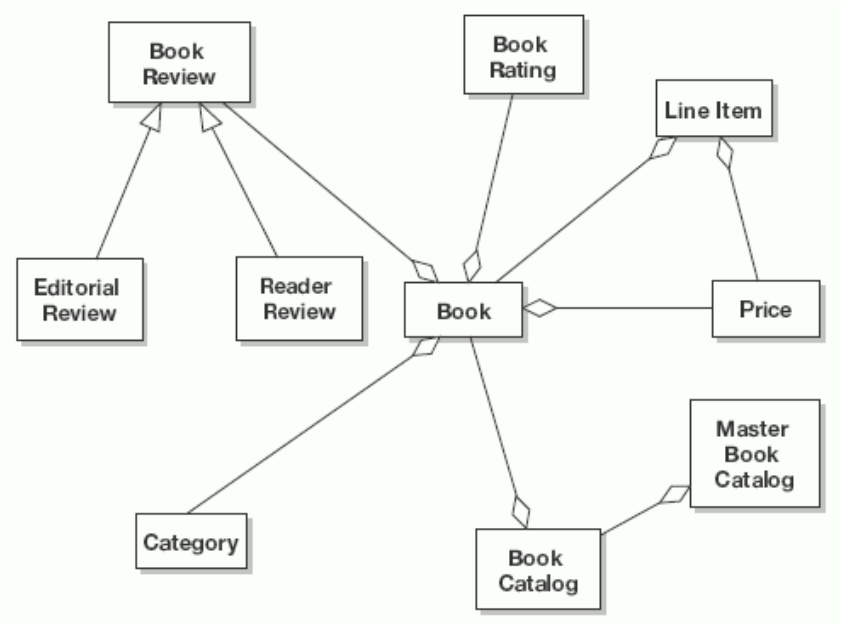


Abbildung 23: Domain Model eines Buches [29]

In Abbildung 23 zeigt solch ein Domain Model die Struktur eines Buches und deren Relation zu anderen Objekten wie dem Preis oder der zugehörigen Kategorie.

Ein Domain Model kann nicht nur, wie oben beschrieben, als eine Art Analyse für einen Problembereich herangezogen werden, sondern auch als ein Wissensnetzwerk. Dieses Netzwerk besteht aus mehreren kleinen sogenannten *domain knowledge elements* (DKE) und soll eine Wissensstruktur darstellen. Jeder dieser einzelnen DKE's repräsentiert einen wichtigen Teil des Wissens innerhalb eines solchen Domain Models. Diese Variante eines Domain Models wird in allen bekannten sequencing engines verwendet [30], aber auch in anderen Systemen. Dort verwendet man hingegen eine unterschiedliche Namensgebung für die DKE's, wie zum Beispiel *concepts*, *topics* oder *learning outcomes.* Trotz der differenzierten Bezeichnung ist jede ein elementarer Part eines sogenannten *DKE Konzepts*.

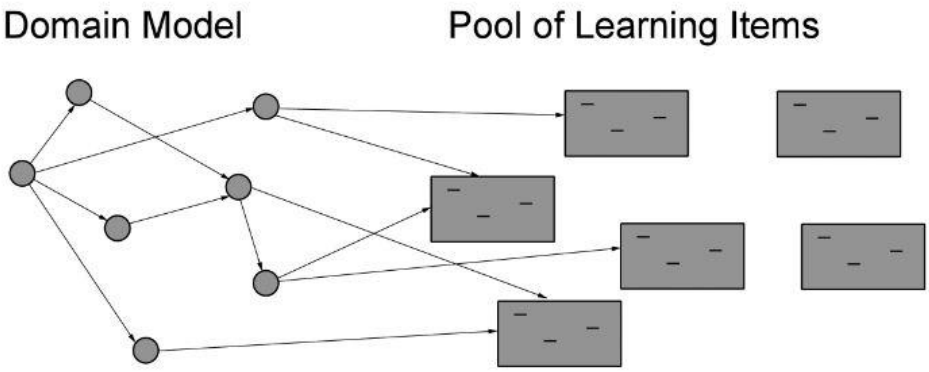


Abbildung 24: Verbindet ein Domain Model mit Learning Items [30]

Die simpelste Form eines solchen Domain Models ist ein Modell, das ohne Verbindungen zwischen den Objekten besteht. Solch eine Version wird als *set model* oder *vector model* bezeichnet, da es keine interne Struktur gibt. Verbindet man diese, erhält man, wie in Abbildung 24 zu sehen ist, ein Netzwerk aus KDE's. Des Weiteren lässt sich erkennen, dass man solch ein Netzwerk auch mit weiteren Informationen verbinden kann [30]. In diesem Beispiel wird das Domain Model noch mit einer Anzahl von zu lernenden Items verbunden, die Einfluss auf das Netzwerk haben und dessen Navigation und Eigenschaften signifikant beeinflussen können.

Es ist nötig, diese Items gegenüber dem Domain Model einem Index zuzuweisen, damit das Netzwerk mit den Informationen arbeiten kann. Hierfür gibt es viele Wege um dies umzusetzen [31]. In dieser Arbeit befasst man sich aber nur mit einem bestimmten Verfahren, da dies für die im nächsten Kapitel beschriebenen Wissensmodelle benötigt wird.

Im Vergleich zu dem in Abbildung 24 dargestellten Verfahren des *multi-concept indexing*, wird nun das *single-concept indexing* erläutert [32,33].

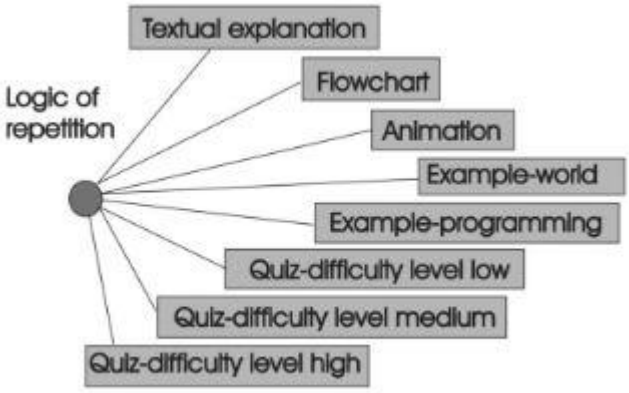


Abbildung 25: Single-concept indexing [30]

Single-concept indexing ist eine simple und mehr intuitive Variante des Indexing. Normalerweise verwendet man diese Version, wenn man kleinere und simplere Domains hat. Man kann an dem Beispiel aus Abbildung 25 erkennen, dass im Vergleich zu *multi-concept indexing* hier jedes einzelne Objekt mit verschiedenen Informationen verbunden wird, die sich aber nur auf dieses eine Objekt beziehen, wobei sich bei *multi-concept indexing* diese Informationen auf mehrere Objekte verbinden lassen und das Konzept somit komplexer wird und besser ausgebildete Autorisierungsteams benötig.

Domain Models sind zusammenfassend ein mächtiges Werkzeug, was nicht nur zur Fehleranalyse eines System verwendet werden kann, sondern kann es auch durch eine anderes Interpretation und Definition zu einer mächtigen Darstellung von zum Beispiel Wissensnetzwerken verwendet werden. Daraus lassen sich große Vorteile in jeder Art von *sequencing systems* und anderen Bereichen festigen. Beispiele für solch ein System wären DCG und CoCoA, die unter anderem *single-concept* und *multi-concept indexing* verwenden [30].

3.5.2 Wissensmodelle

In dieser Arbeit werden sogenannte *Wissensmodelle* für die Entscheidung der Knotenselektion benötigt. Diese werden aus der Kombination eines Domain Models und den mit den einzelnen Knoten verbundenen Hintergrundinformationen erstellt.

Das in dem letzten Kapitel beschriebene Domain Model wird hier in einer simpleren Variante verwendet und repräsentiert das Netzwerk, bei dem jeder einzelne Knoten einem Artikel aus dem "Wikipedia for Schools" [27] entspricht. Dieses Netzwerk besitzt einen Pool aus Items (Categoryfield), die dem "Pool of learning Items" aus Abbildung 24 gleichgesetzt sind. Diese Items repräsentieren jeweils eine Kategorie, wie zum Beispiel Geographie oder Kunst. Untereinander sind diese Kategorien verbunden und zeigen auf die Artikel, die dieser Kategorie entsprechen. Dies würde zum Beispiel bedeuten, dass der Artikel über die "französische Revolution" mit dem Knoten "Geschichte" verbunden ist.

Diese Struktur wird in Abbildung 26 dargestellt und zeigt zentral den Pool aus den Kategorien, die wiederum mit *n* Knoten des Netzwerkes verbunden sein können.

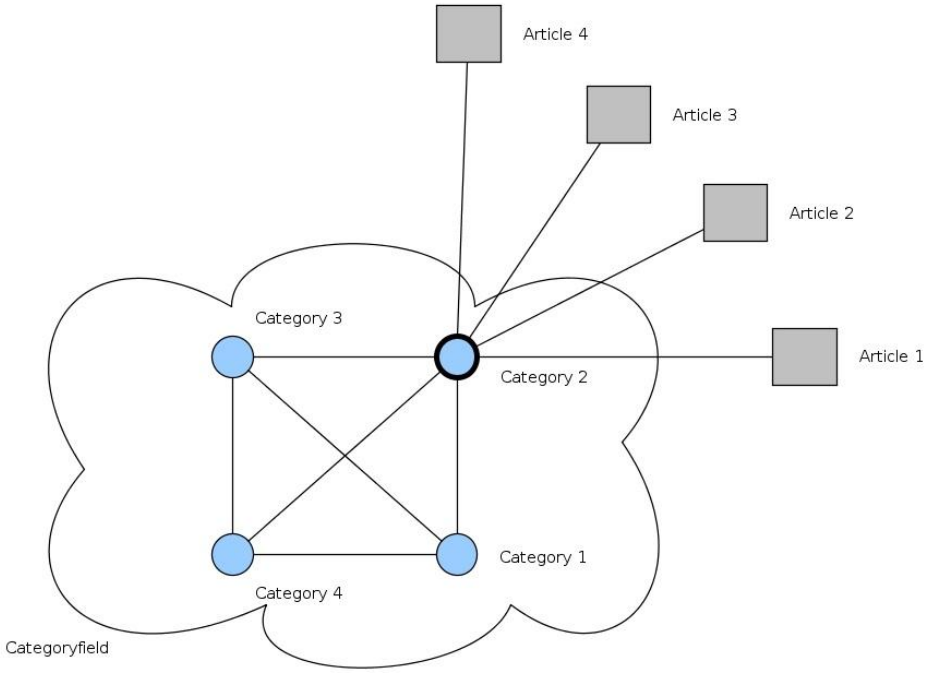


Abbildung 26: Darstellung der "learning Items" mit dem Domain Model

4 Simulation der Wissensmodellnavigation

Es wurden nun in der bisherigen Arbeit grundlegende Informationen über das Thema Graphentheorie beschrieben, die unter anderem wichtig für das Verständnis der Funktionsweise des Algorithmus sind. In diesem Kapitel wird anhand eines detaillierten Beispiels der Ablauf der Knotenselektion illustriert. Dieser beinhaltet die Darstellung eines Wissensmodells und die Beeinflussung durch das benötigte Hintergrundwissen über das Netzwerk.

Darauf folgt eine mathematische Erklärung des Algorithmus und die Einbettung des Codes in das Framework. Dafür benötigte Zusatztools werden hier auch zusammengefasst erklärt. Diese Tools werden benötigt, um zum einen die Codierung der originalen Dateien für das Framework anzupassen und zum anderen, um die Wissensverteilung zu kalkulieren. Der letzte Abschnitt besteht aus der Beschreibung des kompletten Frameworks, also dessen Aufbau, dazugehörigen Klassendiagrammen, interner Ablauf und Konfiguration. Es werden die wichtigsten Klassen und deren Vererbung gezeigt, aber auch wie diese miteinander in Verbindung stehen.

4.1 Navigation innerhalb des Domain Models

In den vergangenen Kapiteln wurde im Allgemeinen das Domain Model und dessen Adaptierung auf das hier verwendetet Netzwerk erklärt. In diesem Kapitel wird anhand eines detaillieren Beispiels erläutert, wie genau die Knotenselektion mithilfe des beschriebenen Wissensmodells und den Hintergrundinformationen arbeitet. Die Knotenselektion bildet eine der Kernpunkte des Programms und bestimmt, wie ein Pfad innerhalb des Netzwerkes von einem bestimmten Startknoten zu einem ausgesuchten Zielknoten gewählt wird. Im aktuellen Framework sind noch weitere Methoden implementiert. Dazu gehören zum Beispiel die Greedy- oder Randommethode, bei denen zum einen der Nachbarknoten per Zufall gewählt (Randommethode) oder versucht wird, über Hintergrundwissen den kürzesten Pfad zu finden (Greedymethode).

4.1.1 Selektion der Knoten

Der in dieser Arbeit entwickelte Algorithmus versucht einen Pfad zu finden, der dem des Menschen so nahe wie möglich kommt, indem er anhand eines Wissensmodells versucht "vorherzusagen", welchen Knoten man als nächstes wählen wird, unter Zuhilfenahme von Hintergrundwissen und dem Wissensmodell.

Um dies etwas zu verdeutlichen, erklärt man das Verfahren an einem Beispiel, nämlich dem sogenannten *Wikispeedia* [26]. Dabei handelt es sich um ein Online-Spiel, bei dem man versucht, mit möglichst wenigen Klicks von einem vorgegebenen Startartikel zu dem angegebenen Zielartikel zu gelangen. Das Spiel benutzt als Artikeldatenbank das "Wikipedia for Schools" [27] Netzwerk (Informationsnetzwerk), welches auch für diese Arbeit die Grundlage als verwendetes Netzwerk bildet. Es wird nun versucht, einen solchen Spielablauf des Users durch das Netzwerk des Wikigames zu berechnen. Dafür stehen auch die Spieldaten von den Usern zur Verfügung, die dieses Spiel schon online getestet haben.

Wie das im Detail aussieht und wofür diese Userdaten benötigt werden, wird im nächsten Abschnitt erklärt.

4.1.2 Navigation anhand des Wissensmodells

Nun nimmt man an, dass in solch einem Spielablauf der User abhängig von seinem Wissen, das er besitzt, sich für bestimmte Artikel beim Weiterklicken entscheidet oder in bestimmten Bereichen (Kategorien), in denen er sich sehr gut auskennt, sich zuerst einen Überblick schafft, um dann speziell nach dem Zielartikel zu suchen. Somit kann man sagen, dass ein User mit höherer Wahrscheinlichkeit den nächsten Knoten (nächster Artikel) wählt, der einer bestimmen Kategorie zugeordnet ist, die seinem Wissensmodell entspricht.

Um dies umzusetzen, benötig man nun die Wissensverteilung, die die User im Durchschnitt haben, um eine Art Wahrscheinlichkeitsverteilung der einzelnen Kategorien aufstellen zu können. In dieser Arbeit werden vier verschiedene Modelle angenommen und getestet, die im nächsten Kapitel genauer beschrieben und erklärt werden.

Angenommen, man hat nun solch eine Wahrscheinlichkeitsverteilung der einzelnen Kategorien, so kann man für jeden einzelnen Knoten, auf dem man sich befindet, den Nachbarknoten mit der höchsten Wahrscheinlichkeit, mit der er auftritt, berechnen. Es werden hier natürlich die restlichen Nachbarknoten auch mit einer Auftrittswahrscheinlichkeit versehen, da es ja

möglich ist, dass der User nicht immer den wahrscheinlichsten Knoten nimmt, sondern auch in Bereichen sucht, in dem sein Wissen nicht so ausgeprägt ist.

Des weiteren wird zur Berechnung der Wahrscheinlichkeit noch das sogenannte Hintergrundwissen (Backgroundknowledge) herangezogen, bei dem es sich um Informationen über das ganze Netzwerk handelt. Im Detail bezieht man hier noch das Wissen über den kürzesten Weg zum Ziel mit ein.

Führt man dieses Verfahren nun für jeden Knoten von einem Start- und Zielknoten aus, errechnet sich dadurch eine Navigation durch das Netzwerk, die dem eines Users ähneln soll.

