Universität des Saarlandes
Naturwissenschaftlich-Technische Fakultät I
Fachrichtung Informatik

Masterarbeit

Von der Parallelen Facettierten Suche zur Parallelen Exploration: Stärkung eines Interaktionsparadigmas

Vorgelegt von:
Adrian Spirescu
am 30. September 2016

Begutachtet von:
Prof. Dr. Anthony Jameson
Prof. Dr. Antonio Krüger

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, den 30. September 2016

Zusammenfassung

Die übergeordnete Forschungsfrage dieser Arbeit nimmt als Ausgangspunkt das Interaktionsparadigma der Parallelen Facettierten Suche (PFB) her, das ich in meiner Bachelorarbeit im Jahr 2013 als eine Erweiterung der traditionellen Facettierten Suche einführte, und welches das gleichzeitige Verfolgen mehrerer Explorationspfade ermöglicht. Die Frage war, wie man PFB aus einem interessanten Vorführmodell in ein mächtiges Interaktionsparadigma verwandeln kann, das einer großen Anzahl von Benutzern mit unterschiedlichen Endgeräten auf diversen Anwendungsdomänen Vorteile bietet.

Es wird zunächst auf die Vorteile von PFB gegenüber traditionellen Suchsystemen eingegangen: das Vergleichen mehrerer Ergebnismengen, die Suche nach einer Kombination passender Elemente und die kollaborative Erkundung. Danach werden die vor Beginn dieser Arbeit bestandenen Beschränkungen zusammengefasst: die fehlende Gewährleistung der reibungslosen Nutzung auf großen sowie kleinen Bildschirmen und die Gewöhnungsbedürftigkeit des neuartigen Ansatzes. Um diese Beschränkungen aufzuheben und das neue Paradigma der Parallelen Exploration (PE) zu erfüllen, ergeben sich die zwei grundlegenden Forschungsziele dieser Arbeit: die Erkundung eines geeigneten Interaktionsdesigns für PE und das Hinzufügen neuer Funktionalität.

Anhand der gegebenen Anforderungen wie Responsivität und Optimierung für mobile Nutzung wird ein Interaktionsdesign vorgestellt, das in drei Schritten iterativ ausgearbeitet wird. Zunächst wird eine Version vorgestellt, die der facettierten Suche ähnelt und den Benutzer Schritt für Schritt an das Paradigma von PE näherbringt. Anschließend wird gezeigt, wie das automatische Zusammenklappen und Auseinanderziehen von Ergebnisfenstern die Übersicht auf allen Geräten – vor allem auf kleineren Displays – verbessert.

Der zweite Teil dieser Arbeit geht auf eine Vielzahl von Erweiterungen der Funktionalität der PFB-Benutzerschnittstelle ein, deren Stärke dadurch wesentlich erhöht wird. Die globale Suche ermöglicht Benutzern, Volltextsuchen auf eine mit dem Interaktionsdesign von PE kompatible Art und Weise durchzuführen. *Pivoting* erlaubt es, ausgehend von einer Menge von Elementen, verwandte Elemente zu finden und hilft beispielsweise bei der Beantwortung der Frage „Welche Hotels sind

in der Nähe dieser Konzerte?“. Die *manuelle Selektion* ermöglicht es Benutzern, ihre parallele Erkundung auf eine feinkörnige Ebene zu übertragen. *Lesezeichen* helfen Benutzern dabei, den aktuellen Zustand ihrer Erkundung (ihres *Explorationsbaums*) zu speichern, sodass sie später zurückkehren oder ihn mit Anderen teilen können. Die *Wunschliste* ermöglicht es nicht nur, einem Benutzer eine für ihn interessante Liste von Elementen zu speichern, sondern kann auch als Ausgangspunkt für weitere Erkundung dienen. *QuickStart-Applikationen* zeigen nach der Auswahl einer natürlichsprachlichen Frage die Ergebnisse komplexer PE-Anfragen. *Kontextspezifische Tooltipps* geben dem Benutzer, trotz der Tatsache, dass der durch PE erstellte Explorationsbaum viele Formen annehmen kann, die Möglichkeit von den Vorzügen traditioneller Tooltipps Gebrauch zu machen.

Das resultierende Paradigma der Parallelen Exploration wird in dieser Arbeit mit Beispielen aus einer Webapplikation, die innerhalb der „EIT Digital Activity“ ExplorMI 360 in Mailand und Nizza entstand, veranschaulicht. Die Arbeit bespricht kurz weitere im Gespräch stehende, praktische Anwendungsfälle sowie weitere vielversprechende Erweiterungen des Paradigmas.

Vorwort

Um einen Vorgeschmack auf die Parallele Exploration zu geben, möchte ich bereits an dieser Stelle anhand eines praktischen Beispiels darauf eingehen. Ich werde zeigen, wie ein erfahrener Benutzer mit der Benutzerschnittstelle interagieren könnte, um möglichst viel Funktionalität zu demonstrieren. Ein neuer Benutzer würde normalerweise mit einfacheren Beispielen beginnen.

Stellen Sie sich vor, Sie planen mit Ihrem Partner einen Aufenthalt in Nizza, wobei Sie nur eine vage Vorstellung von Ihrem Besuch haben und die vorhandenen Angebote erkunden möchten. Während Sie sich für Konzerte interessieren, möchte sich Ihr Partner vielleicht das Theaterangebot näher ansehen. Aus zwei unterschiedlichen Listen wählen Sie und Ihr Partner jeweils eine Veranstaltung, die sie besonders interessiert, in diesem Fall entscheiden Sie sich für das Konzert „Jeunes Prodiges Russes“ und Ihr Partner für das Theaterstück „Panique au Ministere“. Angenommen, Sie wollen die Wahl des Hotels abhängig von den ausgesuchten Veranstaltungen machen, können Sie nun Hotels in der Nähe der beiden Veranstaltungsorte suchen und sich diese anzeigen lassen. Für die endgültige Entscheidung, welches Hotel das Bessere ist, können Sie sich die Detailinformationen der Hotels nebeneinander anzeigen lassen, um diese gleichzeitig zu vergleichen, ohne mehrere Registerkarten im Browser öffnen zu müssen. Die Benutzerschnittstelle strukturiert die gesamte Erkundung, ohne dass Sie sich um die Anordnung der Ergebnisse oder die Anpassung am Bildschirmbereich kümmern müssen. Im Verlauf dieser Arbeit werde ich genauer auf diese und andere Vorteile eingehen.

Diese Arbeit wurde an der Universität des Saarlandes in Zusammenhang mit dem deutschen Forschungszentrum für künstliche Intelligenz (DFKI) im Fachbereich intelligente Benutzerschnittstellen (Intelligent User Interfaces, IUI) geschrieben. Wie im Abschnitt 1.1 näher erörtert wird, wurde der Grundgedanke hinter der Parallelen Exploration im Jahre 2012 von Prof. Dr. Anthony Jameson und Sven Buschbeck eingeführt [7]. Im gleichen Jahr leistete ich meine ersten Beiträge zu diesem Thema [6] [16] und beschäftigte mich im Rahmen meiner Bachelorarbeit [26] mit einem frühen Prototyp. Die Aufgabe dieser Masterarbeit ist es, wesentliche Verbesserungen und Erweiterungen des Interaktionsdesigns und die Funktionalität der Parallelen Exploration zu entwerfen und zu implementieren. Die Parallele Exploration soll als

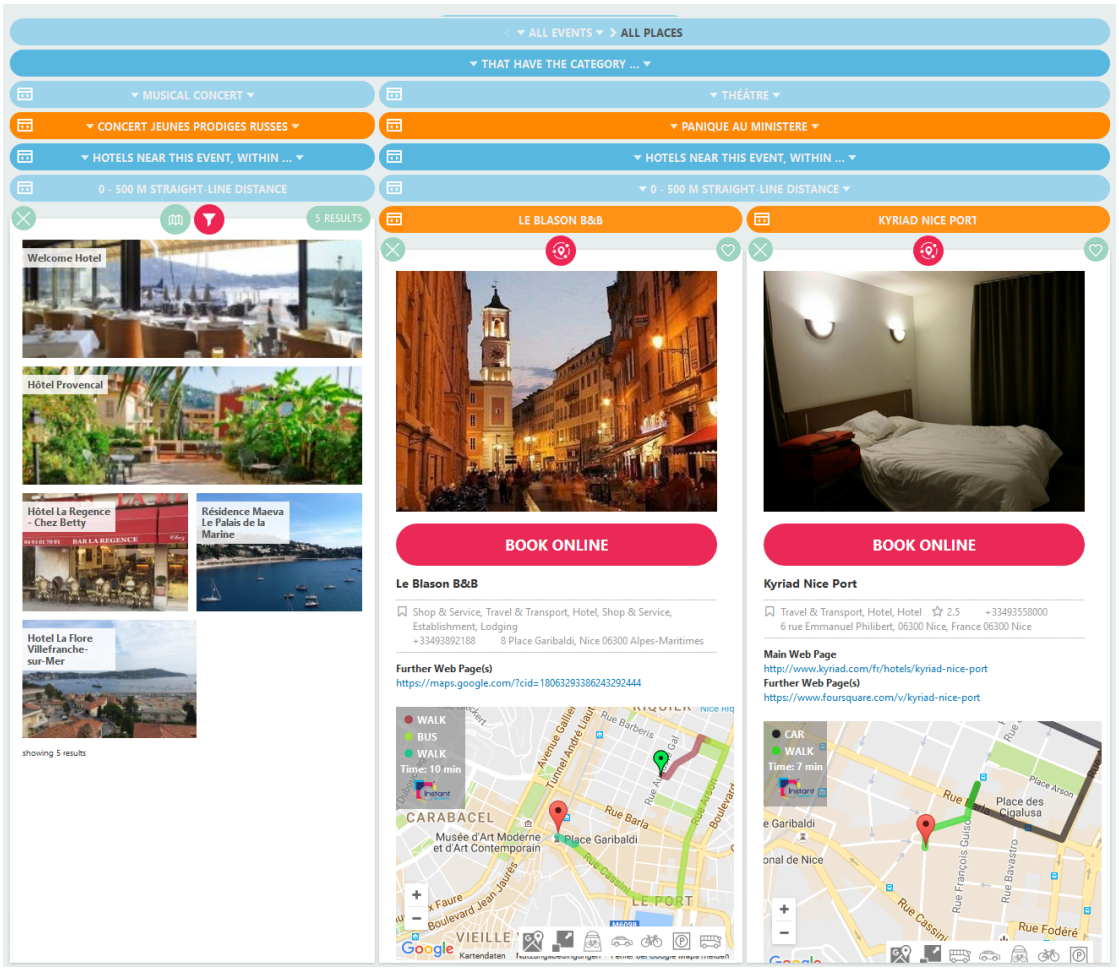

Abbildung 0.1: Parallele Exploration bei der Planung eines Aufenthalts in Nizza; die Webapplikationen sind unter nice.3cixty.com und milan.3cixty.com erreichbar

eine Kernkomponente des vom EIT Digital geförderten Projekts 3CIXTY in den Jahren 2014 bis 2016 und darüber hinaus für die kommerzielle Verwertung dienen. Die PE wurde hierbei vor Allem in der Webapplikation EXPLORMI 360 realisiert (s. Abbildung 0.1), die durch Besucher der Expo 2015 in Mailand verwendet und danach verallgemeinert wurde.

An dieser Stelle möchte ich mich bei allen Personen, die mich bei der Erstellung dieser Arbeit unterstützt haben, bedanken. Mein besonderer Dank gilt Prof. Dr. Anthony Jameson für die fortwährende Unterstützung und Hilfsbereitschaft bei

der Anfertigung dieser Arbeit. Ich möchte mich außerdem bei Kai-Dominik Kuhn, der die datenbankspezifischen Aspekte der Parallelen Exploration übernommen hat, für die reibungslose Zusammenarbeit bedanken. Des Weiteren gilt mein Dank allen Personen, die mich mit wertvoller Rückmeldung zu der Benutzerschnittstelle unterstützt haben.

Inhaltsverzeichnis

1 Einleitung

Zunächst gehe ich darauf ein, was die Parallele Facettierte Suche (nachfolgend „Parallel Faceted Browsing“ oder auch PFB genannt) überhaupt ist und erkläre die wesentlichen Prinzipien von PFB. Anschließend gehe auf dessen Vorteile und Beschränkungen, die vor dem Beginn dieser Forschungsarbeit vorlagen, ein und erläutere, welche Forschungsfragen sich daraus ergeben.

1.1 Was ist PFB?

Die Idee hinter dem Paradigma von PFB entstand im Jahr 2012 und baute auf der Benutzerschnittstelle für das EU Projekt Glocal auf, die von Sven Buschbeck unter der Leitung von Prof. Dr. Anthony Jameson entwickelt wurde [5] [4] [14]. Diese Benutzerschnittstelle unterstützte die Exploration semantischer Daten in Bezug auf Ereignisse und stellte die Exploration des Benutzers in einer baumartigen Struktur, im Folgenden *Explorationsbaum* genannt, dar. PFB ist eine Weiterentwicklung davon und gleichzeitig eine Verallgemeinerung der traditionellen facettierten Suche: während der Benutzer bei FB üblicherweise Filter spezifiziert und die Ergebnisse als eine einzige Menge angezeigt werden [27], erlaubt es der neuartige Ansatz mehrere Suchpfade gleichzeitig zu verfolgen. Der Benutzer kann die Ergebnisse unterschiedlicher Filtereinstellungen in sogenannten Ergebnisfenster (nachfolgend auch *Knoten* genannt) gleichzeitig betrachten, was in gängigen Suchsystemen nicht ohne Umwege und Einschränkungen möglich ist. Des Weiteren hilft PFB dem Benutzer dabei, eine klare Struktur seiner Erkundung aufzubauen. Die Abbildungen 1.1 und 1.2 zeigen zwei Benutzerschnittstellen aus dem Jahr 2013, die zwar auf dieselbe Datenquelle zugreifen und eine ähnliche Gestaltung in Hinblick auf die Farbwahl haben, aber unterschiedliche Interaktionsparadigmen unterstützen. In der FB-UI kann der Benutzer in der Menüleiste, basierend auf seine Interessen, Filter spezifizieren und sich anschließend die Treffer in einer Ergebnismenge anzeigen lassen. Während der Benutzer bei FB stets nur eine Ergebnismenge sieht, die sich beim Setzen oder Ändern eines Filters aktualisiert, erzeugt der Benutzer bei der Erkundung von Daten mit PFB neue Ergebnismengen, die er miteinander

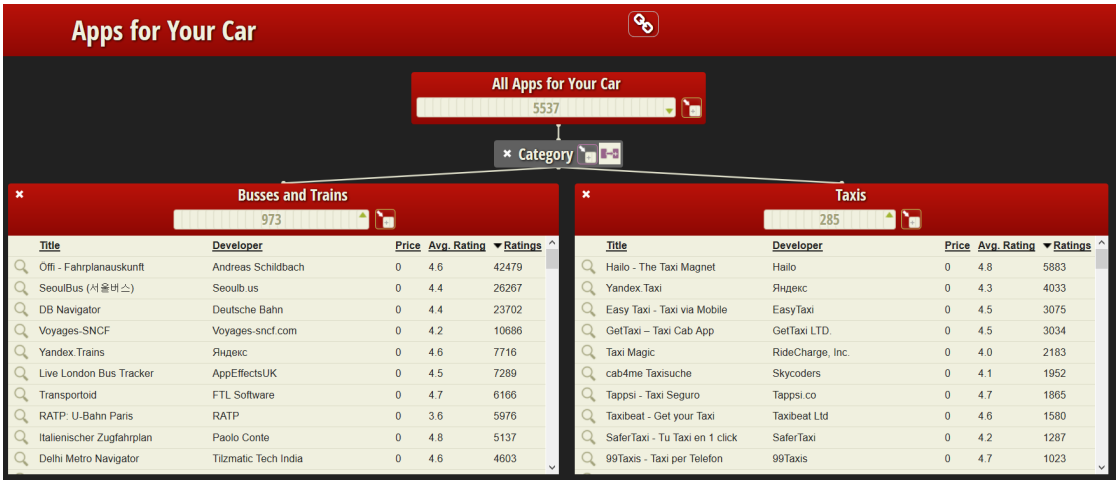

Abbildung 1.1: Eine Benutzerschnittstelle, die das PFB-Paradigma unterstützt

vergleichen kann. In Abbildung 1.1 sehen wir, dass der Inhalt von zwei Kategorien nebeneinander gezeigt wird.

1.2 Vorteile von PFB für Benutzer

Bereits im Jahre 2013 vermutete man, dass PFB Vorteile für den Benutzer darstellen könnte. In diesem Abschnitt möchte ich zunächst die bereits bekannten und bestätigten Vorteile von PFB kurz zusammenfassen und anschließend auf diejenigen Vorteile eingehen, die durch diese Arbeit neu dazugekommen sind.

Bei der Suche nach Kombinationen von einzelnen Elementen aus mehreren Ergebnismengen erwies sich PFB als überaus nützlich. Bereits in der Benutzerstudie, die ich im Rahmen meiner Bachelorarbeit [26] (Kapitel 5.2.2) ausgearbeitet habe, konnten die Teilnehmer erfolgreich aus einer großen Auswahl an Lebensmitteln ihr Frühstück planen und fanden die neuartige Benutzerschnittstelle hilfreich.

Des Weiteren half ihnen PFB dabei, zusammenhängende Dinge gleichzeitig zu sehen: sie konnten mithilfe von PFB beispielsweise ähnliche Lebensmittel unterschiedlicher Hersteller miteinander vergleichen und sich somit über das gesamte Angebot einen Überblick verschaffen.

Auch bei der Suche nach einem einzelnen Element stellte PFB einen Vorteil dar, da mehrere Suchpfade gleichzeitig verfolgt werden konnten und dies von den Benutzern ebenso aufgefasst wurde.

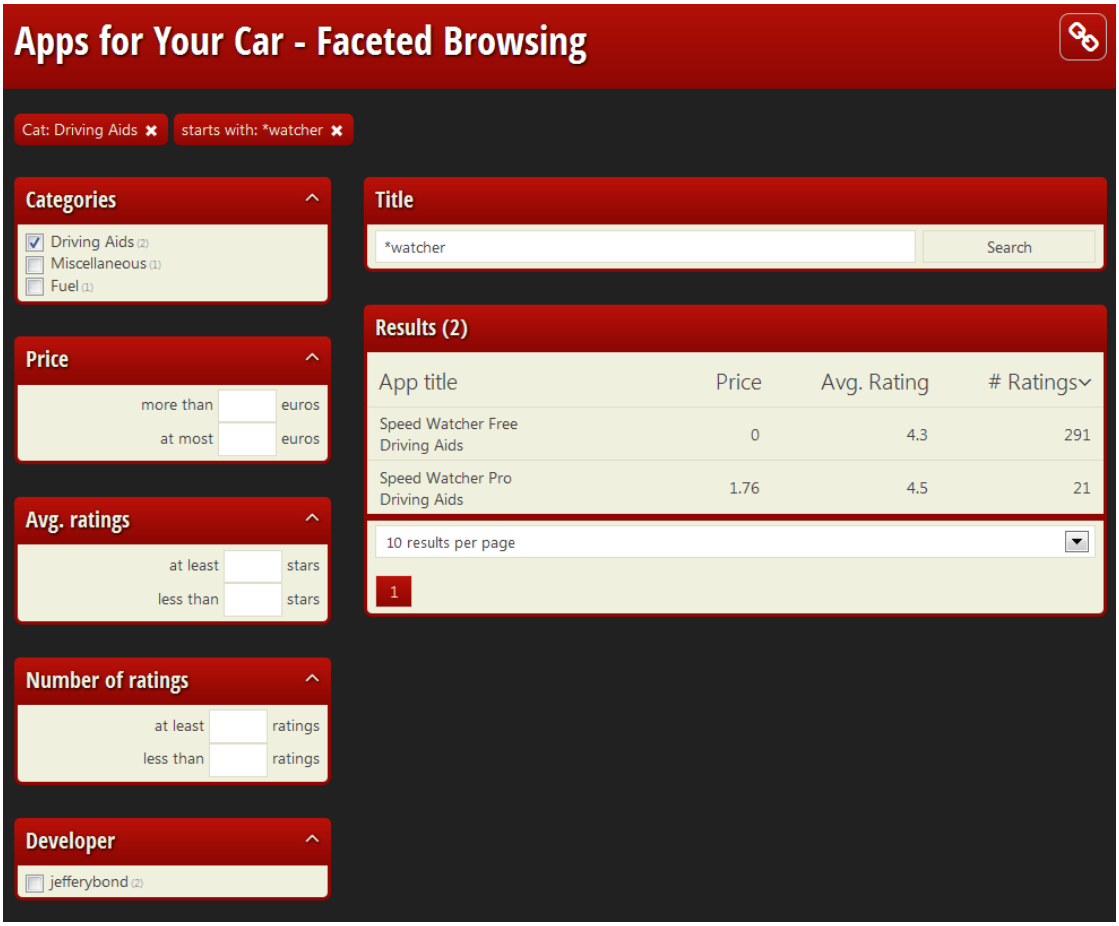

Abbildung 1.2: Eine Benutzerschnittstelle, die das FB-Paradigma unterstützt

Während herkömmliche Suchsysteme, wie beispielsweise die Textsuche, gut bei der gezielten Suche nach bestimmten Sachen sind, zeigen diese Schwächen bei der Erkundung, wenn man nur eine vage Vorstellung darüber hat wonach man sucht [29] [27]. PFB half dem Benutzer bei der explorativen Suche, Elemente aus sehr großen Datenmengen zu erkunden, wenn der gesamte Suchraum sehr komplex ist.

Ein weiterer Vorteil ist die gemeinschaftliche Nutzung, da Benutzer ihre Suchen samt Strukturbaum miteinander teilen konnten. Das Prinzip, parallel und gemeinschaftlich an einer Suche zu arbeiten, lässt sich mit den bereits genannten Vorteilen kombinieren.

Durch die in dieser Arbeit beschriebenen Erweiterungen der Interaktionsmöglichkei-

ten von PFB werden Anfragen ermöglicht, die normalerweise nur mit Anfragesprachen formulierbar sind oder bereits vordefiniert sind. Das Vergleichen von Paaren, die in einer bestimmten Beziehung zueinanderstehen ist ein typisches Beispiel dafür: sucht der Benutzer nach Hotels, die sich in der Nähe von Fitnesszentren befinden, ist das mit herkömmlichen Suchsystemen kaum realisierbar, insofern diese kein vordefiniertes Attribut haben, das die Verknüpfung zu den Fitnesszentren abbildet. Wie in dem einleitenden Beispiel bereits gezeigt, ermöglicht PE dem Benutzer die Beantwortung solcher Anfragen — darüber hinaus können die sich in der Nähe befindlichen Fitnesszentren auch erkundet werden. Herkömmliche Suchsysteme, beispielsweise traditionelles FB mit dem vordefinierten Attribut „hat Fitnesszentren in der Nähe“, würden nur die Information bereitstellen, dass sich solche in der Nähe des Hotels befinden, aber nicht explizit angeben, wo diese genau sind. Ebenso ist das Vordefinieren jeglicher sinnvollen Relation in Form von solchen Attributen in der Praxis nur beschränkt handhabbar, da die mögliche Anzahl solcher sehr hoch sein kann.

Ein weiterer Vorteil, der durch Einbindung mehrerer Entitätstypen ermöglicht wird, stellt das Aufbauen von Plänen, die mehrere unterschiedliche Objekte umfassen, dar. Ein Benutzer der ein Konzert von Elton John besuchen möchte, kann seine ganze Reise mithilfe von PE planen: vor dem Konzert möchte er in einem guten Restaurant essen. Auch die U-Bahnverbindungen zwischen seinen Zielen kann er überblicken und somit Objekte im Hinblick auf Beziehungen zu anderen Typen von Objekten miteinander vergleichen. Ein Nebeneffekt von PE ist in diesem Fall die Rolle als Multifunktionswerkzeug: der Benutzer ist nicht auf externe Applikationen angewiesen.

Als Erweiterung der oben beschriebenen Szenarien könnte der Benutzer mehrere Pläne beziehungsweise Alternativen zuerst hypothetisch aufbauen, um sich später für eine Möglichkeit zu entscheiden. Neben den Vorteilen, einzelne Elemente oder Mengen von Elementen miteinander vergleichen zu können, ist auch der Vergleich von ganzen Abläufen mit einer PE-Benutzerschnittstelle möglich, die mehrere Entitätstypen unterstützt.

1.3 Beschränkungen von PFB für Benutzer

Es stellte sich heraus, dass PFB noch kein vollkommener Ersatz für FB war. Das Feedback in Form eines persönlichen Gesprächs auf der CHI 2013 in Paris mit der UI-Expertin Prof. Marti Hearst – die als Miterfinderin von FB gilt – machte deutlich, dass PFB grundsätzlich Vorteile gegenüber traditionellen Suchsystemen hat, aber die Nutzer einige Funktionalitäten von FB vermissten. Während man bei

FB-Benutzerschnittstellen die Filter beliebig setzen, entfernen und abändern kann, war diese Möglichkeit in der Version von PFB, die vor dieser Masterarbeit bestand, nicht gegeben. Möchte man einen spezifischen Filter durch einen anderen ersetzen, so war der Benutzer dazu gezwungen, einige Operationen, die er bereits mit der UI gemacht hatte, erneut durchzuführen.

Auch die Ansicht einer Ergebnismenge, in der der Benutzer viele Ergebnisse gleichzeitig sah, stellte einen Nachteil gegenüber FB dar. Während bei FB meistens der vollständige Platz auf dem Bildschirm ausnutzt wird, um eine Ergebnismenge darzustellen, war dies in den früheren Implementierungen von PFB nicht der Fall. Die Ergebnismengen wurden jeweils nur als kleinere Knoten dargestellt, und der Benutzer hatte keine Möglichkeit, seinen Fokus bei Bedarf auf eine Menge auszurichten und diese Menge größer anzeigen zu lassen.

Ebenso war die mobile Nutzung stark eingeschränkt, da frühere Ansätze, die PFB realisierten, viel Bildschirmplatz einnahmen. Bisher wurde das Hauptaugenmerk bei der Implementierung von PFB auf die Verwendung der Benutzerschnittstelle auf PCs mit großen Bildschirmen und hohen Auflösungen optimiert; in den Zeiten des wachsenden Mobilmarktes muss jedoch ein Umdenken stattfinden. „Mobile first“ sollte auch für neuartige Benutzerschnittstellen die Devise sein.

Es wurden im Rahmen der Masterarbeit von Edit Kapcari [18] (Abschnitt 6.3.2) und auch im Rahmen des vom EIT Digital geförderten Projekts 3CIXTY von einer kleinen Gruppe von Mitarbeitern der technischen Universität „Politecnico di Milano“ Benutzerstudien durchgeführt, die verdeutlichen, dass nicht alle Benutzer die neuen Interaktionsmöglichkeiten auf Anhieb verstehen. Es müssen daher weitere Bemühungen unternommen werden, um die Erlernbarkeit und Verständlichkeit von PFB zu erhöhen, um somit auch technisch weniger affinen Nutzern die Verwendung von PFB zu ermöglichen.

Alle in dieser Sektion erwähnten Beschränkungen beziehen sich nur auf die Version von PFB, die vor dem Beginn dieser Forschungsarbeit bestand und nicht auf das Paradigma der Parallelen Exploration im Allgemeinen. Im Rahmen dieser Arbeit habe ich nach Möglichkeiten gesucht, um die angeführten Beschränkungen aufzuheben, das Interaktionsdesign zu überarbeiten und wesentliche, neue Erweiterungen hinzuzufügen, die in Kapitel 4 besprochen werden.

1.4 Welche Forschungsfragen ergeben sich daraus?

Wir haben gesehen, dass PFB viele Vorteile für die Benutzer bringt, aber die vor dieser Arbeit vorliegende Implementierung in vielerlei Hinsichten verbessert

werden kann. Dabei sollte das Interaktionsdesign überarbeitet und neue allgemeine Funktionalität hinzugefügt werden, die die im Abschnitt 1.2 beschriebenen Vorteile realisieren und die im Abschnitt 1.3 beschriebenen Beschränkungen aufheben. In diesem Abschnitt gehe ich auf die beiden Forschungsfragen ein, die in dieser Arbeit behandelt werden.