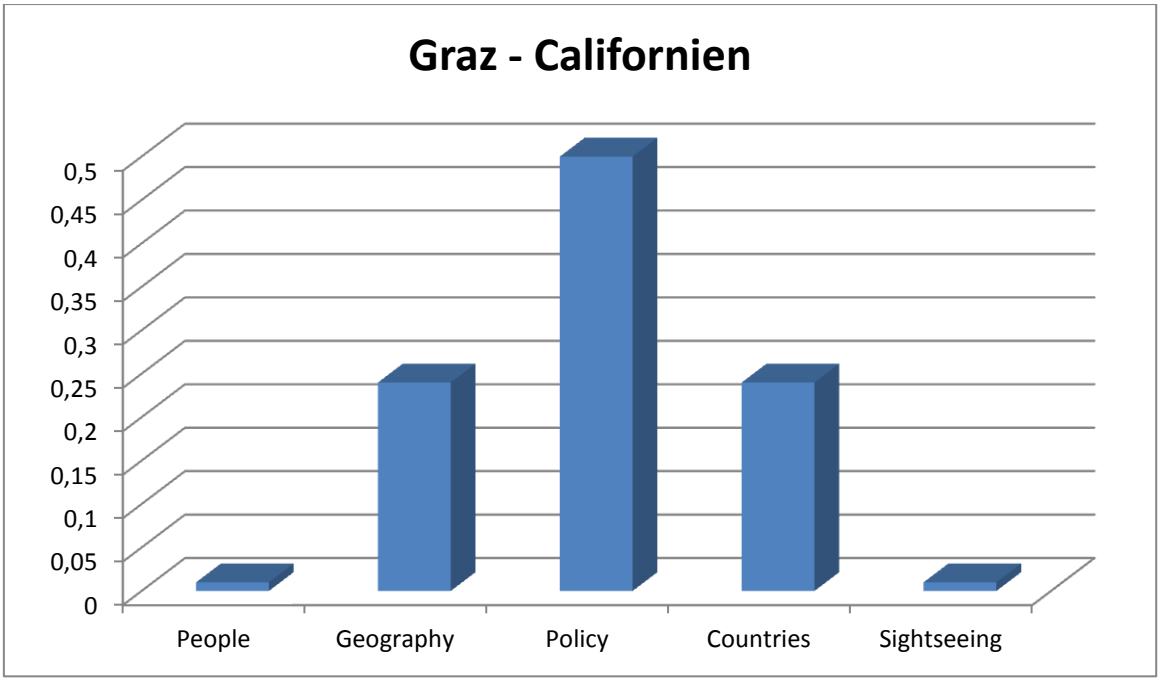
Um dieses Verfahren zu verdeutlichen, wird es nochmal an einem Beispiel erklärt, welches in Abbildung 27 zu sehen ist. Dafür benötigt man noch die einzelnen Kategorien der Knoten und die zu den Kategorien gehörenden Wahrscheinlichkeiten, die wiederrum aus dem Wissensmodell des Users gefiltert werden. In der folgenden Tabelle ist die Kategoriezugehörigkeit der einzelnen Knoten beschrieben, die in der Abbildung 27 enthalten sind.

Artikel	Kategorie
Graz	Countries
Urturm	Sightseeing
Schwarzenegger	People
Europa	Geography
Atlantik	Geography
Politik	Policy
USA	Countries
Staat	Countries
Californien	Countries

Tabelle 2: Kategorien der einzelnen Knoten

Jeder einzelne Knoten besitzt eine bestimmte Kategorie, die im "Wikipedia for Schools" definiert ist. Es kann hier natürlich Artikel geben, die durch mehrere Kategorien definiert sind.

Bei solch einem Fall wird die sogenannte Topkategorie als Kategorie des Artikels gesetzt. Weiß man nun, um welche Kategorie es sich handelt, benötigt man die Wahrscheinlichkeit, mit welcher der User sie auswählen würde (also wie gut sich derjenige im Umfeld des Artikels auskennt). In diesem Beispiel hat der User (Wissensmodell "Graz - Californien) ein sehr ausgeprägtes Wissen im Bereich der Politik und ebenfalls grundlegendes in Geographie und Länderkunde. Weniger weiß diese Person von Dingen, die bekannte Personen angehen oder über das Thema "Sightseeing".



Berechnet man nun einen Pfad von dem Startartikel "Graz" zu dem gesuchten Zielknoten "Californien" (Abbildung 27), abhängig von der Kategorie ohne das Hintergrundwissen, würde sich eine Wahrscheinlichkeit von 60% für den Artikel "Europa" ergeben, da "Europa" zur Kategorie Geographie gehört. Im Gegensatz dazu beträgt die Wahrscheinlichkeit für die anderen Nachbarknoten hier nur jeweils 20%, die aus der niedrigen Wissensverteilung in diesen Bereichen resultieren. Bezieht man jetzt noch die Anzahl an Schritten des kürzesten Weges hinzu, würde die Wahrscheinlichkeit des Nachbarknotens "Urturm" noch weiter sinken, da im Vergleich zu den anderen Nachbarknoten, der Weg zum Zielknoten "wesentlich" länger ist. (Ist in dieser Grafik nicht zu erkennen, wird aber angenommen)

Es wird nun mit hoher Wahrscheinlichkeit von dieser Person der Europaartikel gewählt, in dem er entweder mit der USA oder dem Atlantik fortfahren kann. Hier ist die Entscheidung gleichverteilt, da die Wahrscheinlichkeiten gleich sind. Betrachtet man aber noch das Hintergrundwissen, erkennt man, dass "USA" den kürzeren Weg liefert und somit dieser Artikel

eine höhere Wahrscheinlichkeit erhält. Nehmen wir an, er entscheidet sich für Amerika, gelangt er direkt zu dem gesuchten Zielknoten. Würde er sich nicht für diesen Nachbarknoten entscheiden, gelangt er mit hoher Wahrscheinlichkeit im nächsten Schritt zu diesem Knoten, da dort anhand der Kategorie zu 70% dieser Knoten gewählt wird.

Natürlich kann es auch passieren, dass hier nicht Europa als erster Knoten gewählt wird sondern über Arnold Schwarzenegger angefangen wird. Auf diesem Weg gelangt die simulierte Navigation auch zum Zielknoten.

Nun sollte klar sein, wie der Algorithmus funktioniert und wie dieser Nodeselector arbeitet. Dafür werden Hintergrundinformationen benötigt, die bereit gestellt wurden und die Kategorien der einzelnen Artikel enthalten.

Des Weiteren ist es sehr wichtig, wie man das Wissensmodell wählt, das den User repräsentieren soll. Dafür wurden in dieser Arbeit vier Versionen getestet, welche im späteren Kapitel genauer erläutert werden.

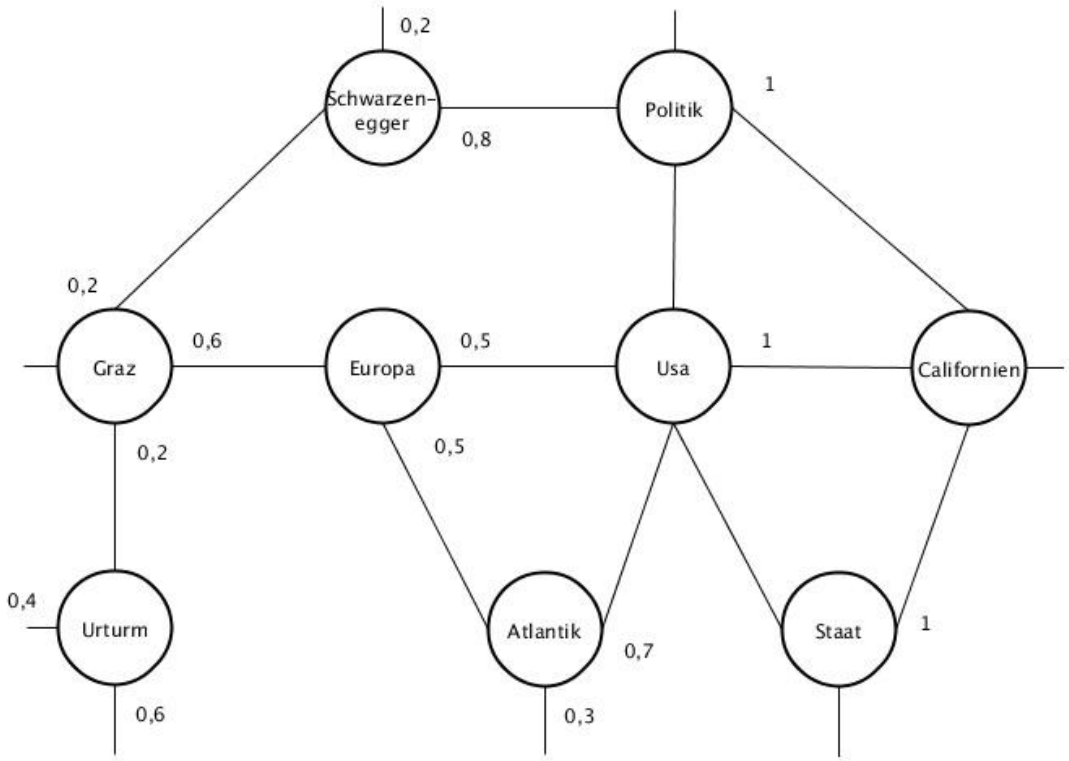


Abbildung 27: Graph mit der Wahrscheinlichkeitsverteilung durch Kategorie und Wissensmodell des Users

4.1.3 Entwicklung des Algorithmus

Nachdem nun die grundlegende Idee hinter dem Verfahren an einem detailiert ausgeführtem Beispiel erläutert wurde, wird nun beschrieben, welche verschiedene Ansätze getestet wurden um das Best mögliche Ergebnis zu erhalten.

- *prob*
  Hier wird, wie im vorherigen Beispiel der Bezug zwischen der Wahrscheinlichkeit der Kategorie des Knotens und dem Hintergrundwissen der minimalen Distanz zum Zielknoten hergestellt. Es stellt die einfachste und direkteste Umsetzung der Idee dar.

$$p_{\,node} = \left[\frac{[p_{category}]_0^{100}}{dist_{min}}\right]_0^1$$

- *prob(*)/dist*
  Um die Performanz der Berechnung der Wahrscheinlichkeit noch zu verbessern, musste das Verhalten der User und die Struktur des Netzwerkes noch genauer analysiert werden. Es stellte sich bei den Analysen heraus, dass sie die Kategorien der Knoten in den Pfaden durch das Netzwerk öfter als gedacht der Kategorie des Zielknotens entsprechen. Des weiteren werden Zielknoten im vorletzten Schritt meist direkt gefunden. Diese Informationen werden hier ausgenutzt und bei der Entscheidung für den nächsten Knoten herangezogen.

$$p_{\,node} = \left[\frac{[p_{category}]_0^{100}}{dist_{min}}\right]_0^1$$

$$p_{\,node=tagetnode} = 1$$

$$p_{\,node_{category} = targetnode_{category}} = \left[\left(\frac{[p_{category}]_0^{100}}{dist_{min}}\right)^{nodedegree}\right]_0^1$$

- *pow(prob,nodedegree)(*)/pow(dist,nodedegree)*
  Für die eben beschriebene Berechnung der Wahrscheinlichkeit für den jeweiligen Knoten wird hier noch versucht eine bestimmte Problematik des Netzwerkes zu lösen. Da es in diesem sehr großen Graphen passieren kann, dass ein Knoten eine verhältnismäßig hohe Anzahl an Nachbarn hat, fallen die Wahrscheinlichkeiten einzelner Knoten kaum ins Gewicht. Um dieses Problem zu beheben, werden die Wahrscheinlichkeiten exponenziert. Diese Problematik wird in einem späteren Abschnitt noch genauer beschrieben.

$$p_{\,node} = \left[\frac{\left(\left[p_{category} \right]_0^{100} \right)^{nodedegree}}{(dist_{min})^{nodedegree}} \right]_0^1$$

$$p_{\,node=tagetnode} = 1$$

$$p_{\,node_{category} = targetnode_{category}} = \left[\left(\frac{\left(\left[p_{category} \right]_0^{100} \right)^{nodedegree}}{(dist_{min})^{nodedegree}} \right)^{nodedegree} \right]_0^1$$

- *shortest dist -> probability (th = +1)*
  Diese Variante ähnelt dem Algorithmus des "Greedy" Verfahren, da man hier ebenfalls eine Liste aus den Nachbarknoten sammelt, die den kürzesten Weg zum Ziel entsprechen. (Kann sich natürlich auch nur um einen Knoten handeln) Aus dieser Liste wird anhand der Wahrscheinlichkeiten der Kategorien der Knoten der nächste Schritt ausgewählt.

$$Node = (node_{minDist\,+\,1})_{category}$$

- *shortest dist -> probability (th = +2)*
  Hier werden nun nicht nur die Knoten mit der kürzesten Distanz in die Liste hinzugefügt, sondern der Grenzwert für die Distanz noch um zwei Schritte erhöht. Somit wird der Bereich im Netzwerk, der analysiert wird noch erweitert. Ebenfalls steigt die Vielfallt der Knoten, die für die Wahrscheinlichkeitenberechnung herangezogen werden.

$$Node = (node_{minDist\,+\,2})_{category}$$

- *shortest dist -> probability (th = +4)*
  Selbiges Verfahren wie zuvor, nur wird hier der Grenzwert nochmals um zwei Punkte nach oben gesetzt, also auf eine Distanzbegrenzung von *4*.

$$Node = (node_{minDist\,+\,4})_{category}$$

In Abbildung 28 sieht man nun, wie gut die einzelnen Algorithmen zueinander abgeschnitten haben. Diese Statistik zeigt auf der X-Achse den aktuellen Schritt auf den Pfaden an. Es lässt sich hier gut erkennen, dass der Großteil der Pfade eine durchschnittliche Länge von 1-15 hat. Dies kann man aus dem stark schwankendem und nicht linearen Verlauf der Grafik ab Knoten 15 erkennen. Auf der Y-Achse ist die Trefferquote in Prozent angegeben, mit der der jeweilige Algorithmus dieselben Entscheidungen wie die User getroffen haben.

Es lässt sich klar erkennen, welcher der entworfenen Algorithmen die besten Ergebnisse liefert. Die effektivste Berechnung liefert hier der in der Statistik mit Grün gekennzeichnete, da dieser eine Overall Trefferquote von 24,8885% aufweist. Wenn man dies nun von den 218671 Knoten der Pfade auf die Anzahl der Treffer runter rechnet, werden 54424 Knoten richtig berechnet bzw. "vorhergesehen".

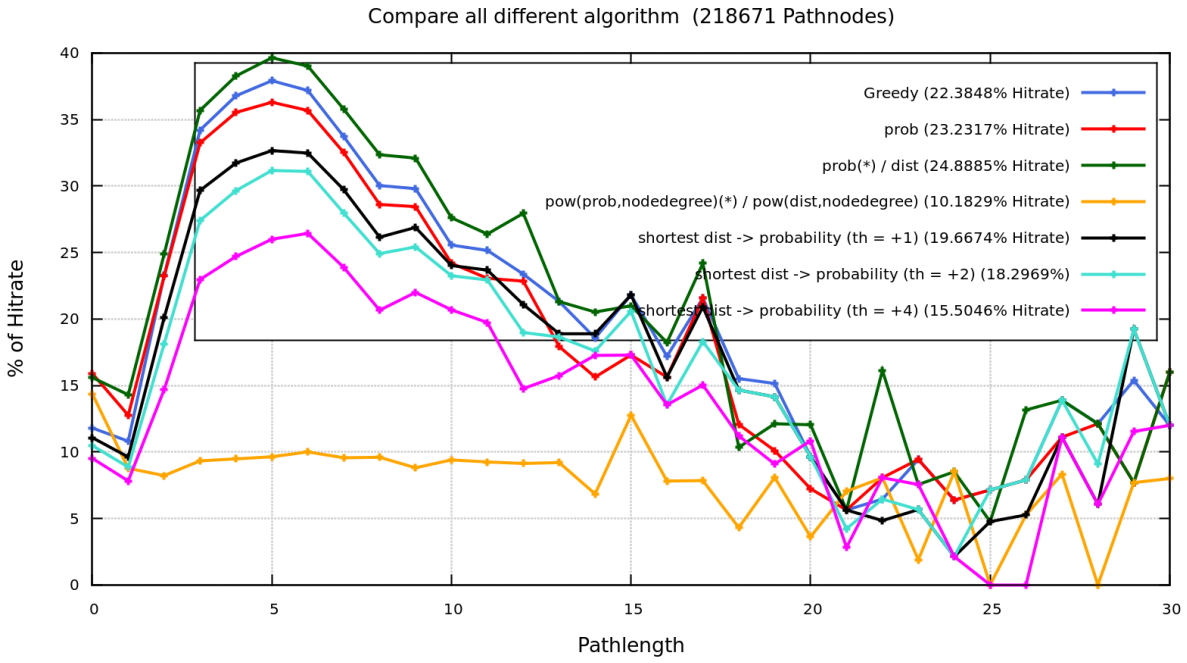


Abbildung 28: Resultate der verschiedenen Algorithmen im Vergleich

4.1.4 Navigation im Detail

In diesem Abschnitt wird noch mal im Detail auf die Formel und Funktionsweise der Formel im Source Code eingegangen. Im Allgemeinen wird für die Berechnung der Wahrscheinlichkeit folgende Formel verwendet:

$$p_{\ node} = \left(\frac{{p_{category}}^{nodedegree}}{dist_{min}}\right)^{nodedegree}$$

$p_{category}$ entspricht der Wahrscheinlichkeit, mit der die Kategorie bzw. der Artikel gewählt wird und $dist_{min}$gibt die minimale Pfadlänge von dem aktuellen Knoten zum Zielknoten an, welches aus dem Hintergrundwissen des Graphen bezogen wird. Setzt man diese beiden Parameter in Bezug zueinander, erhält man die Wahrscheinlichkeit, mit der der Artikel ausgewählt wird. Die Wahrscheinlichkeit wird hier in einem Wertebereich von $[0,1]$ beschrieben, wohingegen die Distanz zum Zielknoten nur ganzzahlige Werte annehmen kann, somit erreicht man durch eine höhere Wahrscheinlichkeit und eine geringere Distanzen einen insgesamt hohen resultierende Wahrscheinlichkeitswert für den jeweiligen Knoten. Dem im letzten Kapitel angesprochene Problem des teilweise sehr hohen Grades mancher Knoten (Hubs) wird nun durch das miteinbeziehen des Knotengrades in die Formel entgegengewirkt. So sollen hier die Kategorien abhängig vom Grad des Knotens einen exponentiellen Wert erhalten.