1.4.1 Welches Interaktionsdesign ist für PE geeignet?

Wie wir bereits in Abbildung 1.1 sehen konnten, verwendet PFB eine baumartige Struktur, die den Benutzern fremd war. Für neue Benutzer ist ein leicht erlernbares Interaktionsdesign essenziell, welches an bekannte Paradigmen anschließt. Überdies wird weder bei großen noch bei kleinen Bildschirmgrößen der Platz optimal ausgenutzt. Daher muss ein Interaktionsdesign ausgearbeitet werden, das den beiden beschriebenen Anforderungen genügt. Die Lösungsansätze zu dieser Forschungsfrage und die konkrete Implementierung wird in Kapitel 3 ausführlich besprochen.

1.4.2 Welche neue, allgemeine Funktionalität kann der PE hinzugefügt werden?

Die parallele Anzeige mehrerer Ergebnismengen wurde in Benutzerstudien als großer Vorteil von PFB festgestellt [26] (Abschnitt 5.4). Daher liegt es nahe, dieses Prinzip auch auf andere Aspekte anzuwenden als nur die facettierte Suche. Da das Paradigma von PFB noch weiter verallgemeinert und erweitert werden soll, haben wir stattdessen den Namen PE für „Parallele Exploration“ gewählt, der diese Tatsache besser widerspiegelt. Für die anderen, einzelnen Erweiterungen gab es jeweils auch spezifischere Gründe, auf die einzeln eingegangen wird.

Um eine Vorschau zu geben, möchte ich bereits an dieser Stelle die Erweiterungen umreißen, auf die ich in Kapitel 4 ausführlich eingehen werde. Die Volltextsuche ist ein vertrautes UI-Element, mithilfe dessen der Benutzer sehr schnell Resultate erhält. Die Unterstützung mehrerer Entitätstypen durch eine Benutzerschnittstelle wird in anderen Suchsystemen durch Pivoting realisiert und ermöglicht dem Benutzer ein Erkunden von heterogenen Datentypen. Ich werde darauf eingehen, warum diese Funktionen wichtig sind und wie sie sich in Form von Erweiterungen in das Paradigma von PE eingliedern lassen. Darüber hinaus bespreche ich den Entwurf und die Implementierung von Erweiterungen, die durch Rückmeldungen von Benutzern entstanden sind oder als Resultat von Benutzertests ersichtlich wurden: die Speicherung eines Explorationsbaums in Form eines Lesezeichens; die Synchronisation von Ergebnislisten zwischen mehreren Applikationen mithilfe einer

Wunschliste; QuickStart-Applikationen, die Antworten auf häufig gestellte Fragen liefern und die Realisierung von kontextspezifischen Tooltipps, die Funktionen der Benutzerschnittstelle erklären.

2 Verwandte Arbeiten

In diesem Kapitel gehe ich auf die verwandten Arbeiten, die von allgemeiner Bedeutung für meine Masterarbeit sind, ein. Sie sollen dabei unterstützen, die Masterarbeit besser eingliedern zu können, die historische Entwicklung der Parallelen Exploration näher zu erläutern, den aktuellen Stand der Forschung zu vermitteln und Ansätze zu liefern, die mir bei der Konzeption der in dieser Arbeit vorgestellten Erweiterungen geholfen haben.

2.1 Relevante Begriffe aus der Entscheidungspsychologie

Ein möglicher Ansatz, Benutzern mithilfe einer UI bei der Entscheidungsfindung zu helfen, ist „Winnowing“ [9]. In vielen Situationen steht der Benutzer vor der Entscheidung, ein (oder mehrere) Elemente aus einer unüberschaubaren Menge von Elementen herauszusuchen. In der Praxis lässt sich das nicht durch Erkunden jedes einzelnen Elementes bewerkstelligen, sondern es müssen spezielle Strategien angewandt werden, um das Problem zu lösen. Eine gängige Methode hierfür ist Winnowing, bei der der Benutzer die zu betrachtende Menge einschränkt. Dies kann durch Anwendung von spezifischen Filtermöglichkeiten, wie beispielsweise der facettierten Suche [27], geschehen oder durch Sortiermaßnahmen, bei denen der Benutzer sich nur auf die ersten Elemente fokussiert. Während dieser Ansatz die zu betrachtende Menge überschaubar macht und dem Benutzer dabei hilft, die richtige Auswahl zu treffen, kann es passieren, dass die bestmögliche Auswahl durch die Selektion bereits im Vorfeld eliminiert wurde.

Bei der Entscheidungsfindung aus einer unüberschaubar großen Menge von Elementen stellt Winnowing daher eine rentable Lösung dar, die den Nachteil hat, die bestmögliche Auswahl möglicherweise frühzeitig auszuschließen. Um dieses Problem abzuschwächen, verwenden Benutzer Backtracking-Strategien, bei der sie Filter zurücksetzen und alternative Suchpfade verfolgen. Traditionelle Suchsysteme bieten für das Durchführen von Backtracking-Strategien nur rudimentäre Hilfsmittel an: der Benutzer kann vor- und rückwärts im Verlauf navigieren oder gesetzte Filter

wieder entfernen. Wie Jameson et al. [13] (Kapitel 5.3) sagen, wäre gleichzeitiges Verfolgen mehrerer Suchstränge daher ein probates Mittel um den Nachteil von Winnowing zu relativieren.

Parallele Exploration ist eine Verallgemeinerung der facettierten Suche und unterstützt dadurch ebenfalls Winnowing, da es Filter- und Sortiermöglichkeiten anbietet um die zu betrachtende Menge überschaubar zu halten. Parallele Verfolgung mehrerer Suchpfade erlaubt es dem Benutzer, Backtracking-Strategien bei der Suche anzuwenden. In Abschnitt 4.3 werde ich darauf eingehen, wie Winnowing mit unterschiedlich granularen Ebenen von Selektionen realisiert werden kann.

2.2 Unterstützung von Exploration in mehrere Richtungen

Es existieren viele Ansätze für Benutzerschnittstellen, um das gleichzeitige Verfolgen mehrerer Suchpfade zu unterstützen. Ich werde zunächst auf relevante Forschungsarbeiten eingehen, die sich mit diesem Thema befasst haben und erläutern, wie diese mit unserem Ansatz von PE zusammenhängen und welche Erkenntnisse für uns nützlich sind.

2.2.1 Anzeige nebeneinanderliegender Ergebnismengen

In dem Artikel von Villa et al. [28] wird auf die Bewältigung von breitgefächerten und komplexen Suchaufgaben eingegangen, auf die Benutzer im Web stoßen. Sie stellen ein System vor, das den Benutzer dabei unterstützt, Aufgaben zu lösen, bei denen die Fragestellung nicht von vornherein vollständig klar ist und sich erst mittels Erkundung entwickelt. Im Sinne des Prinzips von „Teile und Herrsche“ unterstützt es den Benutzer bei seiner Suche, indem es ihm hilft, ein großes Problem in kleinere Aufgaben zu unterteilen, strukturiert anzuzeigen und zu lösen. Wie in Abbildung 2.1 zu sehen ist, können Ergebnisse mehrerer unterschiedlicher Suchanfragen nebeneinander angezeigt werden, sodass der Benutzer ausgehend von der ursprünglichen Aufgabenstellung in unterschiedliche Richtungen explorieren kann. Die Einzelansicht soll ergänzend dazu dem Benutzer die Möglichkeit bieten, eine Ergebnismenge groß darzustellen, indem es den gesamten verfügbaren Bildschirmplatz einnimmt.

Mithilfe einer durchgeführten Benutzerstudie wurde festgestellt, dass Parallelität dabei hilft, auf mehrere relevante Dokumente, Webseiten und Informationen zu

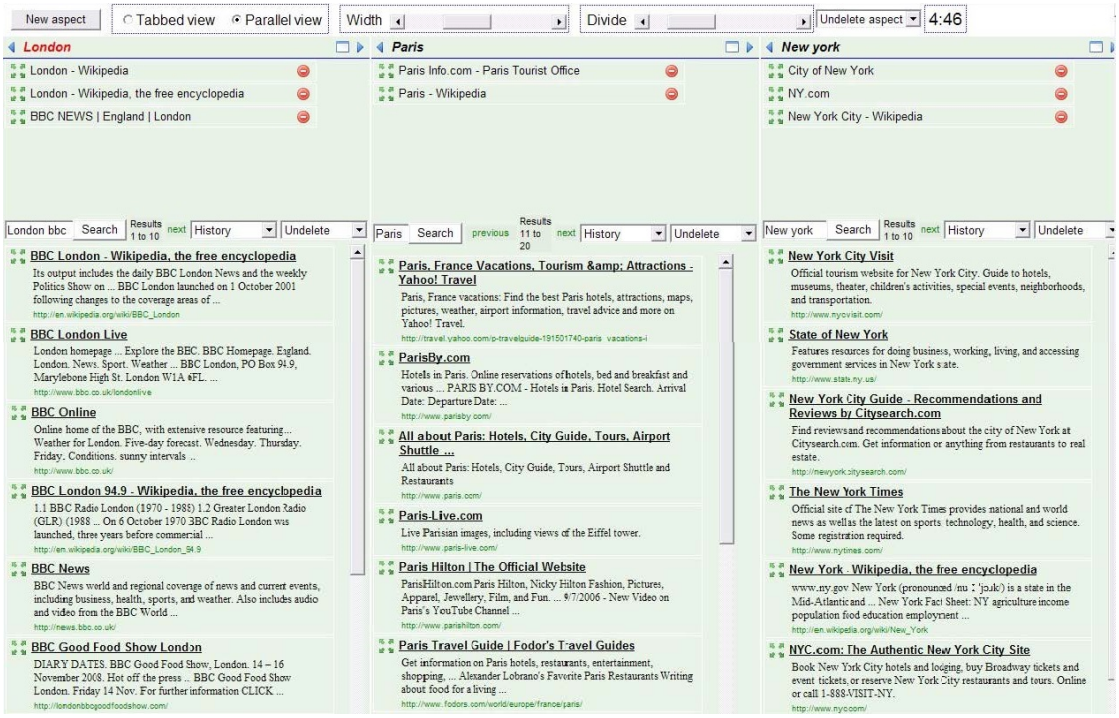

Abbildung 2.1: Paralleles Erkunden mehrerer Suchpfade mittels einer aspektbezogenen Benutzerschnittstelle

stoßen — im Gegensatz zu einer Einzelansicht. Es bringt den Benutzer dazu, sich bei seiner Suche einen breiteren Überblick zu verschaffen und hilft ihm effektiver zu sein.

Das System zeigt die Suchergebnisse mehrerer Suchanfragen in Spalten nebeneinander an und bietet die Möglichkeit, die Anzeige einzelner Ergebnismengen zu vergrößern. Dies stellt eine Gemeinsamkeit mit der aktuellen Implementierung von PE, auf die ich in Abschnitt 3.5 genauer eingehen werde.

2.2.2 Parallele Web Suche

Die Arbeit von Xu et al. [30] beruht auf Beobachtungen, dass Benutzer gerne parallel in mehrere Richtungen suchen. Dies soll eine Erweiterung zu den traditionellen Strategien der Websuche darstellen, in denen Benutzer mehrere Aufgaben gleichzeitig durchführen. Sie verwenden dabei mehrere Registerkarten oder ordnen ihre Browserfenster nebeneinander an, sodass die den gewünschten Effekt erzielen.

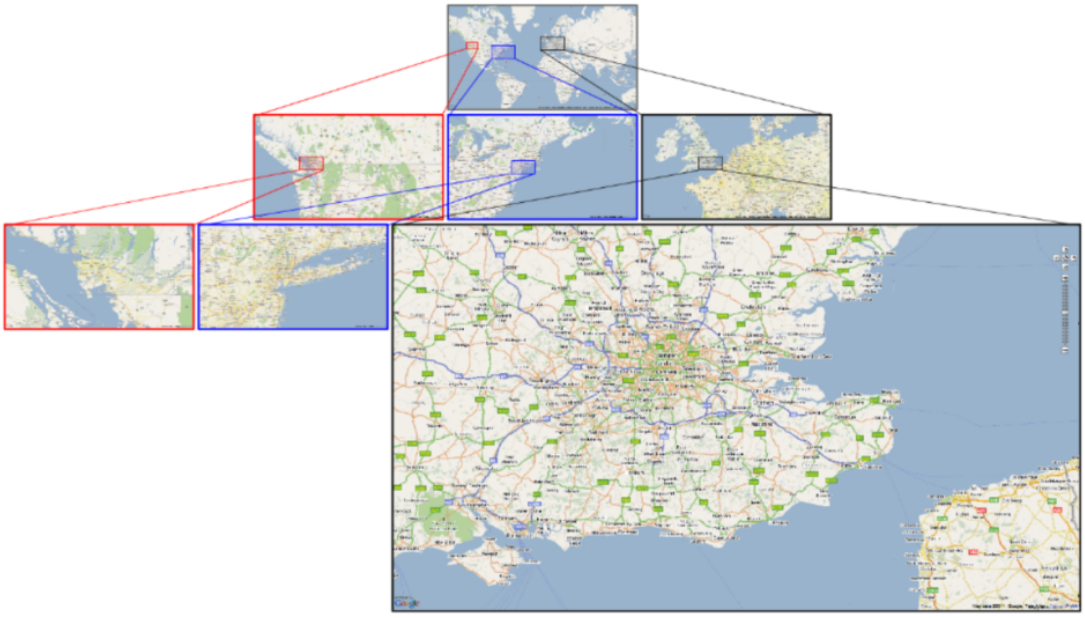

Abbildung 2.2: Multifokale Exploration mit Polyzoom anhand einer Weltkarte

Die vorgeschlagene Lösung erlaubt, dass Benutzer bei ihrer Websuche, mehrere Unterfenster in einem gegebenen Browserfester erstellen, um diese gleichzeitig betrachten zu können. Dabei werden mithilfe von Browsererweiterungen für Firefox und Chrome, Raster unterschiedlicher Größe erstellt, mit nicht überlappendenden Bereichen in denen verschiedene Webseiten angezeigt werden können.

In einer Benutzerstudie zeigen die Autoren, dass die Effizienz der 18 Teilnehmer (d.h. die benötigte Zeit, um die Aufgaben zu erfüllen), durch ihre Erweiterung um 26,3% erhöht werden konnte. Mithilfe von geschlossenen Fragen auf einer Likert-Skala von 1 („Ablehnung“) bis 5 („Zustimmung“) wurden drei Aspekte der Erweiterung abgedeckt. Dabei bewerteten die Teilnehmer das System als einfach zu bedienen mit einem durchschnittlichen Wert von 4.22, dessen Wirkung mit 3.89 und eine zukünftige Wiederverwendung mit 4.50. Daraus schlossen die Autoren, dass das System gut bei den Benutzern angekommen ist.