In Abbildung 29 sieht man den Codeausschnitt, in dem alle Knotennachbarn des aktuellen Standortes durchlaufen werden, der Wahrscheinlichkeitswert der Kategorie ausgelesen (Zeile 121 und 125), die Distanz angefordert wird (Zeile 116 bzw. 129) und die resultierende Wahrscheinlichkeit der einzelnen Knoten dann in ein Array geschrieben werden (Zeile 134).

```
112 for (TIntSet::TIter iter = next_node_candidates->BegI();
113         iter < next_node_candidates->EndI(); iter++) {
114
115     const int node_id = iter.GetKey();
116     const int current_distance = BackgroundKnowledge_->GetDistance(
117             node_id, target_node);
118
119     if (Graph_->GetCategoryOfId(node_id)
120             == Graph_->GetCategoryOfId(target_node)) {
121         current_probability = (double) pow(
122                 probabiltityList_[Graph_->GetCategoryOfId(node_id)],
123                 Graph_->GetOutDeg(node_id));
124     } else
125         current_probability =
126                 (double) probabiltityList_[Graph_->GetCategoryOfId(
127                         node_id)];
128
129     current_probability /= (double) (current_distance);
130     current_probability = pow(current_probability,
131             Graph_->GetOutDeg(node_id));
132
133     probability_sum += current_probability;
134     probabilities[max_node_position] = current_probability;
135     max_node_position += 1;
136 }
```

Abbildung 29: Berechnung der Wahrscheinlichkeit unter Zuhilfenahme der Kategorie und des Hintergrundwissens

Die eigentliche Berechnung wird hier noch modifiziert, indem vorher überprüft wird ob es sich um den letzten Knoten handelt, der im Spiel gesucht werden soll. Handelt es sich um den gesuchten Knoten oder Artikel, wird dieser direkt zurück gegeben. Andernfalls wird die gegeben Wahrscheinlichkeit für den Knoten verwendet. Des weiteren wird noch zusätzlich abgefragt, ob die Kategorie des zu überprüfenden Knoten mit der Kategorie des Ziels übereinstimmt (Zeile 119). Ist dies der Fall, wird diese Wahrscheinlichkeit potenziert und somit stärker gewichtet (Zeile 121-123). Der Grund für diesen Fall der Wahrscheinlichkeitsberechnung ist in Abbildung 30 zu sehen. Es besteht eine Hohe Wahrscheinlichkeit, dass ausgewählte Knoten der User zu der selben Kategorie der Zielknoten gehören und somit einer höheren Priorität zugeordnet werden sollten. Zum Vergleich wurden nicht nur die Resultate der User dargestellt, sondern auch die der anderen Algorithmen.

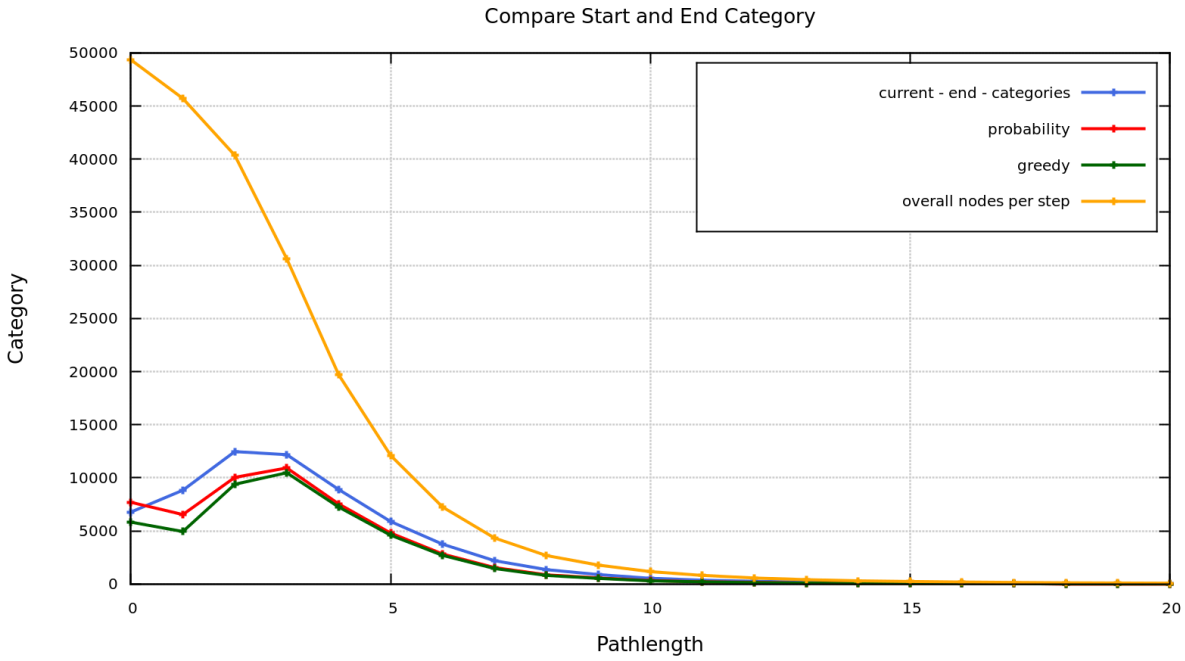


Abbildung 30: Zeigt an, wie viele Knoten es gibt, bei denen die Kategorie der aktuellen Knoten und dem Zielknoten übereinstimmen

Daraufhin wird eine zufällige Zahl zwischen Null und der Summe über alle vorher berechneten Knoten erzeugt (Zeile 211 und 212), die dann die Position des gewählten Knotens im Wertebereich angibt (siehe Abbildung 31). Diese Position wird gefunden, indem das Array durchgegangen wird und alle Wahrscheinlichkeiten summiert werden, bis die Zufallszahl erreicht ist und der Knoten, der an dieser Position im Array sitzt, gewählt wird (Zeile 216 - 221).

```
211 TRnd seed(0);
212 double random_number = seed.GetUniDev() * probability_sum;
213 double current_probabilitiy_sum = 0.0f;
214 int result_position = 0;
215
216 for(int position = 0; position < max_node_position;position++) {
217     current_probabilitiy_sum += probabilities[position];
218     if(current_probabilitiy_sum >= random_number){
219         result_position = position;
220         break;
221     }
222 }
```

Abbildung 31: Bestimmung des Zufallswertes und Bestimmung der Position des Knotens

Wurde die Position des Knotens ermittelt, muss jetzt nur noch der Knoten gesetzt (Zeile 231 und 232), die Schleife unterbrochen werden (Zeile 234) und die Parameter an das Programm zurückgeliefert werden.

```
224 //get node out of its postitions
225 int current_position = 0;
226
227 for (TIntSet::TIter iter = next_node_candidates->BegI();
228         iter < next_node_candidates->EndI(); iter++) {
229     const int node_id = iter.GetKey();
230     if (current_position == result_position) {
231         result->node = node_id;
232         result->distance = BackgroundKnowledge_->GetDistance(
233                 node_id, target_node);
234         break;
235     }
236     current_position++;
237 }
```

Abbildung 32: Knoten setzen und zurückliefern

4.1.5 Verwendeter Datensatz

In dieser Arbeit wurden zwei unterschiedliche Datensätze in Betracht gezogen. Dabei handelt es sich zum einen um das Artikelnetzwerk der Enzyklopädie Wikipedia und zum anderen um den Datensatz eines Schulwikipedias, dem „Wikipedia for Schools“ [27].

Der Grund, warum hier der Schulwikipedia Datensatz herangezogen wurde, ist, dass der Wikipedia Datensatz für die Testung des Programms viel zu groß ist, um damit einen Algorithmus zu entwickeln und zu programmieren. Im Vergleich zu dem Wikipedia Datensatz benötigt man für einen Durchlauf mit dem kleineren Set aus dem Schulwikipedia schon mehrere Minuten und es würde bei dem originalen Wikipedia um ein Vielfaches länger dauern, was die Entwicklung eines Algorithmus sehr erschwert. Des Weiteren stehen die Kategorien des Wikipedias nicht zur Verfügung, wodurch es nicht möglich ist, diesen Algorithmus dort anzuwenden, da die Kategorien eine der wichtigsten Komponenten bei diesem Verfahren sind.

4.1.5.1 Wikipedia for Schools

Bei diesem Datensatz handelt es sich, wie oben schon erwähnt, um eine kleinere Version des Wikipedias, was für Schulkinder ausgelegt ist und insgesamt ca. 6.000 Artikel und 50.000 Bilder enthält und bietet somit eine solide Grundlage, um den Algorithmus effizient zu testen, ohne sich von dem originalen Datensatz zu sehr zu differenzieren.

Abbildung 33: Wikipedia für Kinder und Jugendliche [27]

Der Hintergrund dieses Projekts ist der Gedanke, ein Wikipedia speziell für Kinder und Jugendliche zu schaffen, welches das sogenannte "National Curriculum" erfüllt. Somit soll eine sichere Wissensquelle geschaffen werden, bei der alle Artikel geprüft sind und von keiner nicht autorisierten Person verändert werden können.

Dieses Netzwerk entspricht genau denselben Anforderungen, die hier benötigt werden. Es besteht aus vielen Knoten, die den Artikeln entsprechen, und jeder Knoten besitzt eine bestimmte Zugehörigkeit, also die Kategorien. Des Weiteren gibt es auch Userdaten, die das "Wikigame" innerhalb dieses Netzwerkes gespielt haben.

Somit sind alle Kriterien für diesen Datensatz erfüllt, um ihn für die Entwicklung des Programms zu verwenden.

4.1.5.1.1. Verwendete Konverter

ConvertProbabilityStringData:

Da die Datensätze des „Wikipedia for Schools“ nicht in der passenden Konvertierung vorgelegen haben, musste hierfür noch ein zusätzliches Programm geschrieben werden, welches dieses Problem beheben soll. Die Struktur der Dateien sieht man in Abbildung 31. Es wird in dieser Datei das Netzwerk repräsentiert, in dem jede einzelne Kante mit Anfangs- und Endpunkt definiert ist. Bei dem in Abbildung 31 gezeigten Abschnitt ist die erste Kante die Verlinkung von dem *10th_century* zu dem Artikel *9th_century*. Diese Struktur muss nun an das Framework angepasst werden, damit dieses den Graph einlesen und damit arbeiten kann.

```
10th_century 9th_century
10th_century 11th_century
10th_century Time
10th_century Dark_Ages
10th_century India
10th_century Viking
11th_century 10th_century
11th_century 12th_century
11th_century Time
11th_century Dark_Ages
11th_century Crusades
11th_century Byzantine_Empire
```

Abbildung 34: Kanten der Graphen des Wikipedia for School

Damit das Framework es einlesen kann, muss eine Struktur mit folgendem System entstehen:

```
4626 0
4627 1
4628 2
4629 1
4630 0
4651 3
0 0
1 0
```

Abbildung 35: Benötigtes Eingabeformat

Des Weiteren sind die Kategorien der einzelnen Artikel ebenfalls nicht passend konvertiert und müssen genau zu den Knoten und ihren Identifikationswerten definiert werden. Die Schreibweise der Kategorien ist in Abbildung 33 zu sehen.

Man kann auch erkennen, dass hier nicht nur die Topkategorie angegeben ist, sondern weitere Kategorien, die auf den Artikel zutreffen. Für dieses in der Arbeit verwendete Verfahren werden jedoch nur die Topkategorien ausgelesen.

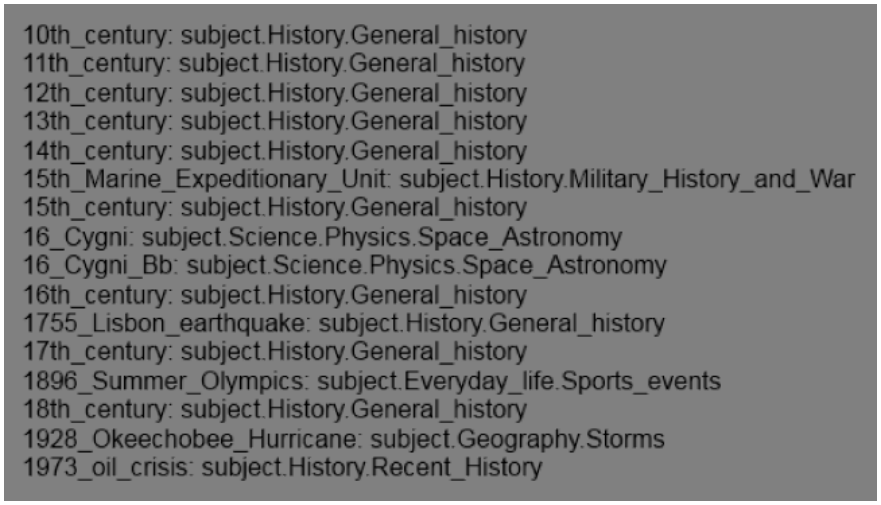


Abbildung 36: Originales Datenformat der Kategorien des "Wikipedia for Schools"

Werden die Kanten und die dazugehörigen Kategorien eingelesen, resultieren daraus zwei Dateien. Die wichtigere Datei wird in Abbildungen 35 dargestellt. Hier handelt es sich zum Beispiel bei *10th_century* um den Knoten mit der Id 4626, welcher die Kategorie "0" besitzt. Kategorien werden so dargestellt, dass zuerst die KnotenId definiert ist und danach die Id der Kategorie. Für die Kategorien existiert natürlich auch eine Datei in der die Id's auf die Kategorien gemappet wird.

CountCategoryClicks:

Mit Hilfe dieses Programms werden die einzelnen Pfade der User durchgegangen und von jedem besuchten Knoten die Kategorien mitgezählt. Aus dieser Statistik lässt sich nun ablesen, welche Kategorien wie oft gewählt werden. Des Weiteren wird berechnet, wie hoch der prozentuale Anteil jeder Kategorie über alle User ist.

Das Programm liest die Pfade jedes Users ein (siehe Abbildung 35) und sucht die passende Kategorie Id aus der Kategoriedatei heraus.

```
2610 1510 2610 r  1168 r  176
2815 1510 2815 r  2756 u  2233 u  1510
4173 1510 4173 r  4517 r  1989 g  2755 r  2164 r  1079 u  1510
4314 1510 4314 r  3195 r  2733 r  2150 r  3880
906  1510 906  g  2460 r  2688 r  1377 r  1545 r  314  u  2755
3778 1510 3778 u  4418 r  1374 r  1789 r  3458 r  1079 u 1510
```

Abbildung 37: Datenformat der Pfade für die Generierung der Wahrscheinlichkeiten

Es wird zum einen die Datei mit den Pfaden benötig und zum anderen die Datei, die die Knoten-Identifikationsnummer und die dazugehörige Kategorie beinhaltet. Die Kategorie wird hier ebenfalls nur als Identifikationsnummer dargestellt.

Der Output dieses kurzen Programms liefert eine kurze Tabelle der Wahrscheinlichkeitsverteilung, die für das Erstellen der Wissensmodelle nötig ist. Diese wird in Abbildung 35 dargestellt.

```
15  0.00813353
12  0.68236
14  0.742719
8   1.25856
3   1.6815
9   2.10059
7   2.27396
11  3.26925
10  3.78894
2   3.83003
13  6.22044
5   7.09843
0   9.32659
1   11.9905
4   21.877
6   23.8509
```

Abbildung 38: Wahrscheinlichkeitsverteilung der Kategorien in Prozent

4.1.5.2 Wikipedia

Das Wikipedia Netzwerk ist der zweite Datensatz, der für dieses Projekt zur Testung in Frage kommen könnte. Es ist im Gegensatz zu dem "Wikipedia for Schools" ein sehr großes Netzwerk mit einer wesentlich höheren Anzahl an Kanten, aus denen auch der immense Rechenaufwand für diesen Graphen resultiert. In Zahlen gesprochen handelt es sich um ca. 9,3 Millionen Knoten im Graphen, die miteinander verbunden sein können.

Abbildung 39: Wikipedia [28]

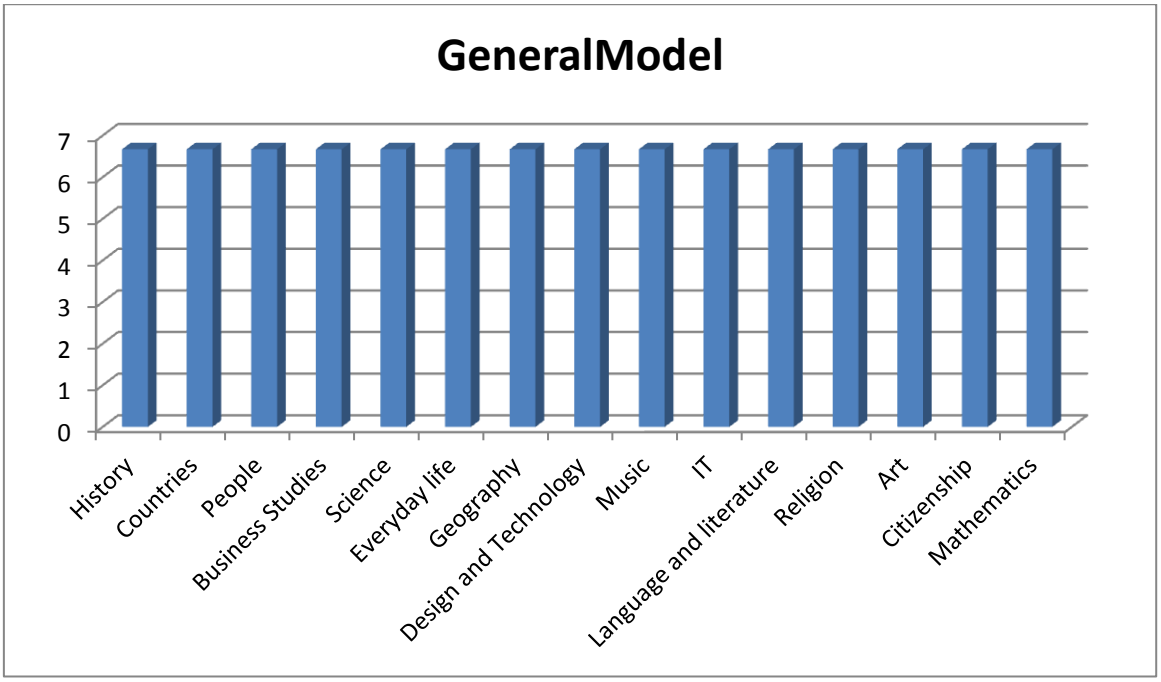
Verbunden sind diese Knoten über 255 Millionen Kanten. Insgesamt bildet dieser Graph mit den Userdaten eine solide Grundlage für die Testung, doch fehlen hier noch die Kategorien zu den jeweiligen Artikeln.

4.1.6 Wissensmodelle

In diesem Kapitel werden die Wissensmodelle im Detail erklärt, die für die Navigation von Start- zu Zielknoten so wichtig sind. Das Wissensmodell gibt an, wie das Wissen der User auf die einzelnen Kategorien verteilt ist und spielt somit die zentrale Rolle für die Entscheidung, welcher Knoten bzw. Artikel als nächstes ausgewählt wird. Da es sich um ein Verfahren handelt, in dem noch nicht viel geforscht wurde, ist es schwer zu sagen, welches die richtige Definition eines solchen Modells ist, um die Allgemeinheit der User so gut wie möglich zu repräsentieren. Aufgrund dieses Problems wurde der Algorithmus mit vier verschiedenen Modellen getestet, die alle ihre speziellen Eigenschaften haben. Dazu gehören das GeneralModel, LessGeneralModel, MoreSpecificModel, MostSpecificModel, deren Konfiguration im nächsten Abschnitt beschrieben und die Herleitung erläutert wird.

4.1.6.1 GeneralModel

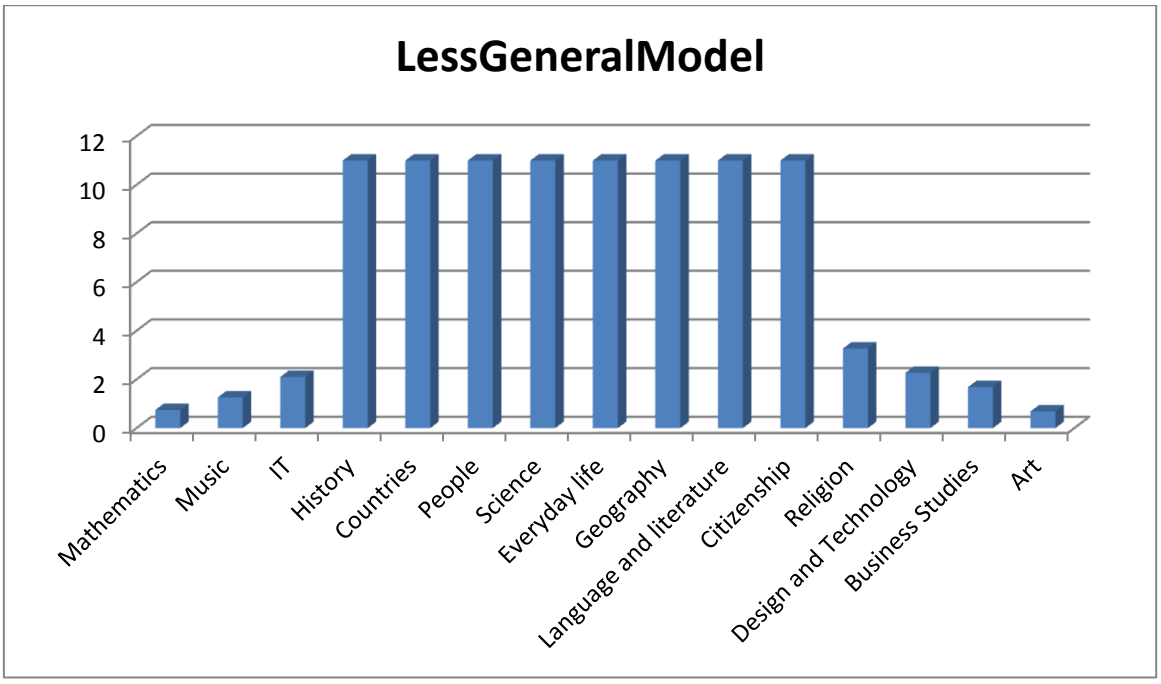
Bei diesem Model handelt es sich um das allgemeinste Modell, welches einen User repräsentieren soll, der sich in allen Bereichen gleich gut auskennt und somit ein sehr gutes Allgemeinwissen besitzt. Um die Wahrscheinlichkeiten zu definieren, wird abhängig von der Anzahl der Kategorien der Durchschnitt berechnet.



Dieser beträgt bei dem verwendeten Datensatz des Schülerwikis 15 Kategorien. Daraus resultiert eine durchschnittliche Wahrscheinlichkeit von 6,67%.

4.1.6.2 LessGeneralModel

Für dieses Modell wurde eine etwas genauere Beschreibung herangezogen, bei der sich die Spezialisierung auf die Hälfte der gegebenen Kategorien beschränkt. Hierfür werden nicht einfach zufällig Kategorien ausgewählt, die die höchsten Wahrscheinlichkeiten repräsentieren, sondern es wird anhand der gesammelten Userdaten berechnet, welche Kategorien am häufigsten angewählt wurden und daraus eine Verteilung gebildet.

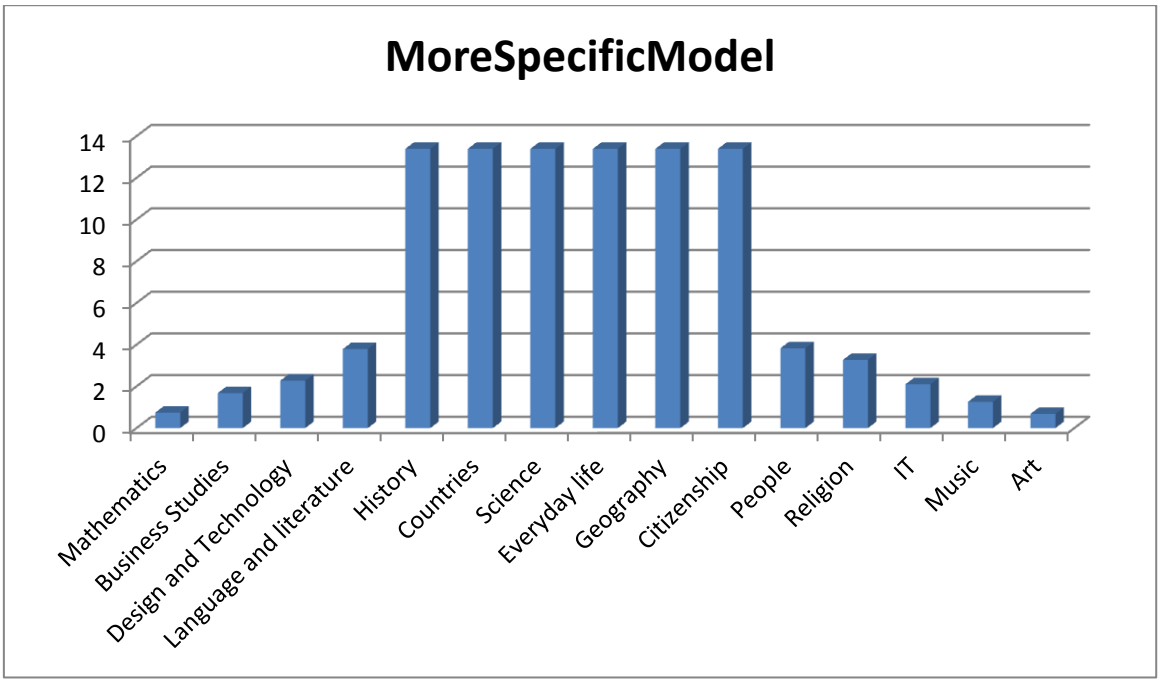


Anhand dieser Verteilung werden die höchsten Werte aufsummiert und durch deren Anzahl dividiert. Somit erhalten alle Topkategorien den gleichen Wert (siehe LessGeneralModel Abbildung).

4.1.6.3 MoreSpecificModel

Bei diesem Modell verhält es sich ähnlich dem LessGeneralModel, da man sich hier ebenfalls auf bestimmte Kategorien spezialisiert und sich somit vom dem allgemeinen Modell immer weiter entfernt.

Es wird sich im Gegensatz zu dem LessGerneralModel nicht auf die Hälfte der gegebenen Kategorien spezialisiert, sondern ein geringerer Teil herangezogen. Dieses Verfahren soll bezwecken, dass sich ein geringerer Teil der Kategorien auf einen höheren Teil der Wahrscheinlichkeit verteilt.



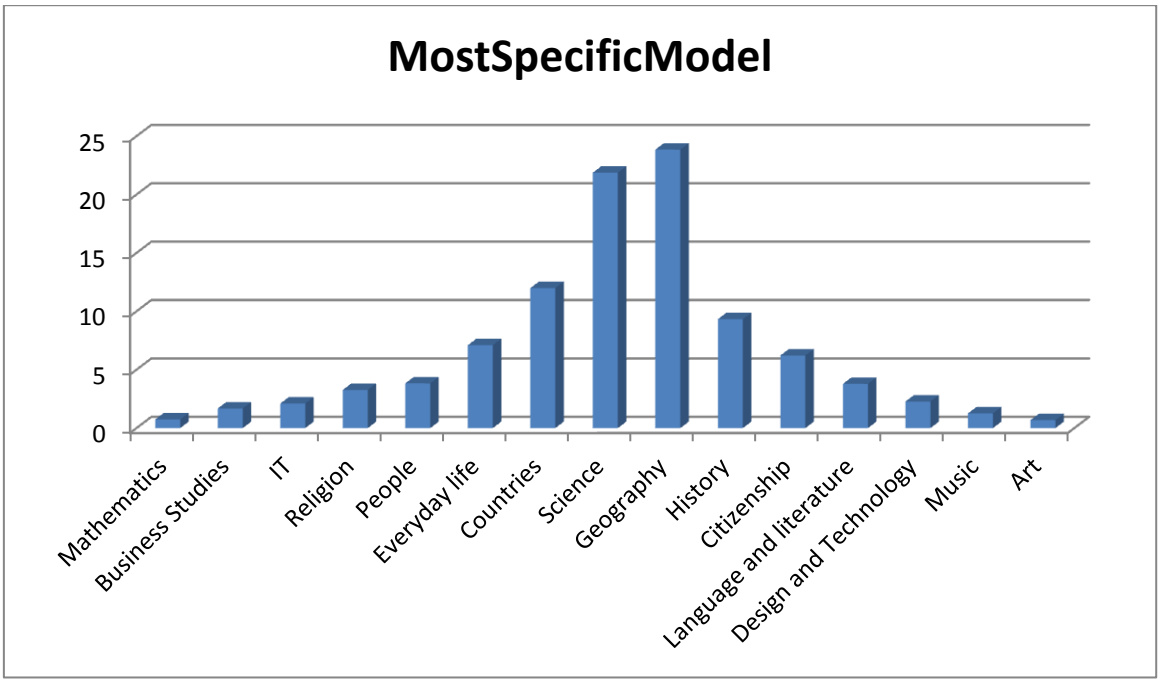
Diese Wahrscheinlichkeitsverteilung wird genauso berechnet, wie im vorherigen Modell, nur wird der Durchschnitt von $^1/_3$ der gesamten Kategorien berechnet. Wie man anhand der Testdaten sehen kann, steigt die Wahrscheinlichkeit der Topkategorien um einen signifikanten Wert.

4.1.6.4 MostSpecificModel

Hierbei handelt es sich um das detaillierteste Modell dieser Testung und es wird versucht, den User anhand seiner genauen Daten aus dem Spiel zu repräsentieren. Alle anderen Modelle verwenden ebenfalls die aus diesem Modell resultierenden Werte.

Um die Werte für dieses Modell zu setzen, benötigt man ein Zusatzprogramm, das im vorherigen Abschnitt schon beschrieben wurde und dieses liefert die Wahrscheinlichkeitsverteilung der Kategorien.

Es zeigt sich, dass die Wissensverteilung auf den Kategorien Geographie und Wissenschaft verteilt ist und die Artikel dieser Kategorien bei der Suche nach dem Zielknoten am wahrscheinlichsten gewählt werden.


MostSpecificModel
25
20
15
10
5
0
Mathematics
Business Studies
IT
Religion
People
Everyday life
Countries
Science
Geography
History
Citizenship
Language and literature
Design and Technology
Music
Art

5 MUN

Bei dem sogenannten **M**odeling **U**ser **N**avigation, kurz NUM, handelt es sich um ein Framework, das von M. Eder [20] entwickelt wurde und es ermöglicht eine verschiedene Navigationsarten durch Informationsnetzwerke zu simulieren. Zu den verschieden Arten gehören unter anderem der sogenannte Greedy-, Teleport- oder Randomalgorithmus. Des Weiteren beinhaltet dieses Framework eine von der Standfort University entwickelten Bibliothek mit dem Namen „SNAP“ (Stanford Network Analysis Platform). Diese wird in diesem Kapitel auch kurz beschrieben und im Allgemeinen für die schnelle, genaue Bearbeitung und Analyse von sehr großen Netzwerken verwendet [21]. Diese Library bildet die Grundlage für MUN und ermöglich erst eine solide Navigation innerhalb solcher Netzwerke.

In diese Kapitel wird die Funktionalität, der Ablauf innerhalb des Frameworks und die Struktur beschrieben, damit ein Verständnis für Verfahrensweise der Simulation der Navigation im späteren Verfahren gegeben ist.

5.1 SNAP

Bei der Stanfort Netzwerk Analyse Plattform handelt es sich um ein Projekt der Universität in Stanford, die seit ca. 10 Jahren an diesem arbeitet und es stetig weiterentwickelt und vergrößert. Es dient der Manipulation und Analyse von großen Netzwerken oder Graphen. Außerdem besitzt es eine hohe performante Laufzeit. Es beschäftigen sich aktuell 13 Mitarbeiter mit diesem Framework. Sie verwenden mehrere große Datensätze für ihr Projekt zur Navigation in Informationsnetzwerken, bei denen das Größte einen Graph aus 240 Millionen Knoten mir 1,3 Milliarden Kanten enthält. Hierbei handelt es sich um das „Microsoft Instant Messenger“-Netzwerk bei dem man sieht, wie die Skalierbarkeit der Größe eines Netzwerkes garantiert wird.