2.2.3 Polyzoom

Polyzoom ist die erste Benutzerschnittstelle aus dem Jahr 2012, die multifokale Exploration unterstützt [17]. Dieser neuartige Ansatz ermöglicht das hierarchische

Erkunden von Bildern, in dem der Benutzer mithilfe einer baumartigen Darstellung rekursiv selbstdefinierte Ausschnitte vergrößern kann. Wie man auf Abbildung 2.2 erkennen kann, wurden Teilbereiche einer Karte fokussiert, die sich wiederrum weiter unterteilen lassen. Somit ist es möglich mehrere unterschiedliche Ausschnitte parallel anzuzeigen und diese miteinander zu vergleichen. In gewisser Hinsicht entspricht dieses Paradigma dem vom PFB: aus einer Obermenge werden hierarchisch strukturiert jeweils Untermengen rekursiv gebildet, die miteinander verglichen werden können.

In einer Benutzerstudie wurden 12 Personen gebeten, bestimmte Ausschnitte auf Landkarten zu finden. Dabei konnte gezeigt werden, dass Personen die Polyzoom verwendet haben, im Vergleich mit dem traditionellen Zoom die gegebenen Aufgaben um 11 Prozent schneller lösen konnten. Es wurde gezeigt, dass multifokale Exploration in Bezug auf die Operation des Zoomens, Vorteile für den Benutzer haben kann.

Die Implementierung von Polyzoom an sich ist jedoch stark eingeschränkt; sie erlaubt nur das Explorieren von Bildern und die einzige Operation, um eine Untermenge bilden zu können, stellt das Zoomen dar. Diese Arbeit lieferte jedoch entscheidende Inspiration für die Entwicklung einer generischen Version von PFB, die es ermöglicht, auch textbasierte Daten zu erkunden.

2.2.4 Verfolgen mehrerer alternativer Suchpfade

Der Ansatz, mehrere alternative Suchpfade bei der explorativen Suche darzustellen [20], wurde auch von Bron et al. [3] im Bereich der Medienforschung verfolgt. Es wurde festgestellt, dass in den gängigen Systemen diese Möglichkeit zu wenig ausgenutzt wird und sie zeigten anhand einer Studie, welche Vorteile sogenannte „Subjunctive Interfaces“ bringen. Die Teilnehmer der Studie – die sich aus Medienforschern zusammensetzte – erhalten mehrere, unterschiedliche Ansichten bei der Erkundung eines Themas was zu einer höheren Diversität bei der Ideenfindung ihrer Forschungsfragen führt. Ein weiterer Vorteil wird bei der Analyse des Datenbestands ersichtlich, da vorhandene Muster leichter erkannt werden können: die Anzahl der Iterationen bei dem Prozess, in dem Daten erfasst, analysiert und anschließend die Forschungsfrage neu formuliert wird, wird durch eine leichtere Zugänglichkeit zu alternativen Suchpfaden reduziert.

Obwohl Systeme, die explorative Suche unterstützen, in Hinblick auf die Bedienbarkeit komplexer als traditionelle Suchsysteme sind und „Subjunctive Interfaces“ dem Benutzer noch mehr Funktionalität bieten, empfanden die meisten Benutzer das System als nicht schwer zu bedienen. Die Schlussfolgerung war, dass explorative

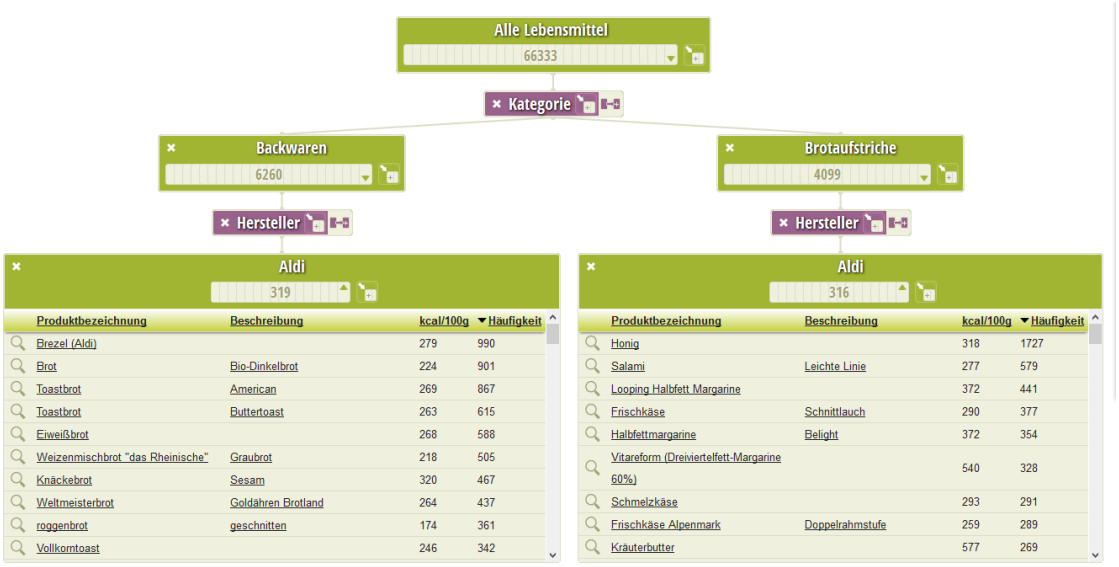

Abbildung 2.3: Erkundung der Lebensmitteldatenbank von My Miracle: Planung eines Frühstücks

Systeme von dieser Funktionalität auf Kosten leicht erhöhter Komplexität und verringerter Intuitivität profitieren können.

Der Grundgedanke von „Subjunctive Interfaces“ ähnelt dem von PE, die Umsetzung und Funktionsweise der Benutzerschnittstelle ist jedoch unterschiedlich und weist erhebliche Limitierungen auf: der Benutzer erhält bei der vorgestellten Implementierung eine erweiterte Version der facettierten Suche, bei der er nur genau zwei Suchpfade gleichzeitig verfolgen kann. Die aus der Studie hervorgehobenen Vorteile dieses Paradigmas stehen im Einklang mit unserem Ansatz von PE.

2.3 Frühere PFB Implementierungen

Im Folgenden werde ich die Entwicklung von PFB in den letzten Jahren beschreiben und definieren, wo meine Arbeit ansetzt. Ich werde erläutern, auf welche Domänen PFB bisher angewandt wurde und den jeweiligen Stand mithilfe von Bildschirmabzügen darlegen.

2.3.1 My Miracle PFB-Prototyp

Im Rahmen meiner Bachelorarbeit [26] griff ich den Gedanken von PFB auf und entwickelte eine allgemeine Benutzerschnittstelle, die im Gegensatz zu Polyzoom das Erkunden anderer Formen und Arten ermöglicht: das Explorieren einer Lebensmitteldatenbank. Während die grundlegende Struktur – wie bei Polyzoom – baumartig aufgebaut war, wurden mehrere Möglichkeiten des Filtrierens hinzugefügt. Es ist möglich nach Lebensmittelkategorie, Hersteller, Kalorien und weiteren Facetten zu unterteilen. In Abbildung 2.3 plant eine Person ihr Frühstück und möchte dabei passende Kombinationen von Backwaren und Brotaufstrichen finden, die bei Aldi erhältlich sind. Die Bachelorarbeit stellt das Fundament dar, auf das diese Masterarbeit aufbaut. Mit der Unterstützung mehrerer Entitätstypen und Erweiterungen, die die Funktionalität erhöhen soll, wird dieses Konzept angereichert und ausgebaut. Aus der ursprünglichen Idee der Parallelen Facettierten Suche, bei der lediglich das Filtern von Mengen möglich war, soll schließlich die Parallele Exploration möglich gemacht werden, die weitergehende Interaktionsmöglichkeiten zulässt.

2.3.2 EventMAP

EventMAP stellt eine weitere Benutzerschnittstelle dar, in der das Paradigma der Parallelen Facettierten Suche implementiert wurde. Dieses erfolgte durch Sven Buschbeck und stützte sich auf den von mir implementierten PFB-Prototyp. Da beide Benutzerschnittstellen laufend verbessert wurden, konnten diese von den Erfahrungen des jeweils anderen profitieren.

Die erste Version von EventMAP wurde im Rahmen des IESD-Workshops im Jahr 2012 [7] vorgestellt, in der die Veranstaltungen in Bezug auf Helsinki, welches „World Design Capital 2012“ war, erkundet werden konnten. In der zweiten Version im Jahr 2013 wurde die Benutzerschnittstelle weiter ausgebaut, an eine andere Datenquelle angebunden und zeigte Veranstaltungen in Paris.

EventMAP erhielt viel Rückmeldung von Experten und potenziellen Benutzern und zeigte dadurch das vorhandene Interesse an Paralleler Exploration. Daraus entwickelten sich auch Ideen in Bezug auf einige der Erweiterungen, die in dieser Arbeit besprochen werden. So stellte sich heraus, dass es einen Bedarf an Pivoting gibt, damit Benutzer auch zu anderen Entitäten navigieren können. Ebenso lieferten direkte Rückmeldungen von Forschern auf diesem Gebiet, wie beispielsweise von Marti Hearst, Erkenntnisse darüber, wie PFB weiter ausgebaut werden könnte. Das Fehlen typischer Vorteile der traditionellen facettierten Suche, dass viele Ergebnisse gleichzeitig sichtbar sind oder die beliebige Reihenfolge des Setzens von Filtern

Abbildung 2.4: Bildschirmabzug der PFB-Benutzerschnittstelle EventMAP aus dem Jahre 2013

sollte in PFB eingebaut werden, um den Benutzern eine gute Benutzererfahrung zu gewähren. Daraus entstand auf der CHI 2013 die Idee, mit einer gewohnten, facettierten Suche zu starten und den Benutzer allmählich an PE heranzuführen, die jedoch bis zu dieser Masterarbeit nicht weitergeführt wurde. Im Kapitel 3 über die Neugestaltung des Interaktionsdesigns gehe ich auf dieses Thema genauer ein.

2.3.3 Apps for your Car

Der Ansatz von PFB wurde auch in einer völlig anderen Domäne, als die bisher vorgestellten Beispiele bearbeitet: bei der Auswahl von mobilen Smartphone-Applikationen, die sich gut für eine Benutzung im Auto eignen. Ziel des PFB-Vorführmodells „Apps for Your Car“ [1] (siehe die bereits oben eingeführten Bildschirmabzüge in den Abbildungen 1.1 und 1.2) ist es, Benutzern dabei zu helfen,

aus mehreren tausend Apps die Richtige zu finden. Dieser Ansatz zeigte, dass sich das Paradigma von PFB auch für den Einsatz in einem App Store eignet.

In einer nachfolgenden Benutzerstudie, die durch Edit Kapcari [18] vorgenommen wurde, stellte sich heraus, dass PFB für den Benutzer kein vollständiger Ersatz von traditionellem FB ist, aber zumindest in einigen Fällen Stärken gegenüber FB aufweist. Insbesondere beim Vergleichen mehrerer Ergebnismengen haben die Teilnehmer der Benutzerstudie mit PFB die Aufgaben besser meistern können und erkannten darin Vorteile. Darüber hinaus wurden sie auch dazu animiert, mehr Zeit in den Vergleich zu investieren und diesen Vorteil auszunutzen.

Auf der anderen Seite wurde im direkten Vergleich zwischen FB und PFB erkannt, dass sich FB als Referenzsystem eignet und die Messlatte hoch anlegt; es bietet in der getesteten Fassung sogar gewisse Vorteile gegenüber PFB. So ist das Ändern der Filterreihenfolge einfacher, da sich im ersten Fall die Ergebnismenge in Echtzeit aktualisiert, wenn man einen Filter modifiziert. Des Weiteren konnten Benutzer die Aufgaben mithilfe von FB insgesamt schneller lösen, was teilweise darauf zurückgeführt wurde, dass die PFB-Benutzerschnittstelle für die Teilnehmer noch neu und eine gewisse Einarbeitungszeit notwendig war. Neben den anderen, bisher genannten Rückmeldungen ermutigte mich dieser Vergleich dabei, PE als eine Erweiterung von FB zu realisieren.

2.4 Systeme mit relevanter Funktionalität

Nach Verdeutlichung bisheriger Implementierungen von PFB, beschäftigt sich das nächste Kapitel mit den möglichen Interaktionserweiterungen, die sich gut in PE integrieren lassen. Insbesondere wird auf Pivoting eingegangen, das dem Benutzer ermöglicht zwischen beliebigen Entitäten zu wechseln.

2.4.1 Parallax

Bei Freebase Parallax geht es nicht – wie der Name in Zusammenhang mit dieser Arbeit vermuten lässt – um Parallele Exploration, sondern um das Navigieren von einer Ergebnismenge zur anderen, die unterschiedliche Entitätstypen haben kann. In dem Artikel von Huynh und Karger [12] wurde auf das mengenbasierte Navigieren eingegangen, welches eine neue Art der Interaktionen für Benutzer darstellt. Dieser soll graphenbasierte, semantische Daten im Web effizient erkunden können, indem ihm die Möglichkeit gegeben wird, von einer Menge zu einer anderen Menge von Elementen zu gelangen. Neben den typischen Filtermöglichkeiten,

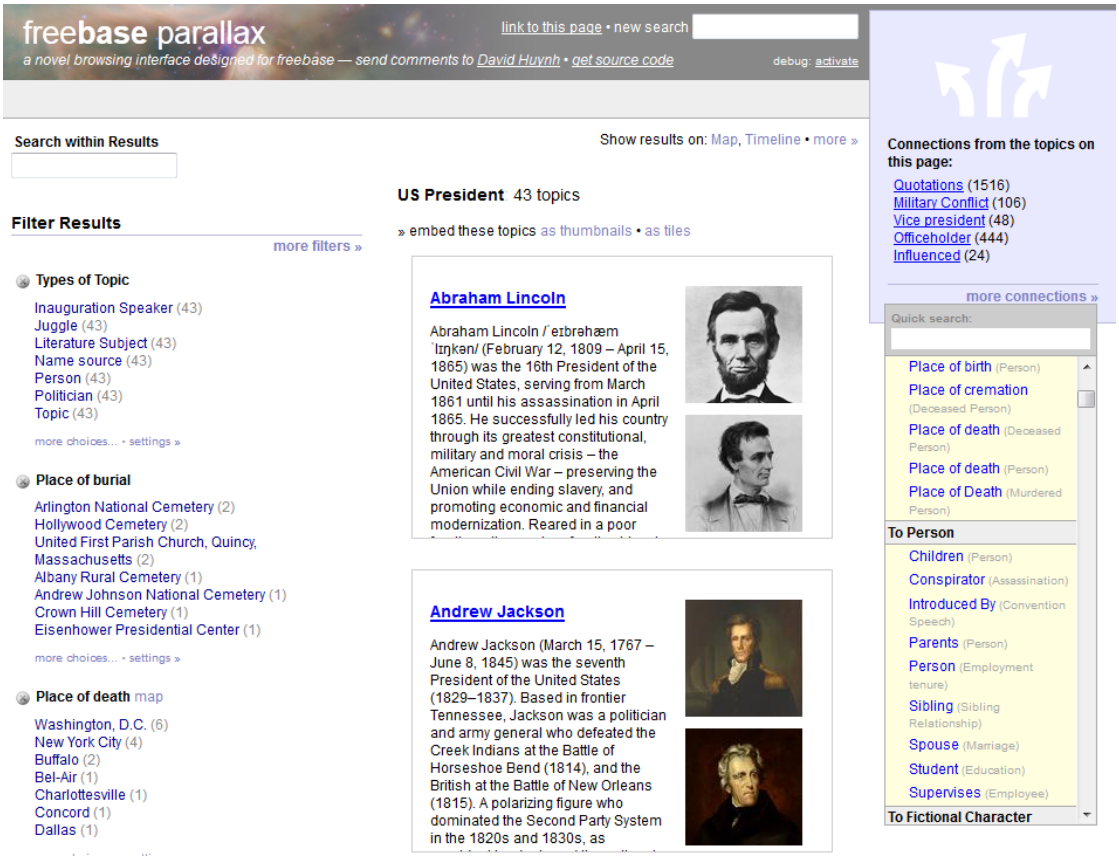

Abbildung 2.5: Parallax: Eine Benutzerschnittstelle, die mengenbasierte Navigation unterstützt

die aus der facettierten Suche bekannt sind, wird „Pivoting“ eingeführt — eine Interaktionsmethode, bei der man mithilfe von Verbindungen von einer Entität zu einer anderen gelangen kann.

Der Benutzer kann beispielsweise von einer Menge von Präsidenten mithilfe einer Relation zu einer vollkommen neuen Menge navigieren, nämlich deren Geburtsorte. Dieses Prinzip der Interaktion ermöglicht es dem Benutzer im Vergleich zur Operation des Filtrierens, bei der eine Ergebnismenge stets verfeinert und dadurch verkleinert wird aber stets denselben Entitätstyp besitzt, auch zu größeren Mengen anderen Typs zu navigieren. Insbesondere in Hinblick auf das Durchforsten von semantischen Daten, bei denen die Anzahl an unterschiedlichen Entitätstypen normalerweise höher ist als bei relationalen Daten, stellt dies einen Mehrwert dar.

Frühere Implementierungen von PFB ließen nur Filteroperationen zu, die gegebene

Mengen anhand eines Kriteriums verkleinern. Das Navigationsparadigma von Parallax, das Pivoting, lässt sich ebenfalls auf PFB anwenden und dadurch die Unterstützung mehrerer Entitätstypen durch eine einzelne Benutzerschnittstelle realisieren. In dem Abschnitt 4.2 werde ich dies näher konkretisieren.

2.4.2 Explorator

Eine weitere Benutzerschnittstelle, die die explorative Suche auf semantischen Daten unterstützt, ist „Explorator“ [8]. Das System ermöglicht Benutzern, die mit RDF vertraut sind, SPARQL-Endpunkte zu erkunden und Anfragen, wie beispielsweise „Finde alle Seen, die sich ausschließlich in Russland befinden“, mithilfe einer grafischen Benutzerschnittstelle zu stellen. Dieses System ermöglicht sogar das Formulieren von komplexeren Anfragen als unsere aktuelle Implementierung, welche zum Beispiel noch nicht die Schnittmenge von zwei gegebenen Ergebnismengen berechnen kann. Nach Meinung der Autoren richtet sich diese UI, aufgrund der Komplexität, hauptsächlich an Nutzer mit einem guten Verständnis von RDF, weshalb eine weite Verbreitung, die wir mit PE erreichen wollen, ausgeschlossen ist.

2.5 Quellen benutzerorientierter Rückmeldung

Weil meine Forschung im Rahmen eines multinationalen Projektes erfolgte, erhielt ich sowohl vor als auch nach der Realisierung der hier vorgestellten Verbesserungen verschiedene Typen von (direkter oder indirekter) Rückmeldung von Endbenutzern, Industrievertretern und anderen interessierten Personen zur Nützlichkeit und Bedienbarkeit von verschiedenen Versionen der betreffenden Implementierungen. Statt eigene Benutzerstudien durchzuführen, habe ich die Informationen aus diesen Quellen analysiert und berücksichtigt. Ich führe die Quellen hier auf, um an entsprechenden Stellen in der Arbeit darauf verweisen zu können.

1. Hervorgehend aus den Resultaten der Benutzerstudie meiner Bachelorarbeit [26, Kap. 4 und 5] des ersten PFB-Prototyps wurden die Wünsche der Teilnehmer analysiert.
2. Meinungen internationaler Experten wurden eingeholt auf dem Workshop Ïntelligent Exploration of Semantic Dataäuf der Tagung EKAW 2012 (s. [7]), bei dem eine frühe Version von PFB den Challenge-Preis gewann, sowie bei der tagelangen Vorführung einer etwas späteren Version auf der Tagung CHI 2013 in Paris (s. [6])

3. Die damalige Informatikstudentin der Universität Saarbrücken, Edit Kapcari, beschäftigte sich in ihrer Masterarbeit [18] empirisch und theoretisch mit dem Vergleich von FB und PFB und leitete Verbesserungsvorschläge für PFB ab.

4. Die von einer kleinen Gruppe von Mitarbeitern der technischen Universität „Politecnico di Milano“ (nachfolgend PoliMI genannt) durchgeführte Benutzerstudie mit 96 Personen in Mailand über ExplorMI 360 Web und ExplorMI 360 Mobile lieferte sowohl Aufschluss über die jeweiligen Benutzerschnittstellen und die Bedeutsamkeit einzelner Erweiterungen, als auch über das Zusammenspiel zwischen der Webapplikation und der mobilen Fassung.

5. Das im Rahmen vom Projekt 3cixty beauftragte User-Experience-Unternehmen ErgoSign lieferte seine kritische Expertenmeinung zu einigen unserer Implementierungen und Mockups.

6. Drei DFKI-Mitglieder des Projekts 3cixty (Catalin Barbu, Sarah Spirescu und Edit Kapcari) befragten und beobachteten Besucher von Mailand und von der EXPO 2015 sowie Hotelpersonal, um ihre Reaktionen auf verschiedene Aspekte von ExplorMI 360 in realistischen Situationen zu ermitteln.

7. Der Webdienst usertesting.com wurde durch dieselben 3cixty-Mitglieder in Anspruch genommen, der es ermöglichte, Bildschirm- und Audioaufzeichnungen von Personen bei der Benutzung von ExplorMI 360 zu erstellen, während sie laut dachten.

8. Frühere Versionen von PFB sowie die PE-Implementierung im Rahmen von 3cixty wurden ab Oktober 2014 in mehreren Ländern regelmäßig auf Fachmessen und Konferenzen sowie auf Treffen von EIT Digital vorgestellt. Daraus erhielten wir sowohl Rückmeldungen zu potenziellen Erweiterungen als auch zur Bedienbarkeit, die aus unterschiedlichen Blickwinkeln kamen: von typischen Endanwendern bis hin zu Geschäftsführern, die kommerzielles Interesse an der Parallelen Exploration für ihr Unternehmen hatten.

3 Anschluss an verbreitete Interaktionsdesigns

Dieses Kapitel beschäftigt sich mit der bestehenden Problematik, dass die bisherige Baumdarstellung von PFB für Benutzer fremdartig wirkt und sich nicht für alle Bildschirmgrößen eignet. Ich werde zunächst erläutern, wie die Verwendung natürlichsprachlicher Formulierungen zu einem besseren Verständnis der Benutzerschnittstelle beitragen und anschließend wie der Ansatz von „Responsive Design“ auf PE übertragen werden kann. Da die mobile Internetnutzung stets ansteigt und die Devise „Mobile First“ in aller Munde ist, sollen alle Interaktionen – vor allem für Smartphones – optimiert sein, um die Anzeige von mehreren Ergebnismengen zu ermöglichen. Aber auch die Anzeige auf großen Bildschirmen soll angepasst werden, um vom vorhandenen Platz zu profitieren.

3.1 Wie beschreiben wir die Verbesserungen?

Ich werde zunächst die Motivation für die jeweilige Erweiterung beschreiben, warum sie wichtig ist und welchen Nutzen sie für den Endbenutzer hat. Anschließend werde ich zeigen, welche Ergebnisse zur Bedienbarkeit und zur Nützlichkeit dem Design zugrunde lagen, die aus den im Abschnitt 3.1 beschriebenen Quellen für Rückmeldungen hervorgehen. In einem weiteren Schritt erkläre ich, wie die Erweiterungen für den Benutzer im Detail aussehen. Dabei werde ich nicht die gesamte Entwicklung beschreiben, sondern jeweils nur die neuste Version der Erweiterung hernehmen. Danach beschreibe ich die grundlegenden UI-Designentscheidungen die ich getroffen habe und wie diese durch Benutzerrückmeldung beeinflusst wurden. In diesem Unterpunkt stelle ich zu jeder Fragestellung mehrere Lösungsansätze vor, bespreche deren Vor- und Nachteile und erkläre, für welche Variante ich mich entschieden habe und wie gut das Resultat in der Endfassung funktioniert. Folgend gehe ich auf die technischen Möglichkeiten ein, die jeweilige Erweiterung umzusetzen. Auch hier beschreibe ich die erwägten Varianten und welche Vor- und Nachteile sich dadurch ergeben. Zum Schluss schildere ich zu jeder Erweiterung, wie diese implementiert wurde und gehe auf programmierspezifische Punkte ein.

3.2 Allgemeine Anforderungen

Die Grundidee hinter den Applikationen von 3CIXTY war es, dass der Benutzer zunächst mit EXPLORMI 360 Web (die in dieser Arbeit besprochene Webapplikation aus Abbildung 0.1, die PE unterstützt) seinen Aufenthalt in Mailand plant und anschließend unterwegs mit EXPLORMI 360 Mobile (eine Smartphone-Applikation, die vom 3CIXTY-Teilnehmer Telecom Italia entwickelt wurde) verfolgt. Informelle Rückmeldungen von Benutzern von Desktop-PCs zeigten, dass viele davon nicht gezwungen werden wollten, zu einer mobilen Applikation zu wechseln, beziehungsweise Nutzer von mobilen Geräten keine Webapplikation nutzen wollten, die nur für größere Bildschirmauflösungen optimiert war.

Sogar einige der Manager von EIT Digital selbst, die mehr als fünf Millionen Euro an Finanzierung für 3CIXTY genehmigt hatten, wollten die Webapplikation ausschließlich auf ihren mobilen Endgeräten nutzen. Daher wurde es notwendig, dass die UI sämtliche Voraussetzungen für mobile Geräte erfüllt.

Insbesondere die Verwendung eines responsiven Designs, das die Darstellung des Inhalts an die tatsächliche Bildschirmgröße anpasst, sowie die Verbesserung der Interaktionen mit mobilen Geräten, gehen aus dem gesammelten Feedback hervor und müssen bei den in diesem Kapitel beschriebenen Entwürfen und Implementierungen beachtet werden.

3.3 Verwendung natürlichsprachlicher Formulierungen in der UI

Um die Verständlichkeit des präsentierten Inhalts in der Benutzerschnittstelle zu fördern, ist es nicht nur wichtig diesen strukturiert anzuzeigen, sondern ihn auch gut zu beschreiben. Dies erfolgt anhand von Breadcrumbs: Textfelder, die dem Benutzer Auskunft darüber geben, in welchem Zustand der Exploration er sich gerade befindet beziehungsweise die Verzweigungen im Explorationsbaum beschreiben. Die Verwendung natürlichsprachlicher Formulierungen in Breadcrumbs soll den Inhalt in ganzen Sätzen erklären, damit auch Benutzer ohne Vorkenntnisse über die UI die dargestellten Inhalte verstehen und interpretieren können. Außerdem wird der Kontext solcher Formulierungen besser durch automatische Übersetzer erkannt, wodurch die Benutzerschnittstelle mithilfe eines Übersetzungsdienstes in mehreren Sprachen angeboten werden kann.

3.3.1 Welche Ergebnisse zur Bedienbarkeit und zur Nützlichkeit lagen dem Design zugrunde?

Prof. Marti Hearst schrieb in ihrem Buch „Search User Interfaces“ [11] (Abschnitt 7.2), dass die Verwendung von Breadcrumbs bei Suchsystemen, wie beispielsweise die facettierte Suche, essenziell ist: sie soll dem Benutzer dabei helfen die Struktur der Suche zu verstehen. Dafür ist es wichtig, die Breadcrumbs detailreich zu beschreiben und argumentiert für die Verwendung von „Structural Breadcrumbs“, die dem Benutzer, neben der Möglichkeit die Anfrage zu erweitern oder zu verkürzen, Informationen über gesetzte Filter geben soll. Die in PE eingesetzten, mit natürlicher Sprache ausformulierten Sätze, basieren auf dem Hintergedanken, mit Breadcrumbs den Stand der Erkundung zu beschreiben.

3.3.2 Wie sieht die gewählte Lösung für den Benutzer im Detail aus?

Abbildung 3.1 zeigt einen vom Benutzer aufgespannten Explorationsbaum, bei dem natürlichsprachliche Formulierungen dem Benutzer dabei helfen, die Struktur seiner Exploration sofort erkennen zu können. Der Inhalt eines Knotens wird durch die Breadcrumbs zwischen diesem und dem Wurzelknoten beschrieben. In dem Falle des linken, untersten Knotens in der Abbildung 3.1 lautet die Beschriftung beispielsweise: „All places that have the category food that include the word or phrase pizza; hotels near at least one of these places within 0 - 500m straight line distance.“

3.3.3 Was sind die grundlegenden UI-Designentscheidungen?

Die grundlegende Entscheidung bei der Verwendung von Breadcrumbs ist es, ob diese kompakt oder weitschweifig formuliert werden sollen. Beide Möglichkeiten haben ihre Stärken und Schwächen, auf die ich näher eingehen werde. Daher muss in unserem Fall eine Abwägung stattfinden.