Die hohe Geschwindigkeit der Programms lässt sich unter anderem auf die verwendete Programmiersprach C++ zurückführen, die die Repräsentation der Graphen optimiert. Des Weiteren können den einzelnen Knoten oder Kanten innerhalb des Graphen bestimmte Attribute zugewiesen werden. Ebenfalls könnten während der Laufzeit Graphen umstrukturiert und somit verändert werden [22].

5.1.1 SNAP Funktionen

Um einen kurzen Einblick in die Library zu geben, wird in den folgenden Abschnitten eine Auflistung wichtiger Funktionen und Datentypen gezeigt, die einem ein Gefühl für den Umgang mit SNAP vermitteln sollen. Es wird beschrieben, wie man einen Graphen jeglicher Art erstellt, welche Netzwerktypen definiert werden können und es wird eine Sammlung von Funktionen dargestellt.

5.1.2 Graphen Definition

Wichtig bei der Definition eines solchen Graphen ist, dass hier die Kanten bei, zum Beispiel einem ungerichteten Graphen, einzeln hinzugefügt werden müssen. Das heißt, gibt es einen Pfad in beide Richtungen, muss einmal die Verbindung von Knoten A nach Knoten B hinzugefügt werden, aber auch die von Knoten B nach Knoten A.

- **TUNGraph**: Erstellt eine Variable, die einen ungerichteten Graphen repräsentiert und Kanten, die immer in beide Richtungen von und zu einem Knoten laufen
- **TNGraph**: Erstellt eine Variable, die einen gerichteten Graphen repräsentiert und Kanten, die eventuell nur in eine Richtung zu einer Verbindung führen
- **TNEGraph**: Erstellt eine Variable, die die einen gerichteten, multiplen Graphen repräsentiert, bei dem Knoten ein vielfaches an Kanten zwischen den Knoten besitzen können

5.1.3 Netzwerktype

Zu den oben erläuterten Definitionen einer Graphenvariable können auch bestimmte Datentypen als Knoten verwendet werden, nicht nur wie im oberen Abschnitt mit Zahlen als Knotenrepräsentation, sondern mit zum Beispiel TNodeData.

- **TNodeNet<TNodeData>:** Der gleiche Datentyp wie *TNGraph*, nur werden hier *TnodeData* als Knoten verwendet
- **TNodeEDatNet<TNodeData, TEdgeData>:** Der gleiche Datentyp wie *TNGraph*, nur werden hier *TnodeData* als Knoten und *TEdgeData* als Kanten verwendet

- **TNodeEdgeNet<TNodeData, TEdgeData>:** Wie bei *TNEGraph*, nur werden hier *TnodeData* als Knoten und *TEdgeData* als Kanten verwendet
- **TBigNet<TNodeData>:** Eine sehr speichereffiziente Implementierung der ersten Definition *TnodeNet*, welche Millionen von Kanten (abhängig von dem RAM) verarbeiten kann und vermeidet die Fragmentierung des Speichers.

5.1.4 Zusätzliche Funktionen

Zuletzt noch einige weitere Funktionen, die relevant sind und im später erklärten Verfahren verwendet wurden. Die nächsten Beispiele dienen dem Durchlaufen der Knoten anhand von Iteratoren:

- **BegNI()**: Iterartor, der auf den ersten Knoten zeigt.
- **EndNI()**: Iterator, der auf den letzten Knoten zeigt.
- **GetNI(u)**: Iterator, der auf den Knoten mit der ID „u“ zeigt .
- **BegEI()**: Iterator, der auf die erste Kante zeigt.
- **EndEI()**: Iterator, der auf die letzte Kante zeigt.

Folgende Funktionalität dient dem Auslesen von Daten des Netzwerkes oder bestimmter Knoten innerhalb des Graphens:

- **GetEI(u,v)**: Retuniert die Kanten zwischen Knoten „u“ und Knoten „v“.
- **GetEI(u)**: Gibt die Kante mit der ID „u“ zurück, ist aber nur für multiple Graphen möglich.
- **GetId()**: Liefert die ID des Knoten.
- **GetOutDeg()**: Retuniert den Grad eines Knoten (nur die Anzahl der Kanten, die von diesem Knoten weggehen).
- **GetDat()**: Liefert die *TNodeData*.

5.2 Framework

Da nun die Kernbibliothek beschrieben wurde, kann nun das eigentliche Framework im Detail erklärt werden. Wie oben in der Einleitung angesprochen, handelt es sich hier um ein Programm, das es möglich macht, mit verschiedenen Algorithmen ein Informationsnetzwerk zu simulieren. Es handelt sich hier um einen komplexen Ablauf innerhalb des Systems und wird in zwei große Teile aufgeteilt. Dabei geht es zum einen um den Framework Kern, zu dem auch die SNAP Library gehört und zum anderen der Navigationsbereich, der die verschiedenen Navigationsmöglichkeiten und unterschiedlichen Algorithmen zur Knotenselektion beinhaltet. Im letzteren Abschnitt befindet sich auch das entwickelte Verfahren.

5.2.1 Core

- *Environment*
  In der Environment-Klasse werden die, an das Programm übergebenen Parameter behandelt und entsprechend gesetzt. Diese Variablen konfigurieren die Art und Weise, wie das Netzwerk durchlaufen wird und um welchen Netzwerktyp es sich handelt. Die dafür vorhandende Config Datei wird später noch beschrieben.

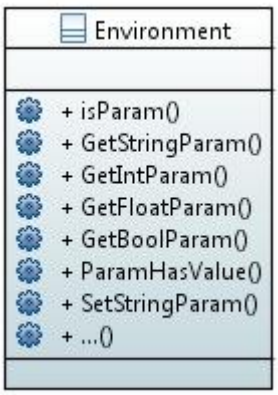


Abbildung 40: Environment-Klasse

- *Graph*
  Hierbei handelt es sich um eine abstrakte Klasse, von der die zwei Graphentypen abgeleitet werden. Zum einen der gerichtete (GraphD) und zum anderen der ungerichtete Graph (GraphU). Es werden alle Funktionen zur Manipulation und Analyse des Graphen bereitgestellt. Dazu gehören zum Beispiel die Getter und Setter Methoden der Knoten, die der Graph besitzt, das Hinzufügen oder Löschen einzelner Kanten.

- *GraphU*
  Repräsentiert einen ungerichteten Graphen, der von der Klasse *Graph* abgeleitet wird und basiert auf dem Datentyp *TUNGraph* und *PUNGraph* eines ungerichteten Graphen aus der SNAP Library.

- *GraphD*
  Entspricht einem gerichtetem Graphen, der von der *Graph* Klasse abgeleitet wird und wie *GraphU* auf dem Datentyp *TNGraph* und *PNGraph* eines gerichteten Graphen aus der SNAP Library basiert.

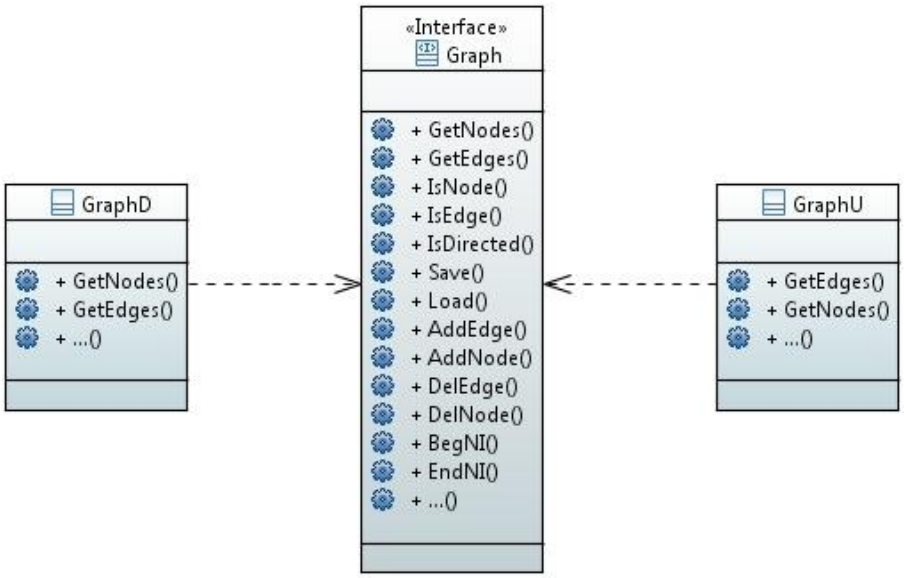


Abbildung 41: Graph Interface mit abgeleitetem GraphU und GraphD Klasse

- *Nodel*
  Nodel bildet die abstrakte Oberklasse von *NodelD* und *NodelU* und repräsentiert den Knoteniterator für beide Arten von Graphen. Des Weiteren liefert ein solcher Iterator die benötigte Funktionalität für das Durchlaufen der Knoten des Graphen. Dazu gehören Funktionen, die das Auslesen von Informationen der einzelnen Knoten möglich machen. Beispiele für solche Informationen wären die ID oder der Grad (sowohl ausgehender und hineingehender) des Knotens. Außerdem lassen sich Eigenschaften und Informationen über Eingangsknoten, Ausgangsknoten und Nachbarknoten auslesen.

- *NodelD / NodelU*
  NodelD und *NodelU* sind die Knoteniteratoren für gerichtete und ungerichtete Graphen und sind von der abstrakten Klasse *Nodel* abgeleitet.

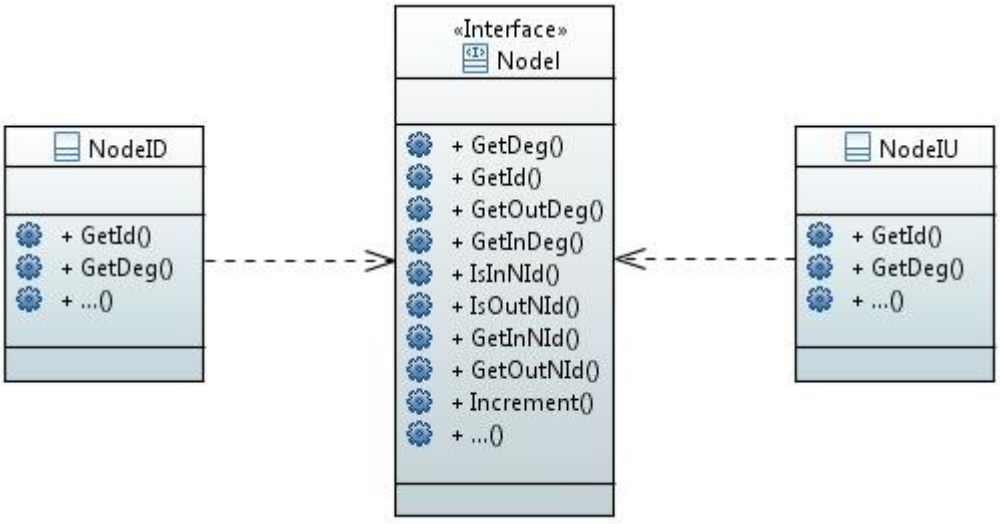


Abbildung 42: Klasse *NodelU* und *NodelD* abgeleitet von der abstrakten Klasse *Nodel*

- *Pairs*
  Bei dieser Klasse handelt es sich um den Datentyp, der einen Start- und Zielknoten beinhaltet, zwischen denen navigiert werden soll. Zum einen können diese Paare aus einer Datei eingelesen werden oder zum anderen zufällig vom Framework generiert werden. Bei der in dieser Arbeit beschriebenen Aufgabenstellung werden die Paare aus dem sogenannten "Wikigame" [26] eingelesen und zum Vergleich für den Algorithmus herangezogen.

- *Path*
  Die Klasse *Path* repräsentiert einen kompletten Pfad durch das Netzwerk von einem bestimmten Start- zu einem Zielknoten. Dieser beinhaltet Informationen über die einzelnen Knoten, aber auch Informationen über den Pfad an sich, also auch wie lang dieser ist.

- *PathCollection*
  Bei einem Objekt des Types *PathCollection* handelt es sich um eine Sammlung von Objekten der Klasse *Path*. Es lassen sich Pfade hinzufügen und wichtige Informationen über diese Sammlung auslesen. Ebenfalls sind Funktionen bereitgestellt, die es ermöglichen, diese Pfade in einer Datei zu speichern und für andere Projekte weiter zu verwenden.

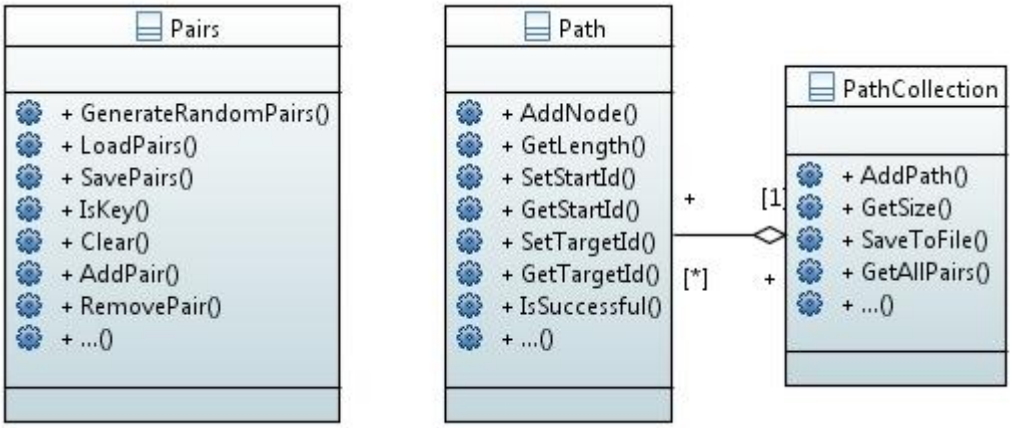


Abbildung 43: Pairs-, Path- und PathCollection-Klasse

5.2.2 Navigation

Nachdem nun grundlegende Parameter wie der Datentyp des Graphen, die darin enthaltenen Knoten, Pfade und deren Iteratoren beschrieben wurden, wird in diesem Kapitel nun die Navigation innerhalb des Netzwerkes beschrieben. Der Navigator dient der simulierten Pfadgenerierung anhand unterschiedlicher Algorithmen, die der des Menschen gleichen sollen.

Um eine Navigation zu starten, wird ein *Environment* Objekt mit den beinhalteten Parametern aus der Config-File an diese Klasse übergeben. Anhand der Konfigurationen wird eine Navigations-Strategie festgelegt, zu der zum Beispiel der Randomwalk oder der Greedyselector gehören (siehe Abbildung 27). Ist die Strategie gesetzt, kann nun mit der Generierung von Pfaden von einem Start- zu einem Zielknoten begonnen werden. Start- und Zielknoten können anhand des oben gewählten Algorithmus berechnet werden. Es resultiert eine hohe Anzahl an Pfaden, die in einer *PathCollection* zurückgeliefert werden. Diese Pfade können nun mit den gesammelten Daten der User verglichen werden.

Des Weiteren ist es möglich, mehrere Strategien miteinander zu kombinieren und einfach weitere Strategien hinzuzufügen. Also kann zum Beispiel der Randomwalk ausgeführt werden und ebenfalls ein Greedywalk. Eine andere Möglichkeit wäre einfach per Zufall eine Kombination auszuwählen. Die Navigations-Strategien werden über eine *NavigationStrategyFactory* erzeugt (siehe Abbildung 28). So lassen sich nachträglich auch weitere neue Navigationsstrategien erstellen und einbinden.

Im nächsten Bereich wird der für diese Arbeit entwickelte Algorithmus eingebettet. Ein wichtiger Teil der Generierung der Pfade ist die Entscheidung welcher Knoten als nächstes gewählt wird. Um diese zu bestimmen, kommt die Klasse der *NodeSelectors* ins Spiel:

- *INodeSelector*
  Diese Klasse dient den von der Knotenselektionsstrategie ausgewählten Knoten zu bestimmen und zu übergeben.

- *NSBaseSelector*
  Der *NSBaseSelector* bildet die abstrakte Oberklasse für die verschiedenen Knotenauswahlstrategien, wie zum Beispiel NSRandom, NSGreedy oder NSProbability und liefert die benötigten Funktionen.

- *NSRandom*
  Diese Strategie wählt zufällig einen Knoten aus der Liste der Nachbarknoten aus.

- *NSGreedy*
  Es wird der Knoten ausgewählt, der zu dem Zeitpunkt der Selektion den kürzesten Pfad zum Zielknoten darstellt.

- *NSProbability*
  Es wird anhand einer Wissensverteilung entschieden, welcher der Nachbarknoten der Wahrscheinlichste ist, den ein Mensch auswählen würde.