Die kompakte Darstellung hat den Vorteil, dass man hintereinandergereihte Breadcrumbs einer linearen Exploration sogar nebeneinander anzeigen könnte um weiteren Bildschirmplatz zu sparen. Die weitschweifige Darstellung hat jedoch andere Stärken: insbesondere Funktionalitäten wie einfaches oder multiples Pivoting (siehe Abschnitt 4.2 für die Erklärung dieser Erweiterung) sind mit einem längeren, beschreibenden Text für den Benutzer leichter zu verstehen — in vielen Fällen stellt es eine große Herausforderung dar, die durchgeführten Benutzerinteraktionen kurz

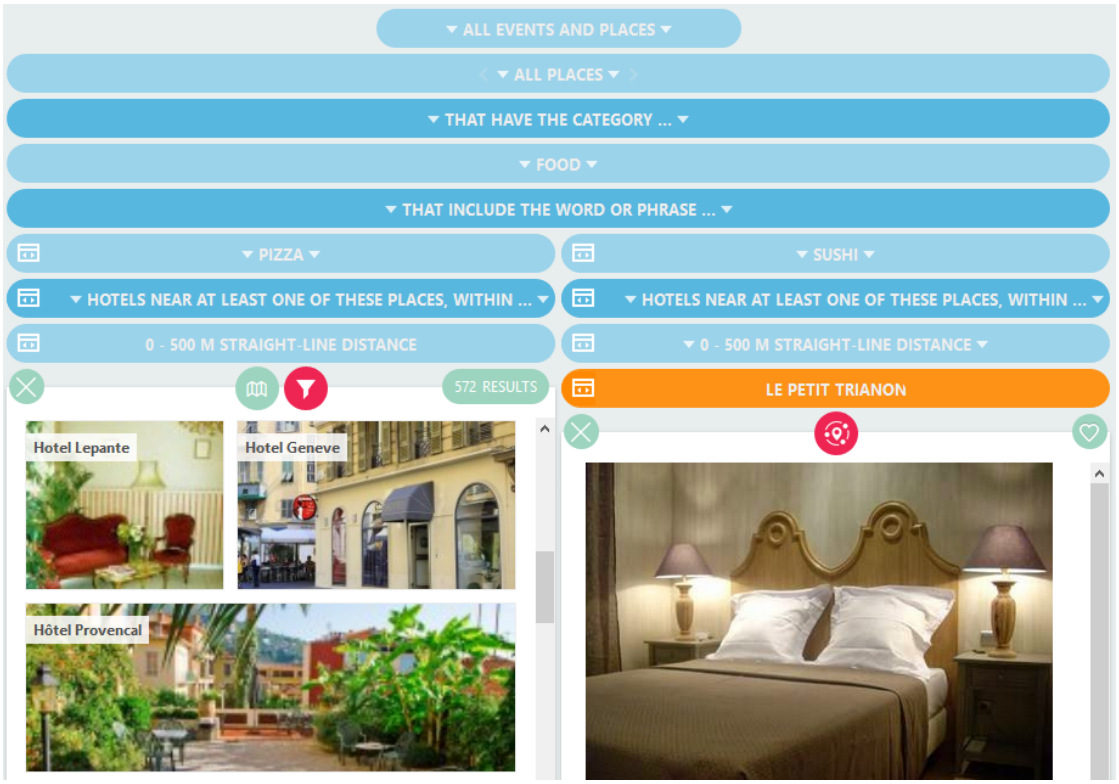

Abbildung 3.1: Natürlichsprachliche Formulierungen beschreiben die Struktur des Explorationsbaums

und knapp darzustellen. Die Pivotingoperation, die in Abbildung 3.1 mit „Hotels near at least one of these places“ beschriftet ist, erklärt diese Relation sehr gut; es ist schwer, dafür ein passendes Pendant zu finden, das diesen Sachverhalt kurz und bündig erklärt und gleichzeitig für den Benutzer leicht zu verstehen ist.

Der Grund, weswegen ich mich für die Realisierung der weitschweifigen Formulierungen entschieden habe, ist die essenzielle Anforderung an die Benutzerschnittstelle, in möglichst vielen Sprachen verfügbar zu sein. Wenn es reichen würde nur wenige Sprachen zu unterstützen, könnte man diese, inklusive der Übersetzungen, in den Code integrieren; ist die Anzahl der Sprachen groß, so kommt man nicht um den Einsatz eines Übersetzungsdienstes herum. Ich habe mich für den Einsatz von „Google Translate“ entschieden und wir haben schnell feststellen müssen, dass kurze Formulierungen teilweise schlecht übersetzt werden: so wurde beispielsweise das englische Wort „Places“ ins Deutsche mit „Stellt“ übersetzt. Liefert man der API jedoch einen größeren Kontext, so lässt sich dieses Problem reduzieren, da der

Abbildung 3.2: Übersetzung der weitschweifigen Formulierungen der Benutzerschnittstelle ins Französische (FR) und Deutsche (DE)

Sinn und der Zusammenhang durch den Dienst besser interpretiert werden kann. Abbildung 3.2 zeigt, dass die Instanziierung des vorigen Beispiels für den Benutzer ins Französische und Deutsche übersetzt werden kann. Übersetzungen aus dem Englischen in andere Sprachen funktionieren normalerweise gut, das Deutsche mit seiner komplexeren Grammatik stellt hierbei eine größere Herausforderung dar. Der Sinn lässt sich jedoch, wie in Abbildung 3.2 (DE) zu sehen ist, gut erfassen. Selbst die Übersetzungen in Hindi und viele andere, der in Indien verwendeten Sprachen, wurde von einem indischen Geschäftspartner als ausreichend bewertet.

3.4 Version A des responsiven Designs: Jeweils nur ein Ergebnisfenster sichtbar

Weil das neue Design, um Responsivität in die Benutzerschnittstelle einzuführen, viele Aspekte hat, teile ich es in drei Versionen auf, wobei jede Version nach dem beschriebenen Schema erläutert wird. Die Implementierung wird allerdings erst nach der Betrachtung von Version C beschrieben, um die logischen Zusammenhänge auf einen Blick zu haben und Redundanzen zu vermeiden.

Version A ist der erste Schritt zur responsiven Fassung der PE-Benutzerschnittstelle und beschreibt die Annäherung des Paradigmas PE an FB. Dabei wird zwar nur ein Ergebnisfenster zu einem gegebenen Zeitpunkt angezeigt und nebeneinanderliegende Knoten kommen nicht vor, aber der Benutzer erhält Zugriff auf Parallelität durch hin- und herschalten von Ergebnisfenstern innerhalb eines zusammengefassten Knotens.

3.4.1 Warum ist diese Erweiterung wichtig?

Die meisten Benutzer sind die traditionelle facettierte Suche [10] gewohnt, da sie im Internet sehr weit verbreitet ist und auf übliche Entwurfsmuster zurückgreift, beziehungsweise neue Entwurfsmuster durch ihre große Verbreitung bekannt gemacht hat [27] [11] [24]. Um das neue, innovative Bedienkonzept von PE für neue Benutzer leicht verständlich zu machen, ist es ein guter Ansatz, sie durch die gewöhnliche facettierte Suche langsam an die neuen Interaktionsmöglichkeiten heranzuführen. Ziel dieser Erweiterung ist es, dass eine Abbildung zwischen dem entstehenden Explorationsbaum mit traditionellem FB und dessen Anzeige mithilfe von PE gefunden wird: der Benutzer soll im Optimalfall eine facettierte Suche bedienen und sich auf Knopfdruck den Explorationsbaum parallel anzeigen lassen können. Die baumartige Struktur der Suche soll durch den Wechsel zur parallelen Darstellung in den Vordergrund rücken und dem Benutzer die Vorteile von PE dadurch leicht und verständlich näherbringen.

3.4.2 Welche Ergebnisse zur Bedienbarkeit und zur Nützlichkeit lagen dem Design zugrunde?

Wir erhielten im Laufe der Zeit viele Rückmeldungen über die Benutzerschnittstelle von Personen, die das grundlegende Prinzip von PE zwar gut, aber deren Darstellung zu neu und fremdartig fanden. Die Benutzerstudien auf usertesting.com sowie Anmerkungen von ErgoSign (Abschnitt 3.1) zeigten, dass es notwendig ist, dem Benutzer eine Darstellung der UI zu bieten, die vertrauter und gewohnter zu bedienen ist. Auch die Rückmeldung zu unserer Benutzerschnittstelle der Jury-Vorsitzenden des CeBIT Innovation Awards, Frau Prof. Joost, zielte darauf ab: „Sie machen es komplizierter statt einfacher“.

3.4.3 Warum haben wir es nicht früher gemacht?

Die Idee, dass der Benutzer zunächst mit der facettierten Suche anfängt und schrittweise zu PE wechselt, entstand vor längerer Zeit: auf der CHI 2013 gingen die Vorschläge von Marti Hearst in diese Richtung, aber es wurde keine konkrete Lösung, beziehungsweise kein konkreter Realisierungsansatz vorgeschlagen. Auch bei dem Wettbewerb „3CIXTY App Challenge“ in einer Einreichung von Kai-Dominik Kuhn und mir im November 2014 haben wir diese Idee aufgegriffen, jedoch keine gute Alternative zum alten Design gefunden. Dieser Ansatz wurde auch von ErgoSign durchleuchtet, die starke Befürworter von FB sind, ohne einen guten Weg zu finden, wie ein flüssiger Übergang zwischen FB und PE aussehen könnte. Wie wir sehen, war der Wunsch nach einem hybriden Ansatz von mehreren Seiten vorhanden, aber niemand konnte zu diesem Problem eine passende Lösung finden.

Die Schlüsselideen für die Verwirklichung von diesem Ansatz kamen erst Anfang dieses Jahres, als wir von der Baumdarstellung in der Benutzerschnittstelle weggegangen sind und stattdessen eine „tabellarische“ Darstellungsform gewählt haben, die die volle Bildschirmbreite ausnutzt. Wir gingen zunächst von mobilen Applikationen und dem Gedanken von „mobile first“ aus, bei dem es essenziell ist, auf kleineren Displays mit dem verfügbaren Platz auszukommen. Durch das Kombinieren von einzelnen Spalten, welches das Zusammenfassen von mehreren nebeneinanderliegenden Knoten darstellt, wurde die Lösung ermöglicht, die das neue Interaktionsdesign widerspiegelt.

3.4.4 Wie sieht die gewählte Lösung für den Benutzer im Detail aus?

Die Abbildung 3.3 zeigt, wie ein Benutzer die PE-Benutzerschnittstelle genauso benutzen kann, wie die traditionelle facettierte Suche. Zunächst sucht der Benutzer nach Restaurants, die eine überdurchschnittliche Bewertung aufweisen (a).[1] Dabei setzt er – wie er es wahrscheinlich von der traditioneller facettierten Suche gewohnt ist – einen Filter nach dem anderen. Es entsteht ein linearer Suchpfad und der Benutzer sieht nur eine Ergebnismenge. Ist er mit seiner vorigen Auswahl von Restaurants unzufrieden oder möchte nun stattdessen Bars angezeigt bekommen, müsste er nach dem Stand bisheriger Interaktionsmöglichkeiten einen Teilbaum neu aufbauen. Die facettierte Suche lässt es zu, die Filter in beliebiger Reihenfolge anzugeben; daher wird bei einem wiederholten Setzen des Filters für die Kategorie

[1] Der Benutzer kann jederzeit auf ein Element klicken, um zu eine Schnellansicht mit weiteren Details zu erhalten.

auf „Bars“ der bisherige Pfad geändert, wie in (b-d) zu sehen ist. In der Darstellung wird die Kategorie „Restaurants“ durch „Bars“ ersetzt, im Hintergrund wird ein weiterer Suchpfad aufgespannt, der zwar nicht parallel angezeigt wird aber vom Benutzer jederzeit zugänglich ist. Er kann nun wieder zurück zu Restaurants wechseln, da im Gegensatz zur facettierten Suche die Ergebnisse früherer Filteroperationen nicht verworfen werden, wenn einer der Filter ersetzt oder entfernt wird. Vom rechten Zustand ist es ihm möglich zurück in den linken zu gelangen, indem er die schwarze Überschrift mit „Restaurant“ anklickt.

3.4.5 Was sind die grundlegenden UI-Designentscheidungen?

Um die Benutzerschnittstelle so nah wie möglich der traditionellen facettierten Suche anzupassen, müssen Abbildungen der grundlegenden Interaktionselemente von FB nach PE gefunden werden. Breadcrumbs geben dem Benutzer bei FB Rückmeldung darüber, welche Filter gerade ausgewählt sind, und ermöglichen ihm, Teile davon zu entfernen. Baut der Benutzer einen linearen Explorationsbaum in PE (d.h. alle Filter- und Pivotingoperationen bauen aufeinander auf, ohne eine Verzweigung im Explorationsbaum zu erzeugen), so würden die obenliegenden Elternknoten diese Breadcrumbs abbilden.

Die Frage die sich hierbei stellte, war, ob diese (wie bei PE) übereinander oder (wie bei FB) nebeneinander angezeigt werden sollten. Der Vorteil sie nebeneinander anzuzeigen, wäre, dass die Benutzerschnittstelle anfänglich für den Benutzer vertrauter aussehen würde. Andererseits würde sich bei einem Wechsel zur parallelen Ansicht in der Benutzerschnittstelle mehr ändern müssen, um die nebeneinanderliegenden Breadcrumbs in das Paradigma von PE eingliedern zu können. Wie wir in Abschnitt 3.3 gesehen haben, sind die Formulierungen wortreich, sodass sie ohnehin mehr horizontalen Platz verbrauchen und zur Erörterung der aktuellen Erkundung beitragen. Ich habe mich dafür entschieden, die Breadcrumbs übereinander anzuzeigen, da ich es für vorrangig hielt, die Abbildung zwischen FB und PE möglichst einfach zu gestalten. Außerdem wurde eine Funktion eingebaut, die vertikal angeordnete Breadcrumbs zusammenfasst: ausgehend von der Idee eines Rollladens können Benutzer die Breadcrumbs (mit einem Klick auf die Symbole beziehungsweise) auf- und zuklappen, sodass sie nicht viel Platz verbrauchen (siehe Abbildung 3.4).