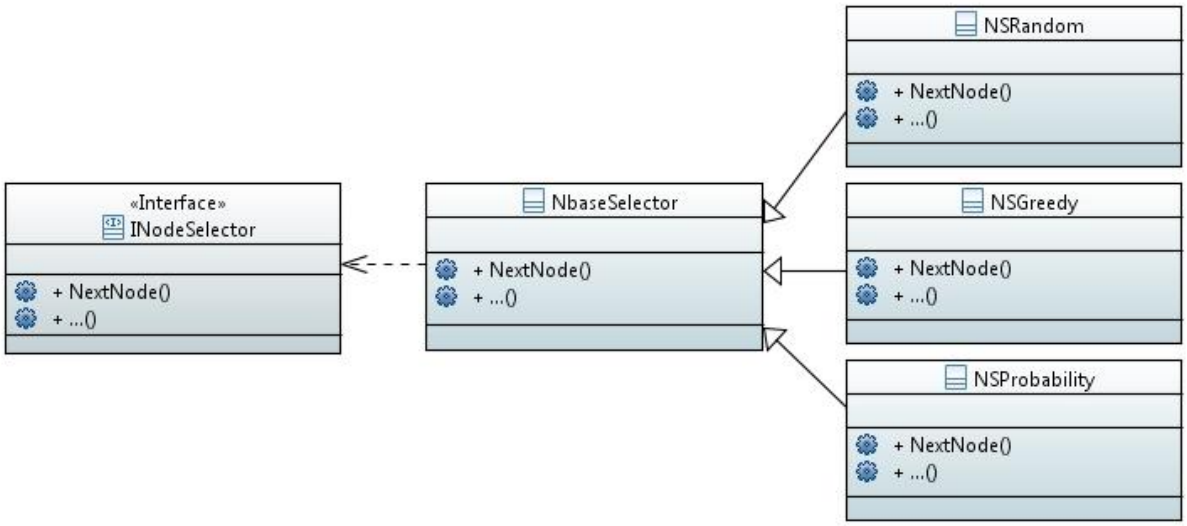


Abbildung 44: NodeSelector Klassen mit abgeleiteten NSRandom, NSGreedy und NSProbability Klasse

- *INavigationStrategy*
  Definiert die Strategie, die für die Navigation ausgewählt wurde. Zu den Strategien gehört unter anderem die zufällige oder kombinierte Navigation.

- *SBaseStrategy*
  Beschreibt die Oberklasse der unterschiedlichen Auswahlverfahren für den nächsten Knoten, der dem Pfad hinzugefügt wird.

- *SRandomNavigator*
  Durch diese Klasse wird zufällig eine Strategie für die Knotenseletkion ausgewählt und an den *INodeSelector* weitergegeben.

- *SCombineNavigator*
  Bei dem *SCombineNavigator* können mehrere Strategien zusammen verwendet werden und nacheinander ausgeführt werden.

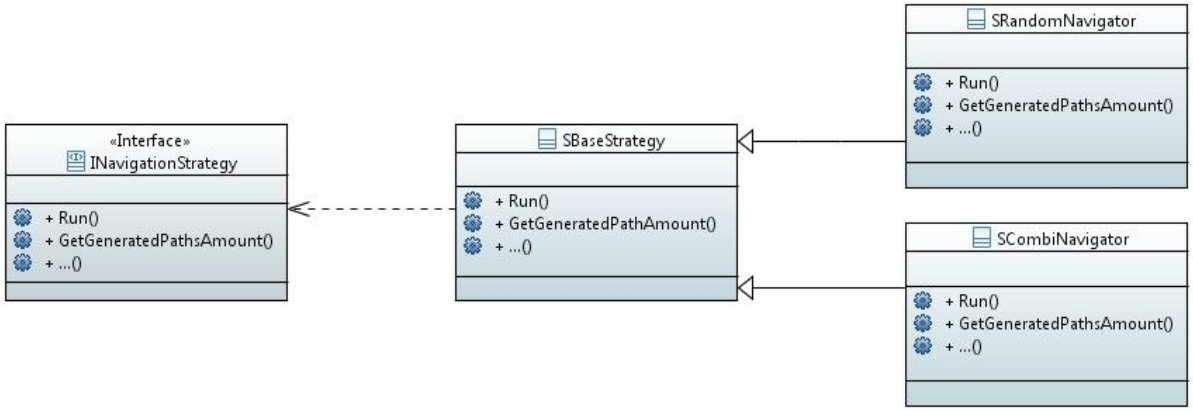


Abbildung 45: INavigationStrategy

5.2.3 Config

Da nun die grundlegende Funktionsweise der MUN Frameworks bekannt ist, werden in diesem Abschnitt die benötigten Parameter für die Konfiguration gelistet und der jeweilige Befehl im Detail erläutert. Hier werden nicht nur die Werte für den einfachen Durchlauf des Programms beschrieben, sondern auch die Einstellung für die hier entwickelten Knotenselektoren *NSProbability*.

Folgende Parameter müssen gesetzt werden:

- *Starten des Programms*

  Das Framework wird in der Kommandozeile mit dem Befehl „./munFramework“ gestartet und es können Parameter aus der Config-File mit übergeben werden. Möchte man zum Beispiel den Pfad zur Config-Datei mit angeben, geschieht dies über „--config: PFAD ZUR DATEI“

- *global.amount*
  Gibt dem maximalen Wert an generierten und navigierten Paaren für das MUN an. Ist der Wert auf 0 gesetzt, ist keine Obergrenze gesetzt und es werden alle Paare verwendet.

- *global.output*
  Gibt den Pfad zu dem Ordner an, in dem die Output-Dateien gespeichert werden sollen. Hier ist wichtig darauf zu achten, dass am Schluss des Befehls das Backslash vorhanden ist. Ein Beispiel dafür wäre: *global.output:test/output/*

- *graph.file*
  Definiert den Pfad zur Datei, die den Graphen enthält

- *graph.type*
  Gibt dem Framework vor, um welche Art von Graphen es sich bei der der eingelesenen Datei handelt. Es kann sich hier entweder um einen gerichteten („*d*“) oder ungerichteten („*u*“) Graphen handeln. Ein Beispiel wäre: *graph.type:u*

- *combi*
  Dieser Parameter bestimmt die Einstellungen für die Navigation. Es können hier mehrere Navigatoren mit unterschiedlichen Eigenschaften definiert werden, die nacheinander ausgeführt werden. Jeder einzelne Navigator repräsentiert eine bestimmte Art von Knotenselektor wie zum Beispiel „greedy“ oder „teleport“. Würde man als Beispiel eine Greedy-Navigation in Kombination mit Teleport starten wollen, würde das folgendermaßen aussehen:

  combi.ns0:teleport
  combi.ns0.tobetter:T
  combi.ns1.greedy
  combi.ns1.useamount:10

- *navigation.strategy*
  Setzt die Art der Navigation fest, die für die Knotenselektion verwendet wird. Mögliche Variablen wären hier *greedy*, *teleport*, *probability* oder *random*.

- *navigation.maxhops*
  Definiert die Anzahl der maximalen Sprünge, die von einem Start- zu einem Endknoten benötigt werden dürfen.

- *navigation.revisits*
  Gibt die Anzahl der Besuche an, die bei einem Pfad pro Knoten auftreten können.

- *paths.comparewith*
  Um die generierten Pfade mit den gesammelten Userdaten zu vergleichen, wird dieser Parameter verwendet und liefert eine Statistik über das Ergebnis. Wird dieser Parameter nicht gesetzt, wird auch kein Vergleich durchgeführt.

Die oben beschriebenen Parameter bestimmen den allgemeinen Ablauf. Jetzt fehlen noch die Variablen für den *NSProbability* Knotenselektor:

- *category.path*
  Gibt den Dateipfad zu der benötigten Kategorie Liste an, in der die Sammlung der Knoten ID`s in Verbindung mit den Kategorie ID`s stehen. Also würde im Falle des Wikipedias zum Beispiel ein gültiger Eintrag „4 5“ heißen, was bedeutet, dass der Knoten (in dem Fall der Artikel) mit der ID 4 der Kategorie 5 zugewiesen ist. Ein Beispiel für diesen Befehl sieht folgendermaßen aus:
  category.path:data/graph_category.txt

- *probabilityfilepath*
  Gibt den Dateipfad zu der Liste der Wahrscheinlichkeiten für jede Kategorie an. Diese enthält jede einzelne Wahrscheinlichkeit für die unterschiedlichen Kategorien, die es im Wikipedia gibt. Der Aufruf sieht folgendermaßen aus: *probabilityfilepath:data/graph_proberbilities.txt*

6 Experiment

In diesem Kapitel werden das Experiment und die Resultate diskutiert. Es werden nochmals die Wissensmodelle und ihre Verteilungen gezeigt. Darauf folgen die Ergebnisse, welche miteinander verglichen werden. Dieser Vergleich wird zum besseren Verständnis in einer übersichtlichen Statistik dargestellt.

Nachdem die Ergebnisse untereinander verglichen wurden, werden die Resultate auf die vorhandenen Algorithmen bezogen und aus diesem Vergleich wichtige Schlussfolgerungen geschlossen.

6.1 Analyse der Modellergebnisse

Nun werden die Resultate der insgesamt acht getesteten Wissensmodelle gezeigt. Zu den Modellen gehören das General-, Less General-, More Specific- und Most Specific Model. Diese wurden im vorherigen Kapitel schon erläutert. Um noch besser zeigen zu können, welche Auswirkungen ein gut bzw. schlecht gewähltes Wissensmodell hat, wurden noch zwei weitere Modelle hinzugefügt. Dabei handelt es sich zum einen um eine Usergruppe, die sich nur gut in Musik auskennt und zum anderen um eine, die ein großes Wissen in Geographie besitzt. Diese werden nun in Gruppen zueinander verglichen.

6.1.1 General-, Less General-, More Specific- und Most Specific Model

Zuerst werden die grundlegenden Wissensmodelle analysiert, die im vorherigen Kapitel schon beschrieben wurden. Diese werden in Abbildung 45 nochmals gezeigt. Hier soll die jeweilige Verteilung in einem kleinen Überblick dargestellt werden. Detaillierte Ansichten dieser wurden im Kapitel "4.1.6 Wissensmodelle" gezeigt.

Die in Abbildung 46 gezeigten Statistiken zeigen nun die jeweiligen Resultate, die die Wissensmodelle geliefert haben. Dort werden die Anzahl der gleichen Entscheidungen mit den Userdaten gezeigt. Auf der Y-Achse sieht man die genaue Anzahl der Treffer und auf der X-Achse den dazugehörigen Schritt im Pfad.

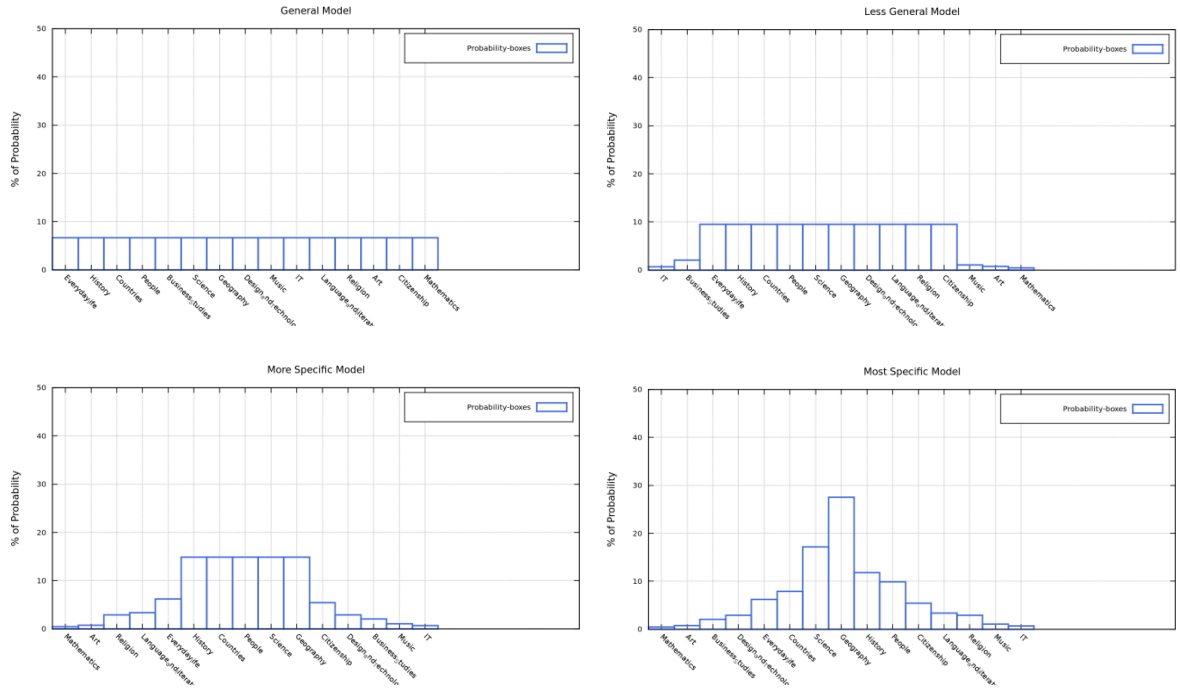


Abbildung 46: Wissensverteilung eines Users der Allgemeinwissen, weniger allgemeines Wissen, mehr spezielles Wissen und fast nur spezielles Wissen besitzt. (von oben links nach unten rechts)

General Model:

Bei einer Gleichverteilung des Wissens wird eine Trefferquote von 24,5902% erreicht. Das entspricht genau 54252 Knoten, die richtig entschieden wurden. Damit liefert dieses Wissensmodell ein vergleichsmäßig gutes Ergebnis.

Less General Model:

Verwendet man nun eine etwas weniger allgemeine Verteilung als Wissensmodell und spezialisiert sich etwas mehr auf häufig vorkommende Kategorien, so erreicht man eine Quote von 24,4876%, was einer Trefferzahl von 53664 Knoten im Vergleich zu den Usern entspricht. Somit liefert diese Verteilung durchschnittliche Ergebnisse, wenn man diese in Relation zueinander betrachtet.

More Specific Model:

Spezialisiert man sich noch weiter auf die am meisten vorkommen Kategorien und verteilt diese entsprechend auf das Wissensmodell, erreicht man das beste Ergebnis unter den sechs verschiedenen Wissensmodellen. Hierbei hat man eine Übereinstimmung von 24,7335%, was einer Anzahl von genau 54203 Knoten innerhalb der Pfade entspricht.

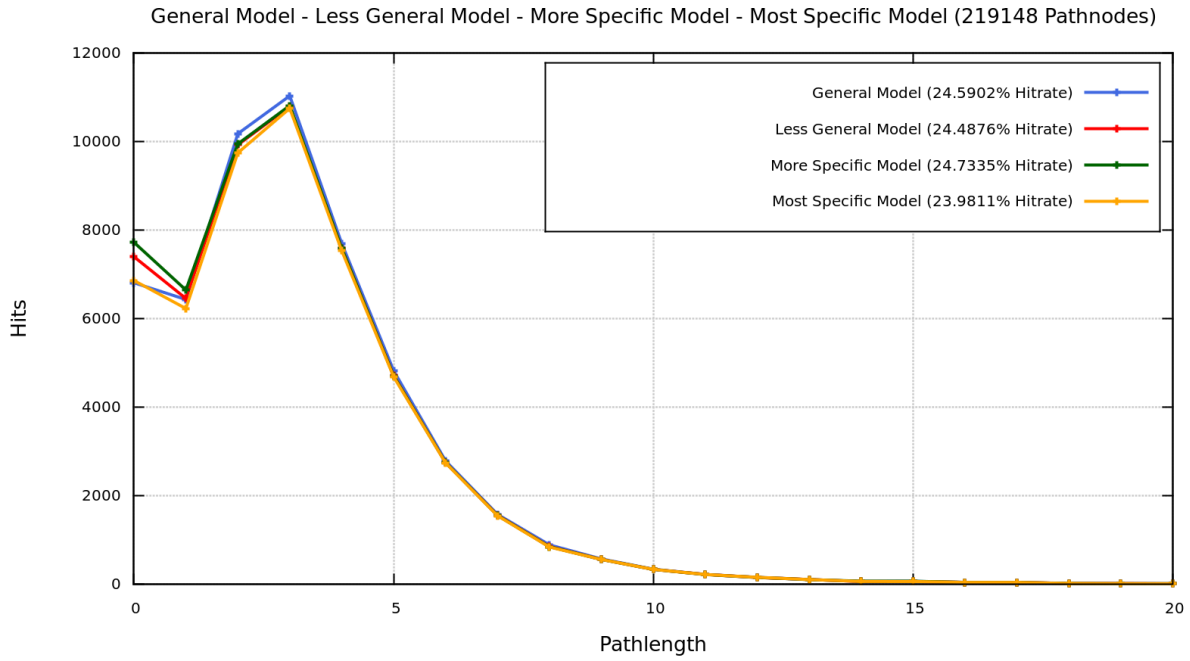


Abbildung 47: Resultate der General-, Less General-, More Specific- und Most Specific Models in Hits

Most Specific Model:

Nun definiert man das Wissensmodell so, dass es sozusagen einen Spezialisten in einem Fachgebiet repräsentiert und dieser sich zusätzlich in anderen Kategorien zu einem bestimmten Prozentsatz auskennt. Dieses Wissensmodell liefert die schlechtesten Resultate in diesem Vergleich. Im Detail hat dieses Modell eine Trefferquote von 23,9811% was einer Knotenanzahl von 52554 entspricht, also ein vergleichsweise schlechtes Ergebnis.