Man könnte dem Benutzer die Möglichkeit geben, mit traditionellem FB anzufangen. Auf einer getrennten FB-Benutzerschnittstelle, wie beispielsweise die aus Abschnitt 1, kann er seine Erkundung anfangen und zu jedem Zeitpunkt zu einer PE-Benutzerschnittstelle wechseln. Es hätte den Vorteil, dass diese UI alle nativen

Abbildung 3.3: Vertrauter Einstieg in die PE-Benutzerschnittstelle durch Facettierte Suche

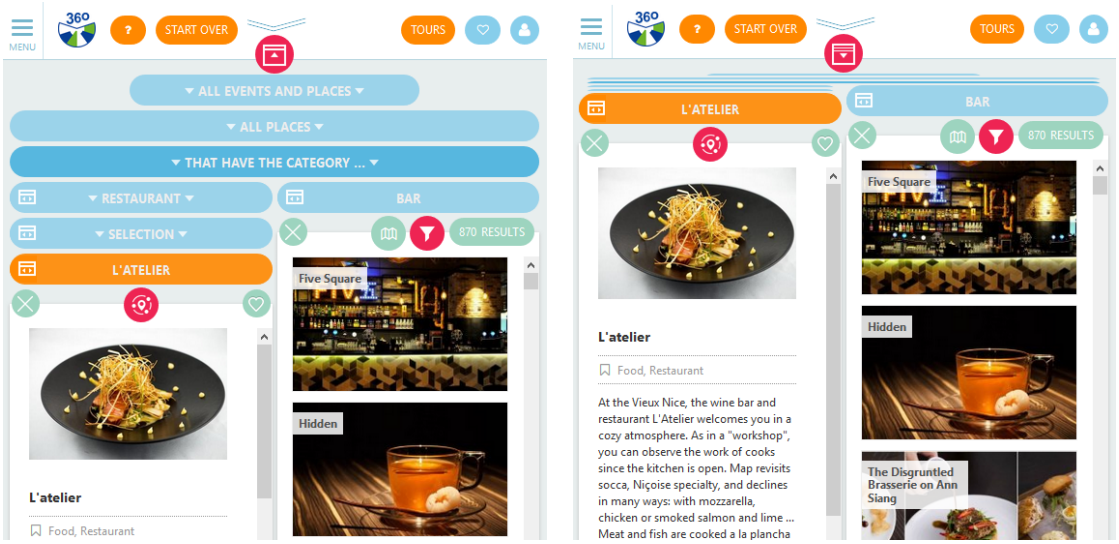

Abbildung 3.4: Vertikales Zusammenklappen mehrerer Breadcrumbs: der Initialzustand (links), dessen kompakte Darstellung (rechts)

Möglichkeiten von FB bezüglich Interaktion bereitstellt. Es wäre machbar, die durchgeführten Nutzerinteraktionen zu dokumentieren und auf Wunsch ein valides Lesezeichen (siehe Abschnitt 4.5 für einen möglichen Ansatz, dies zu realisieren) zu erstellen, das in PE aufgerufen werden kann. Neue Nutzer könnten gedanklich eine Brücke bilden zwischen dem, was sie in FB exploriert haben und was sie anschließend in PE sehen. Der Nachteil hierbei wäre, dass die beiden Systeme in gewisser Hinsicht getrennt wären und sich die Personen nicht an die grobe Struktur von PE gewöhnen können, während sie FB nutzen. Ein zusammengefasster Knoten zeigt stets den aktuellen Inhalt des aktiven Knotens und bietet die Möglichkeit, zu einem benachbarten Knoten zu wechseln. Bei einem solchen Wechsel muss auch dafür gesorgt werden, dass die Kindesknoten ebenfalls gewechselt werden. Wenn diese Knoten ebenfalls zu einem zusammengefasst sind, so stellt sich die Frage, welches der Kinder gerade in den aktiven Bereich gerückt wird. Die naive Implementierung, die ohne großen Aufwand zu realisieren wäre, sieht vor, jeweils das erste Kind zu nehmen. Dies hätte einen entscheidenden Nachteil, wenn er zwischen „Restaurants“ und „Bars“ wechselt und dabei die beiden rotgestrichelten Knoten auf Abbildung 3.5 (a) vergleichen möchte, da er jedes Mal auch zum passenden Kindesknoten navigieren müsste.[2]

[2] Abbildung 3.5 (a) stellt die bisherige Exploration des Benutzers in der parallelen Ansicht dar, auf die ich im nächsten Abschnitt näher eingehen werde.

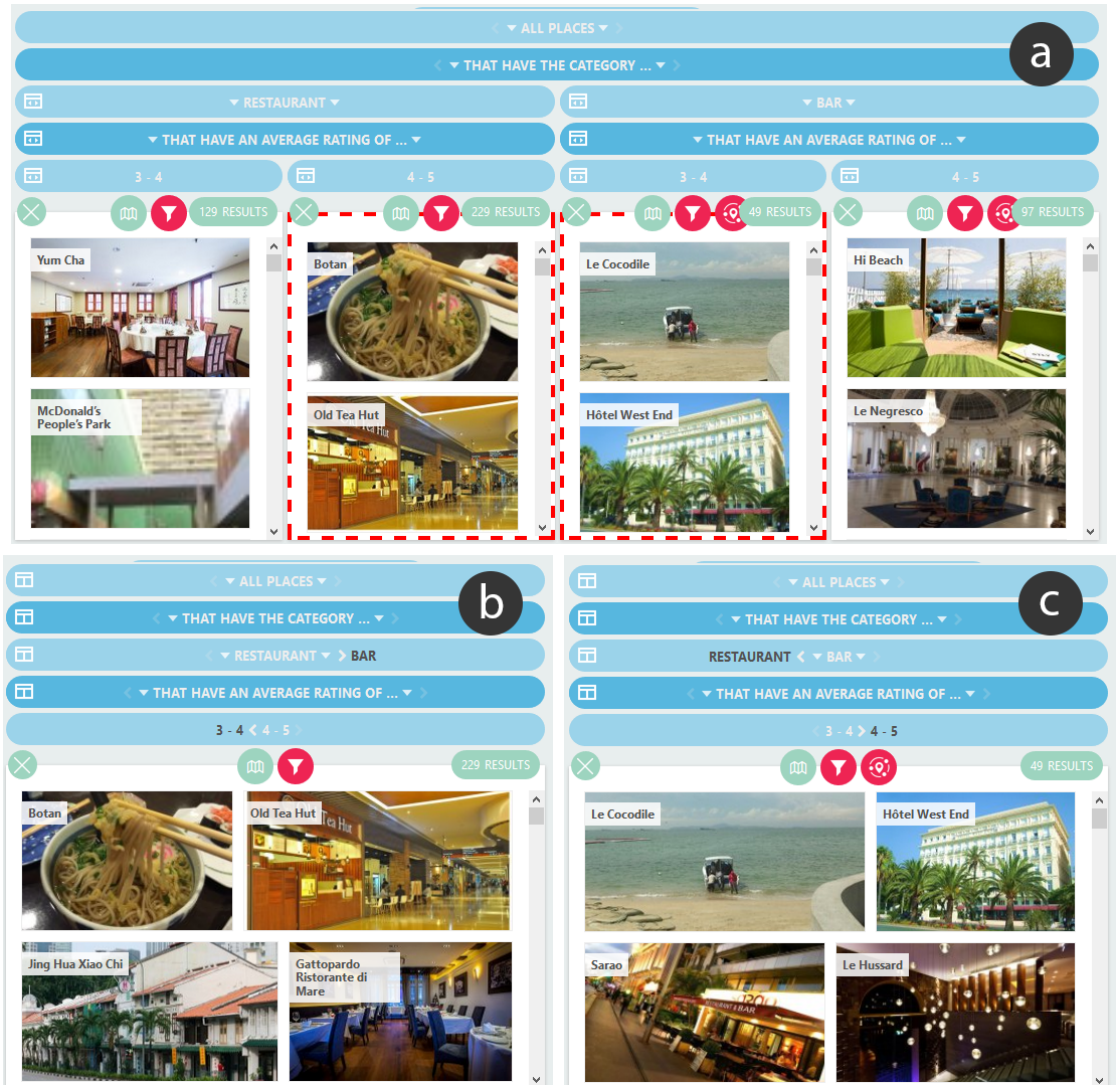

Abbildung 3.5: Speicherung des zuvor ausgewählten Pfades beim Wechseln eines Filters sorgt für eine persistente Ansicht

Der zweite, komplexere Ansatz umgeht dieses Problem. Er sorgt für eine persistente Ansicht nach einem Wechsel und behält dabei den zuvor ausgewählten Pfad bei; so würde der Benutzer bei hin- und herschalten zwischen Restaurants und Bars zwischen den Zuständen, die in (b) und (c) dargestellt sind, alternieren. Hierfür muss ebenfalls gespeichert werden, welcher Knoten einer zusammengefassten Menge von Knoten vom Benutzer selektiert wurde und nach einem jeweiligen Wechsel beibehalten werden soll. Trotz des höheren Aufwands habe ich mich für die zweite

Version entschieden, um Benutzern stets den von ihnen ausgewählten Knoten in einem zusammengefassten Knoten anzuzeigen und auch in den beschriebenen Spezialfällen das Vergleichen mehrerer Mengen einfacher zu gestalten.

3.5 Version B: Auch gleichzeitige Darstellung von Ergebnisfenstern

Im Abschnitt 1.2 wurden die Vorteile der Parallelität für Benutzer besprochen. Die Tatsache, Informationen nebeneinander sehen zu können bietet einen weiteren, entscheidenden Vorteil, wenn man diese vergleichen möchte: der Benutzer behält die entsprechenden Informationen im Blick und kann mit schnellen Augenbewegungen zwischen den Informationsdarstellungen beliebig wechseln. Ist er jedoch darauf angewiesen, mit einer Schaltfläche zwischen den beiden Ansichten zu wechseln, muss er stets nicht sichtbare Informationen im Kurzzeitgedächtnis behalten. Insbesondere beim Vergleich ähnlicher Elemente (beispielsweise zwei ähnliche Hotels, die in zwei Individualknoten gezeigt werden), bei denen sich jeweils nur wenige Attribute unterscheiden, ist es schwer, diese durch häufiges Umschalten der Ansichten zu identifizieren — diese Beziehung lässt sich viel schneller erkennen, wenn die Elemente nebeneinander gezeigt werden.

Die gleichzeitige Darstellung nebeneinanderliegender Elemente beeinflusst, wie oben beschrieben, die Effizienz des Entscheidungsprozesses, in dem sich der Benutzer befindet. Wie aus dem Artikel von Bazerman et al. [2] hervorgeht, werden außerdem auch die Entscheidungskriterien und das Ergebnis des Prozesses dadurch beeinflusst, dass bei gleichzeitiger Darstellung auch andere Vorgehensweisen für den Benutzer möglich sind.

3.5.1 Wie sieht die gewählte Lösung für den Benutzer im Detail aus?

In Abschnitt 3.4 haben wir gesehen, dass Benutzer zwischen zwei Kategorien hin- und herwechseln können. Ausgehend vom Zustand aus in Abbildung 3.6 (a) kann der Benutzer das Symbol auswählen, um die beiden Alternativen – Restaurants und Bars – gleichzeitig zu sehen; das Resultat ist (b) ersichtlich.

Der Benutzer kann sich zu jedem Zeitpunkt dafür entscheiden, einen Teilbaum der Gesamtansicht zu vergrößern und diesen zu fokussieren, indem er auf den Knopf für die „Einzelansicht“ des Knotens klickt. Benachbarte Knoten werden hierbei

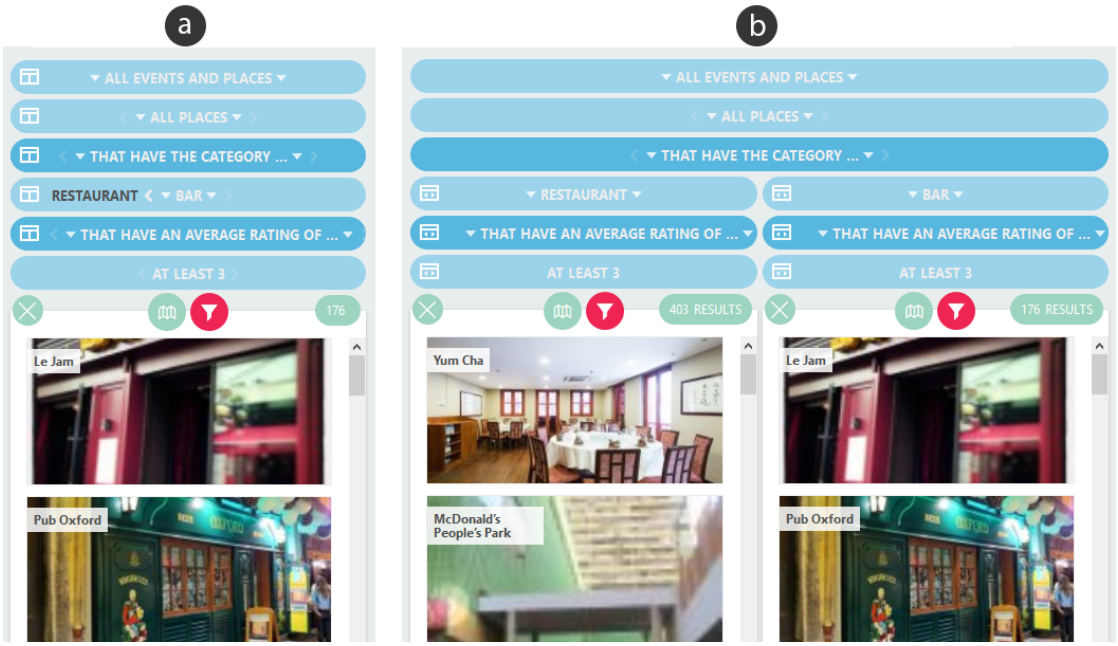

Abbildung 3.6: Zusammengefasste Ansicht (a); Parallele Ansicht (b)

mit dem Ausgewählten zusammengefasst und sind anschließend über die Kopfzeile des Knotens erreichbar. Bei diesem Vorgang wird die maximal verfügbare Breite für das ausgewählte Ergebnisfenster und dessen Elternknoten ausgenutzt, da davon auszugehen ist, dass dieser Teilbereich für den Benutzer im Vordergrund steht. Um die oben beschriebene Aktion wieder rückgängig zu machen, muss der Benutzer auf den Knopf für die „Parallelansicht“ des Knotens „Bar“ klicken und gelangt wieder aus dem Zustand von Abbildung 3.6 (b) in den Linken (a).