Zusammenfassend ist hier zu sagen, dass das "More Specific Model" die besten Resultate liefert, da es im Allgemeinen die wichtigsten Bereiche des Netzwerkes abdeckt, die die User bei ihrer Navigation verwenden, aber im Gegensatz zu dem "Most Specific Model" nicht zu sehr auf eine Kategorie spezialisiert ist und somit eine höhere Erfolgsrate hat. Das "Less General Model" hat hier eine allgemeinere Verteilung, die zwar für Usergruppen und ein Netzwerk funktionieren würde, die gleichverteilter sind, als die vorhandenen Testdaten, aber liefern für diese Arbeitsumgebung im Vergleich zu den anderen Modellen keine ausreichenden Ergebnisse. Das "General Model" schneidet relativ gut ab, da es ein Art globales Wissen im Netzwerk repräsentiert und man sich somit in allen Bereichen des Netzwerkes auskennt.

Auffällig ist jedoch, dass die Ergebnisse sehr nahe beieinander liegen. Einer der Gründe dafür ist die von Grund auf bestehende Trefferquote, die durch das Hintergrundwissen im Netzwerk

zustande kommt. Durch das Wissen über den kürzesten Weg im Netzwerk von dem Start- zum Endknoten erreicht man ohne das Wissensmodell schon eine bestimmte Trefferquote für die Knoten. Somit spielt bei dieser Teilmenge die Definition des Wissensmodell kaum eine Rolle. Trotzdem ist es natürlich durch diese Modelle möglich die Erfolgsquote erheblich zu steigern, wie man im Vergleich zu den Musik oder Geographie sehen kann.

Abbildung 48: Resultate der General, Less General, More Specific und Most Specific Modells in Prozent

Des Weiteren kommt hier ebenfalls die im vorherigen Abschnitt schon erwähnte Problematik des hohen Grades bei Knoten, also den Hubs, zum tragen. Kurz zusammengefasst wird bei solchen Hubs die Wahrscheinlichkeitsverteilung kaum ins Gewicht fallen, da zum Beispiel unter 1000 Knoten ein einzelner Knoten mit einer im Vergleich zu den anderen Knoten hohe Wahrscheinlichkeit keinen signifikanten Einfluss auf das Ergebnis haben. Dies wurde zwar in der Formel berücksichtigt und verbesserte das allgemeine Resultat, bei ähnlichen Verläufen des Wissensmodells kommen allerdings keine sehr unterschiedlichen Ergebnisse zustande. Aber Trotzdem ist es wichtig, solch ein Modell richtig zu definieren, was im nächsten Abschnitt gezeigt wird.

6.1.2 General Model, Most Specific Model, Music- und Geographic Specialist

Um zu zeigen, dass die Wahl der Verteilung des Wissensmodell eine große Rolle spielt, werden in dem Abschnitt noch zwei weitere Wissensmodelle gezeigt und die Ergebnisse mit den im vorherigen Kapitel gezeigten Modellen verglichen. Die neuen Verteilungen sind in Abbildung 49 zu sehen (oben rechts und unten links) , die zum einen den Spezialist in Musik repräsentieren sollen, der sich nur in Musik sehr gut auskennt und in anderen Wissensbereichen überhaupt nicht. Zum anderen ein Äquivalent zum Musik Spezialist, der hier aber ein ausgeprägtes Wissen in Geographie besitzt.

Die Resultate werden in Abbildung 50 und 51 und im Vergleich zu dem "General Model" und "Most Specific Model" gezeigt.

Music Specialist:

Der Musik Spezialist erreicht nur eine Trefferquote von 18,7852%, was einer Knotenanzahl von 41039 entspricht. Dieses Ergebnis ist bei Weitem das schlechteste Resultat der Wissensmodelle.

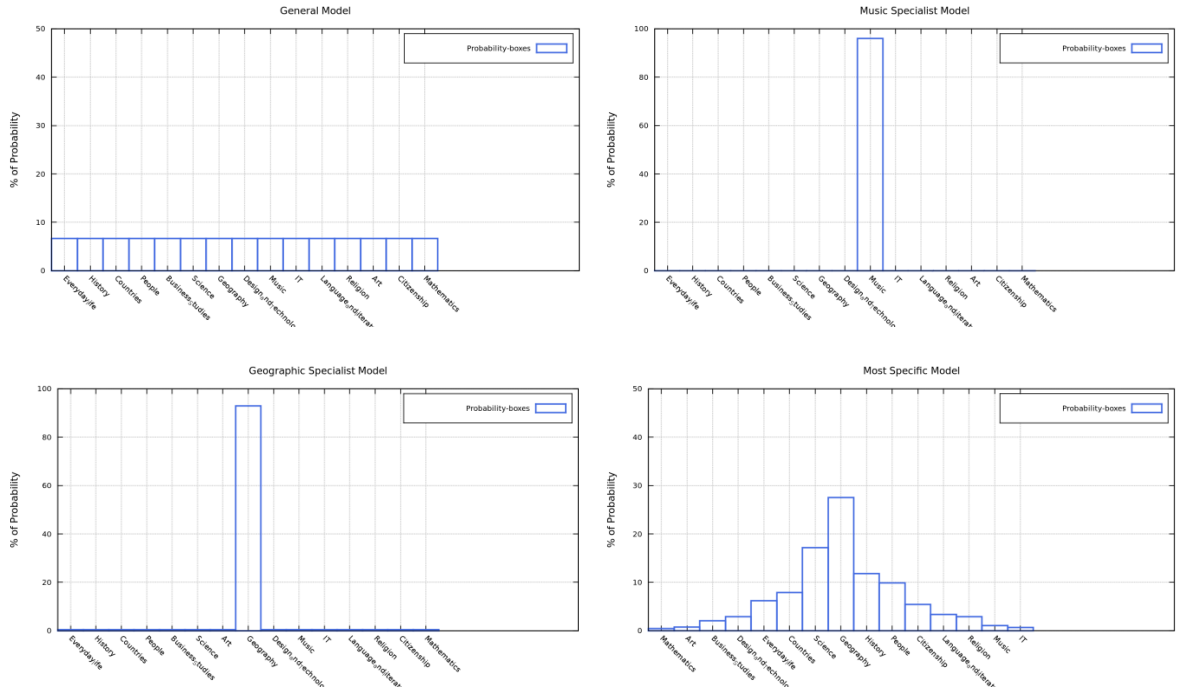


Abbildung 49: Wissensverteilung eines Users der Allgemeinwissen, Musik spezifisches, Geographie spezifisches und fast nur spezielles Wissen besitzt (von oben links nach unten rechts).

Geography Specialist:

Betrachtet man abschließend noch den Geographie Spezialist, liefert dieser nicht ganz so schlechte Ergebnisse wie der Musik Spezialist, aber im Vergleich zu den eher gleichverteilten Wissensmodellen im Abschnitt zuvor trotzdem keine ausreichende Trefferquote. Diese beträgt hier 21,5829%, was einer Knotenanzahl von 47151 entspricht.

Fasst man nun diese Statistiken zusammen, lässt sich sagen, dass eine etwas allgemeinere Wissensverteilung ein besseres Resultat liefert, als Modelle bei denen der Fokus auf einem einzelnen Gebiet liegt. Dies sieht man zum einen an der Höhe der Treffer und zum anderen lässt es sich auch sehr schön an der Statistik in Abbildung 50 erkennen, bei der die Hitrate der allgemeineren Wissensmodelle ca. 5% über der der Spezialisten liegt.

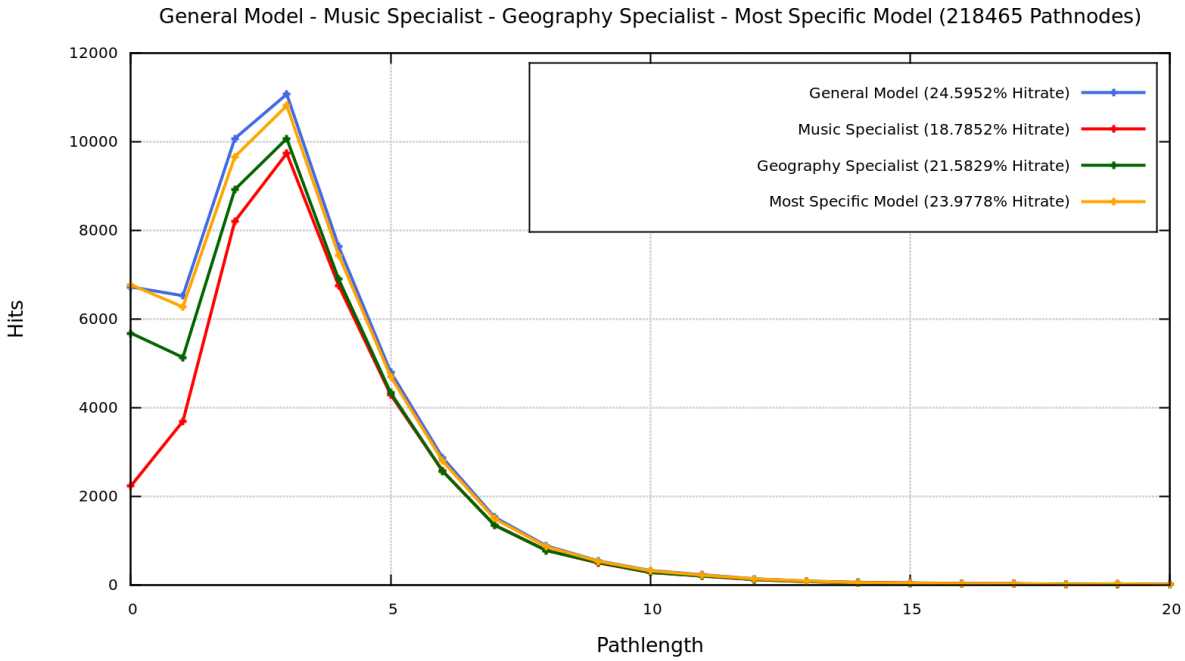


Abbildung 50: Resultate der General, Most Specific Models, Music Specialist und Geographic Specialist in Hits

Es zeigt sich ebenfalls, dass eine schlechte Definition des Wissensmodells gravierende Auswirkungen auf die Trefferquote des Algorithmus hat und sie dem allgemeinen User entsprechen sollte. Dies liefert zugleich vergleichsweise gute Resultate, aber es hindert den Algorithmus auch daran, noch bessere Ergebnisse zu liefern. Dieses Problem wird im Kapitel "Fehlerquellen" noch genauer erläutert.

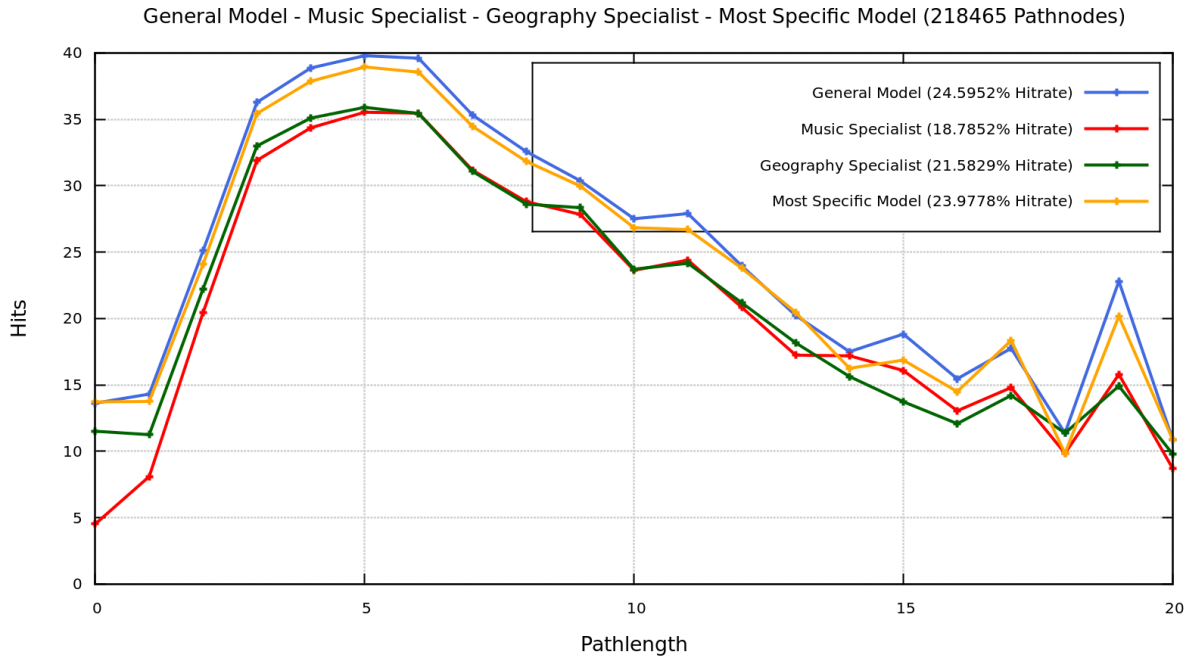


Abbildung 51: Resultate der General-, Most Specific Models, Music Specialist und Geographic Specialist in Hits

6.1.3 Wissensmodelle im Vergleich zu anderen Algorithmen

Da nun die Wissensmodelle untereinander verglichen und die Resultate gezeigt wurden, muss dieser Algorithmus noch den bereits vorhandenen Algorithmen im MUN System gegenüber gestellt werden.

Diese Gegenüberstellung ist in Abbildung 52 und 53 zu sehen. Es werden die definierten Modelle und der "Greedy" Algorithmus gezeigt. Der Greedy-Algorithmus berechnet seine nächsten Knoten nur anhand des Hintergrundwissens. Hier wird der Knoten gewählt, der den kürzesten Weg zum Zielknoten bildet. Falls mehrere Knoten die selbe Distanz bilden, wird zufällig einer dieser Knoten ausgewählt. Dieser Algorithmus liefert eine Trefferquote von 22,2735%, was einer Knotenanzahl von 48752 entspricht.

Somit liefert der Algorithmus mit den Wahrscheinlichkeitsmodellen eine bessere Trefferquote von ungefähr 2,5%, was einer Knotenanzahl von ca. 5500 im Schnitt gleicht. Vollständigkeitshalber wird noch der Vergleich zu den anderen Algorithmen gezeigt, die wesentlich schlechter abschneiden als der Greedy-Algorithmus.

Abbildung 52: Prozentuale Trefferquote pro Schritt im Pfad im Vergleich zu "Greedy"

Bei der "Random" Variante werden alle Schritte zum Ziel per Zufall ausgesucht. Es wird so versucht zum Zielknoten zu gelangen. Logischerweise liefert dieser Algorithmus eine sehr niedrige Trefferquote von 3,5725%, da hier gerade bei Hubs oder ähnlichen Knoten die Wahrscheinlichkeit verschwindend gering ist, die selben Knoten zu wählen, die ein User wählen würde.

Ähnlich dem "Random" Algorithmus schneidet auch der "StochastikGreedy" ab. Dieser ähnelt dem Verfahren des Greedy, doch wird hier noch mit den Parametern der Fitness und Utility gearbeitet. Er erreicht eine Trefferquote von 3,49353%, was einer Knotenanzahl von 7656 entspricht.

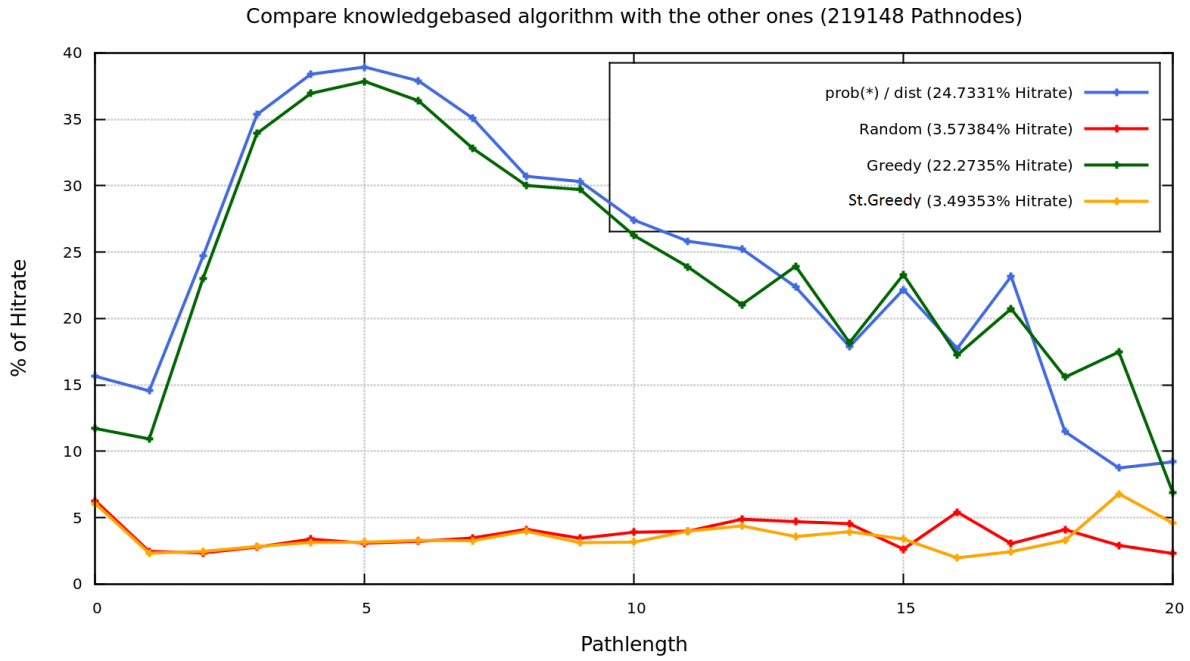


Abbildung 53: Das Best abschneidende Wissensmodell im Vergleich zu den vorhandenen Algorithmus

6.2 Fehlerquellen

In diesem Kapitel werden die Fehlerquellen im Detail beschrieben, die dazu führen, dass dieser Algorithmus keine signifikant besseren Ergebnisse liefert. Wie der Algorithmus mit den Wissensmodellen diese Probleme lösen kann, wird dann im Kapitel "Futurework" erläutert.

Betrachtet man nun die Statistik in Abbildung 54, erkennt man den Ergebnisverlauf des Algorithmus bezogen auf die Wissensmodelle. Diese Resultate werden nun der prozentualen Verteilung des Grades der Knoten pro Schritt gegenüber gestellt. Es lässt sich erkennen, dass bei dem Startknoten und dem ersten Knoten auf dem Pfad prozentual der Grad der Knoten am höchsten ist und im Gegensatz dazu, die Trefferquote des Algorithmus am niedrigsten. Nachdem der Grad sein Maximum erreicht hat und in Schritt 2 wieder abfällt, steigt auch die Trefferquote des Algorithmus. Also ist klar, dass hier eine mögliche Verbindung der schlechten Resultate mit der Höhe des Grades besteht.

Weitere Analysen bestätigten diesen Zusammenhang zwischen einem hohen Grad und einer schlechten Trefferquote. Hierfür wurden verschiedenste Artikel und deren Wahrscheinlichkeiten für die Nachbarknoten im Detail betrachtet. Dabei zeigte sich, dass gerade bei den sogenannten Hubs (sehr hohe Anzahl an Nachbarknoten) die Wahrscheinlichkeiten der Kategorien keine große Rollen spielen. Nimmt man nun an, dass solch ein Knoten zum Beispiel 1000

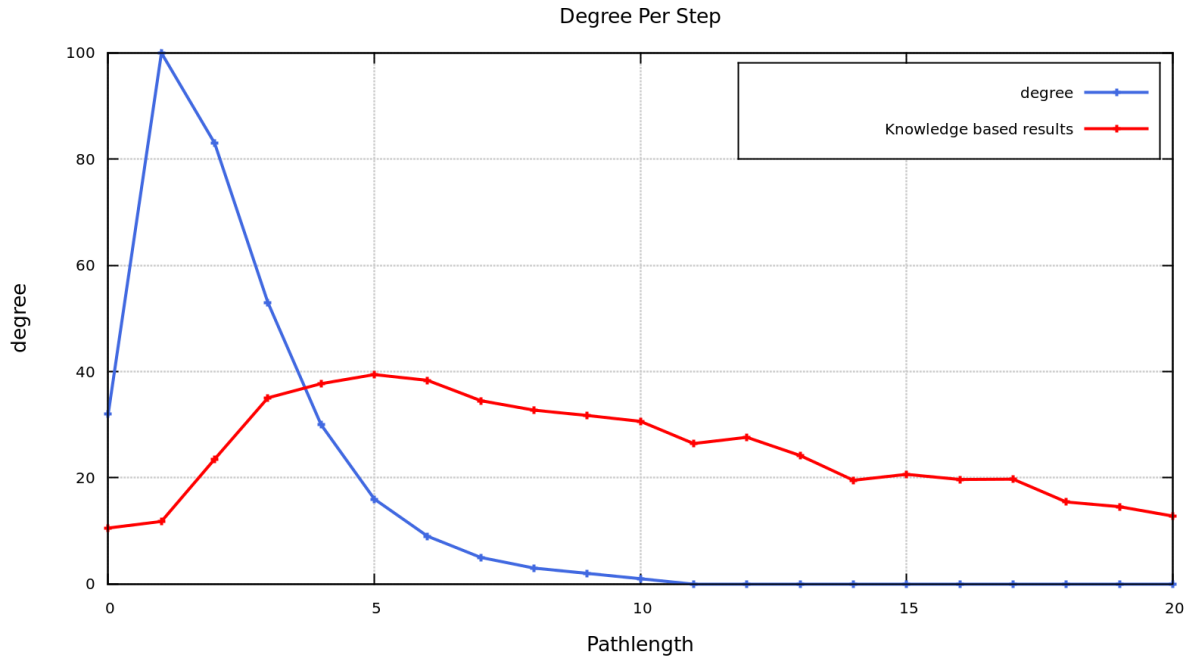


Abbildung 54: Vergleich des Verlaufs der Trefferquote des Algorithmus und der durchschnittlichen Höhe des Grades pro Schritt

Nachbarknoten hat und sich darunter 2 Artikel befinden, die der Kategorie angehören, die die höchste Wahrscheinlichkeit repräsentieren. Angenommen es handelt sich hier um eine Wahrscheinlichkeit von 25%, sinkt dieser Wert aber bei der Skalierung für 1000 Knoten drastisch nach unten. Durch diese Skalierung fallen diese Knoten nun nicht mehr stark genug ins Gewicht, um hier bei der Auswahl des nächsten Nachbarknotens signifikant aufzutauchen.

Man kann diesem Problem, wie weiter oben schön erwähnt, durch das Miteinbeziehen des Knotengrades in die Formel entgegenwirken, doch wird so das Problem nicht komplett gelöst, da jetzt jede größere Veränderung in dem Wissensmodell Veränderungen in den Resultaten mit sich zieht, wie man es zum Beispiel bei dem Musik - Spezialist sehen kann.

Die Höhe des Knotengrades bei Hubs reduziert zwar die Trefferquote, ist aber nicht der Punkt, der die Resultate so signifikant eindämmt. Es stellte sich nach vielfachem Testen und Analysieren des Algorithmus und der Resultate heraus, dass bei den Userdaten verschiedene Usergruppen, oder anders gesagt, verschiedene Wissensgruppen existieren. Dadurch, dass es verschiedene Gruppen gibt, die ihre jeweilige Wissensspezialisierung besitzen, ist es auch nicht möglich, diese mit nur einem einzigen Wissensmodell zu beschreiben und zu versuchen, deren Navigation bestmöglich zu simulieren. Das bedeutet, man muss versuchen, den einzelnen Gruppen ein eigenes Wissensmodell zuzuweisen und anhand dieser, die jeweilige Gruppe zu simulieren.

Um zu zeigen, dass es nötig ist, die Wissensmodelle auf einzelne Gruppen anzuwenden, wurden Wissensmodelle für kleinere zufällige Gruppen innerhalb des Netzwerkes berechnet. Die Trefferquote dieser Gruppen wurden miteinander verglichen und hierbei stellte sich heraus, dass Wissensmodelle, bei denen die Wissensbasis relativ gleich war, auch die Ergebnisse besser waren als in Usergruppen, die eine allgemeinere und breitgefächertes Wissensmodell besaßen.

So lässt sich schlussfolgern, dass man die Wissensmodelle auf spezielle Gruppen oder Bereiche des Netzwerkes anwenden sollte, um signifikant bessere Ergebnisse zu erhalten.

6.3 Résumé

Fasst man die Resultate zusammen, kommt man zu dem Schluss, dass der Algorithmus im Vergleich zu den bereits bestehenden Algorithmen bessere Werte, die aber nicht signifikant besser sind. Des Weiteren wurde gezeigt, welche Auswirkungen ein schlechtes Wissensmodell auf die Ergebnisse haben und welches Modell die beste Trefferquote liefert. Spezialisten schneiden also wesentlich schlechter ab als Modelle die mehr Allgemeinwissen orientierter definiert sind. Dies ist darauf zurückzuführen, das man sich durch eine allgemeine Wissensverteilung in fast allen Bereichen des Netzwerkes etwas auskennt, was dem "Greedy" Algorithmus ähnelt und somit ähnlich gute bis bessere Resultate erreicht.

Es ist jedoch noch viel Spielraum nach oben. Mögliche Problemquellen für die nicht signifikant besseren Trefferquoten sind zum einen die Hubs innerhalb des Netzwerkes (Abbildung 55) und die Unschärfe die bei der Definition und Verwendung der Wissensmodellen auftritt (Abbildung 56). Diese werden im nächsten Kapitel "Futurework" erläutert.

Abschließend ist zu sagen, dass durch den hier in der Arbeit beschriebenem Algorithmus bessere Ergebnisse erzielt wurden, es aber noch mögliche Verbesserungen gibt. Des Weiteren kann die menschliche Navigation somit anhand eines Wissensmodell mit einer 25%igen Trefferquote simuliert werden.

7 Futurework

Für zukünftige Arbeiten an diesem Algorithmus, muss die Verwendung und Erstellung des Wissensmodell angepasst und die Userdaten modifiziert werden. Man kann nicht alle vorhandenen User mit nur einem Wissensmodell simulieren, da diese alle zu unterschiedlich sind. Also muss man einzelne Usergruppen separat betrachten und versuchen, die Pfade die von den selben Usergruppen geklickt wurden zu clustern und für diese wiederum eigene Wissensmodelle erstellen.

Durch dieses Verfahren sollten sich nun verschiedenste Usergruppen bilden, die eine bestimmte Wissensrichtung oder einen bestimmten Wissensbereich abdecken. Diese sollten dann auch in den bestimmten Bereichen des Netzwerkes getestet werden und können anderen Usergruppen gegenüber gestellt werden. Daraus würde resultieren, welche Wissensauslegung am besten abschneidet.

Ein weiterer Schritt wäre die Problematik mit Hubs zu beheben. Für eine bessere Entscheidbarkeit bei solch einer hohen Anzahl an Knoten werden mehr Informationen benötig. 16 Kategorien sind hier, wie im „Wikipedia for schools“ ‚nicht ausreichend um bei 1000 Nachbarknoten gut genug entscheiden zu können.

Hier könnte man zum Beispiel noch weitere Informationen über Synergien zwischen den einzelnen Kategorien zwischen Nachbarknoten und dem aktuellen Knoten bereit stellen und in die Entscheidungen einfließen lassen. Des Weiteren könnten nicht nur die Topkategorien der Artikel als Wahrscheinlichkeitsmaß verwendet werden, sondern auch die darunter liegenden Kategorien zur Bestimmung der Wahrscheinlichkeit.

Somit ist bei diesem Bereich der Forschung noch Potenzial vorhanden, das durch die eben genannten Verbesserungen ausgeschöpft werden kann.

8 Literaturverzeichnis

[1] D. Easley und J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*, Cambridge University Press, 2010.

[2] S. Milgram, *The small world problem*. Psychology Today, 1967

[3] Shawn M. Douglas, Gaetano T. Montelione, and Mark Gerstein. *PubNet: A flexible system for visualizing literature derived networks*. Genome Biology, 6(9), 2005

[4] Graph structure in the Web
http://www.sciencedirect.com/science/article/pii/S1389128600000839

[5] Diestel: *Graphentheorie*, 2. Auflage, 2000

[6] Alain Degenne, Michel Forsè: *Introducing Social Networks*

[7] Stanley Milgram: *The Small World Problem*. In: Psychology Today. Mai 1967

[8] Travers, J., & Milgram, *S. An experimental study of the small world problem*., 1969

[6] Facebook Userzahlen http://www.allfacebook.de/userdata/?period=1month 15.7.2013 10:30 Uhr

[7] Soziale Netzwerke http://www.uibk.ac.at/psychologie/mitarbeiter/leidlmair/soziale-netzwerke.pdf 15.7.2013 10:30 Uhr

[8] Christina Prell. *Social Network Analysis: History, Theory and Methodology*, 2011

[9] Wayne Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[10] Google Search http://www.google.com/insidesearch/howsearchworks/ 15.07.2013 12:30 Uhr

[11] Lada Adamic and Natalie Glance. The political blogosphere and the 2004 U.S. election: Divided they blog. *In Proceedings of the 3rd International Workshop on Link Discovery*, pages 36–43, 2005.

[12] Koudelka Otto. Communication Networks lecture at TU Graz, 2009

[13] F. Heart, A. McKenzie, J. McQuillian, and D. Walden. *ARPANET Completion Report*. Bolt,Beranek and Newman, 1978.

[14] Falk Schreiber. *Analyse und Visualisierung biologischer Netzwerke*. 2009

[15] Le Novere N, Moodie S, Sorokin A, Hucka M, Schreiber F, Demir E, Mi H, Matsuoka Y, Wegner K, Kitano H (2008) Systems biology graphical notation: Process diagram lev el 1. Nature Prec, DOI: npre.2008.2320.1

[16] Dyer, R.J. and Nason, J.D. (2004) *Population graphs: the graph theoretic shape of genetic structure*. Mol. Ecol.13, 1713–1727

[17] Jon Kleinberg. *Authoritative sources in a hyperlinked environment*. Journal of the ACM, 1999.

[18] Sergey Brin and Lawrence Page. *The anatomy of a large-scale hypertextual Web search engine*. In Proc. 7th International World Wide Web Conference

[19] Alice X. Zheng, Andrew Y. Ng, and Michael I. Jordan. Stable algorithms for link analysis. In Proc. *24th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 258–266, 2001.

[20] M. Eder. *Entwicklung eines Frameworks zur Navigationssimulation*. Technische Universität Graz, 2013.

[21] David A. Bader and Kamesh Madduri. *SNAP: Small-world network analysis and partitioning: An open-source parallel graph framework for the exploration of large-scale networks*. In Proc. 22nd IEEE International Symposium on Parallel and Distributed Processing, 2008.

[22] http://snap.stanford.edu/index.html 19.08.2013 18:30

[23] Yuan Shen, Santiago Mazuelas, Moe, Z. Win. *A Theoretical Foundation of Network.* Massachusetts Institute of Technology

[24] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. Berkeley 1997

[25] Ted G. Lewis. *Network Science: Theory and Applications.*

[26] *Wikispeedia.* http://www.cs.mcgill.ca/~rwest/wikispeedia/ 05.09.2013 13:00

[27] *Wikipedia Selection for Schools.* http://schools-wikipedia.org/ 07.09.2013 12:00

[28] *Wikipedia.* http://www.wikipedia.org/ 08.09.2013 10:00

[29] *Domain Model* http://consultingblogs.emc.com/vanessatong/archive/2008/06/25/where-do-i-start.aspx 15.09.2013 11:15

[30] Peter Brusilovsky, Julita Vassileva. *Course sequencing techniques for large-scale web-based education*

[31] Brusilovsky, P. *Developing adaptive educational hypermedia systems: from design models to authoring tools*, in T. Murray, S. Blessing and S. Ainsworth (Eds.) Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive, and Intelligent Educational Software,Ablex, Norwood, In Press

[32] Brusilovsky, P. L. *A framework for intelligent knowledge sequencing and task sequencing*, in C. Frasson, G. Gauthier and G.I. McCalla (Eds.) Intelligent Tutoring Systems, Springer-Verlag, Berlin, pp.499–506.

[33] Vassileva, J. Dynamic *CAL-courseware generation within an ITS-shell architecture*, in I. Tomek. (Ed.) Computer Assisted Learning. Lecture Notes in Computer Science, Vol.602, Springer-Verlag, Berlin, pp.581–591.

[34] Graph: http://www.kennedy-capital.de/netzwerk-2/ 9.07.2013 15:00 Uhr