3.5.2 Was sind die grundlegenden UI-Designentscheidungen?

Die Darstellung der Kopfzeile eines zusammengefassten Knotens basierte auf Überlegungen zu Bedienkonzepten, die in aktuellen mobilen Applikationen verwendet werden, in denen responsives Design oftmals benötigt wird.[3] Die Navigation zwischen den jeweiligen zusammengefassten Knoten kann entweder mit Pfeilen realisiert werden, die auf die Existenz weiterer, versteckter Knoten hindeuten, mithilfe derer der Benutzer zwischen ihnen wechseln kann, oder es kann eine Art Vorschau auf die

[3]Das Buch „Mobile Design Pattern Gallery“ [23] lieferte eine nützliche Übersicht von Entwurfsmuster.

Bezeichnung des Vorgängers und Nachfolgers geboten werden. Obwohl die zweite Option die Benutzerschnittstelle durch die Anzeige mehrerer Bezeichner vielleicht überladen könnte, habe ich mich dazu entschlossen diese dennoch anzuzeigen, da es dem Benutzer wertvolle Informationen liefert, wohin er navigiert. Es bietet eine gute Orientierungshilfe, um die Struktur von PE besser im Blick zu haben.

Nachdem die Einzelansicht eines Knotens aktiviert wurde, stellt sich die Frage, wie der Benutzer zur vorigen Ansicht zurückkehren kann. Ein globaler Knopf könnte dafür sorgen, dass alle Knoten in ihren ursprünglichen Zustand zurückversetzt werden, so wie sie vor der Durchführung der Einzelansicht waren. Eine geeignete Position für einen solchen Knopf zu finden, stellte sich als kompliziert heraus. Man könnte argumentieren, dass der Benutzer eine Schaltfläche, um die letzte Aktion rückgängig zu machen, dort erwartet, wo er diese ursprünglich gemacht hat: in dem entsprechenden Knoten. Doch die Gesamtansicht ist dynamisch und kann sich ändern, sodass der Knoten zum Zeitpunkt, wenn der Benutzer die Aktion rückgängig machen möchte, gar nicht mehr sichtbar ist. Lediglich der Einbau eines solchen globalen Knopfs im Wurzelknoten würde das oben beschriebene Problem umgehen, aber dafür neue schaffen: wie könnte man dem Benutzer verständlich machen, dass dieser Knopf die Ansicht zurücksetzt? Der globale Knopf wäre nämlich unter Umständen sehr weit entfernt von der Schaltfläche für die Einzelansicht.

Ich entschloss mich daher für eine andere Alternative. Durch Auswählen der „Parallelansicht“ im Wurzelknoten werden alle unterliegenden Knoten nebeneinander angezeigt. Der Benutzer hat bei diesem Ansatz auch die Möglichkeit, die Parallelansicht auf einen beliebigen Knoten zu aktivieren und somit nur einen ausgewählten Teil des Explorationsbaums parallel anzeigen. Der einzige Punkt, den diese gewählte Lösung nicht abdeckt, ist, dass der Benutzer einen Knopf, um die Aktion rückgängig zu machen, dort erwarten würde, wo er sie ursprünglich ausgelöst hatte.

3.6 Version C: Auch automatisches Zusammenklappen und Auseinanderziehen

Wie wir gesehen haben, ist es ein wichtiger Punkt, dem Benutzer die Möglichkeit zu geben, einzelne Ergebnisfenster manuell zusammenfassen zu können, um sich auf die gewünschten Ausschnitte in der Benutzerschnittstelle besser konzentrieren zu können. Wenn das Browserfenster zu klein wird um den gesamten Inhalt darzustellen, müssen Ergebnisfenster zusammengefasst werden, damit die Anzeige nicht unordentlich beziehungsweise unleserlich wird. In diesen Fällen sollte die notwendige Zusammenfassung automatisch erfolgen, sonst müsste der Benutzer

regelmäßig Arbeit leisten um eine anspruchsvolle Darstellung zu erhalten. Dies gilt hauptsächlich für Mobilgeräte, auf denen diese Erweiterung offensichtlich einen Nutzen hat, da der verfügbare Platz beschränkt ist — aber auch bei der Benutzung auf einem PC kann es für den Benutzer einen Mehrwert darstellen. Insbesondere wird diese Erweiterung umso wichtiger, desto größer der Explorationsbaum wird, da es in diesem Fall besonders wahrscheinlich ist, dass der Baum ohne automatisches Zusammenklappen nicht vollständig auf den Bildschirm passt. Diese Erweiterung ist unumgänglich, wenn man die Verwendung auf kleinen Geräten ermöglichen will und stellt die logische Weiterführung des manuellen Zusammenfassens von Ergebnisfenstern dar.

3.6.1 Wie sieht die gewählte Lösung für den Benutzer im Detail aus?

Das automatische Zusammenfassen von Knoten ist keine Funktionalität der Benutzerschnittstelle, die aktiv gesteuert werden kann, sondern sie wird passiv ausgelöst. Sie reagiert auf die Bedürfnisse der aktuellen Ansicht der Benutzerschnittstelle in Relation zum vorhandenen Bildschirmplatz. Wie auf Abbildung 3.7 zu sehen ist, wird Ansicht (a) bei Verringerung der Breite entsprechend angepasst, sodass in Ansicht (b), so viele Knoten wie möglich gezeigt werden. Dieses passive Verhalten der Benutzerschnittstelle kann durch folgende Aktionen ausgelöst werden:

1. Hinzufügen oder Entfernen eines Knotens

2. Aktivierung der Parallelansicht oder der Einzelansicht

3. Änderung der Fensterbreite

4. Wechseln vom Landschafts- in den Porträtmodus mobiler Geräte

Mithilfe eines Algorithmus, auf den ich noch zu sprechen komme, werden dem Benutzer die gewünschten Inhalte und Mengen an Elementen bestmöglich mit der aktuell verfügbaren Bildschirmgröße angezeigt. Ändert der Benutzer die Breite seines Browserfensters, wird die Anzeige der Knoten neu berechnet und das System entscheidet, welche Knoten ausgeblendet werden oder wiedererscheinen. Ein Spezialfall dieses responsiven Verhaltens bei mobilen Geräten stellt das Wechseln vom Landschafts- in den Porträtmodus und umgekehrt, dar, wo sich die verfügbare Bildschirmbreite ebenfalls ändert.

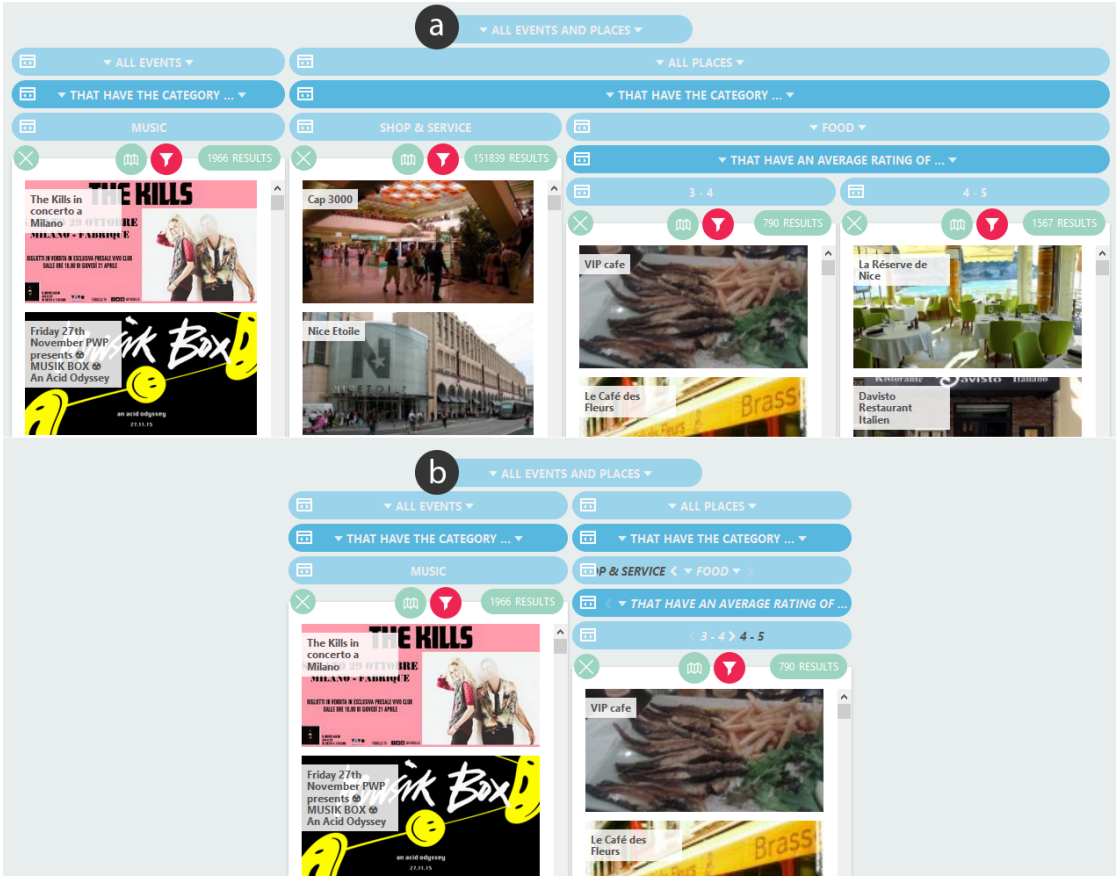

Abbildung 3.7: Automatische Zusammenfassung: Ein Explorationsbaum (a); dessen kompakte Ansicht (b)

3.6.2 Was sind die grundlegenden UI-Designentscheidungen?

Es stellte sich die Frage, welche Knoten beim Zusammenklappen priorisiert werden sollen. Analog zur Strategie des manuellen Zusammenklappens, die im Abschnitt 3.5 beschrieben wurde, wurden im ersten Fall – wie auf Abbildung 3.8 zu sehen ist – die Knoten in der Nähe des Wurzelknotens priorisiert, rechts die terminalen Knoten. Die erste Strategie verwirft dabei vollständig den Suchstrang mit Veranstaltungen, was nützlich sein kann, wenn der Benutzer seinen Fokus nur auf einen bestimmten Ausschnitt des Explorationsbaums setzen möchte. Automatisches Zusammenfassen, welches für den passiven Einsatz gedacht ist, sollte jedoch nicht solche starken Änderungen der Ansicht hervorrufen, sondern feingranularer agieren. Wie Abbildung 3.9 veranschaulicht, erfüllt die zweite Strategie genau diesen Zweck.

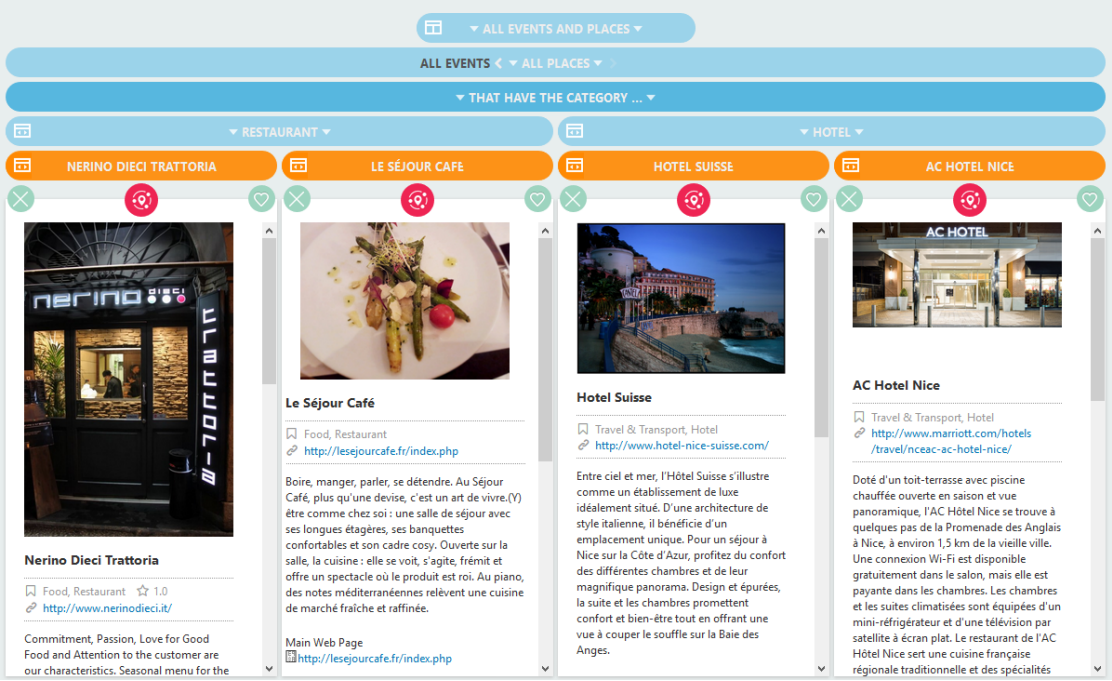

Abbildung 3.8: Erste Strategie für das Zusammenfassen von Knoten: Priorisierung der Knoten in der Nähe des Wurzelknotens

Beim Hinzufügen eines neuen Knotens werden nur terminale Knoten, anstatt die grundlegende Ansicht zu verändern, zusammengeführt. Hierbei ist anzumerken, dass auch höhere Ebenen im Explorationsbaum bei der Zusammenfassung in Betracht gezogen werden, falls durch die Zusammenfassung terminaler Knoten nicht genügend Platz geschaffen werden kann.

3.6.3 Welche Möglichkeiten gibt es, diese Entscheidungen technisch umzusetzen; und was sind deren Vor- und Nachteile?

Bei der Wahl eines geeigneten Algorithmus, um aus einer Menge von zusammenzufassenden Knoten eine Untermenge auszuwählen, war ein gieriger Algorithmus die erste Option. Dieser wählt zunächst den Knoten aus, der am wenigsten Platz einspart und macht so lange weiter, bis die vorhandene Breite der benötigten Breite entspricht. Dass dies nicht immer zur optimalen Lösung führt, möchte ich anhand eines Minimalbeispiels näher erörtern: der aufgespannte Explorationsbaum aus

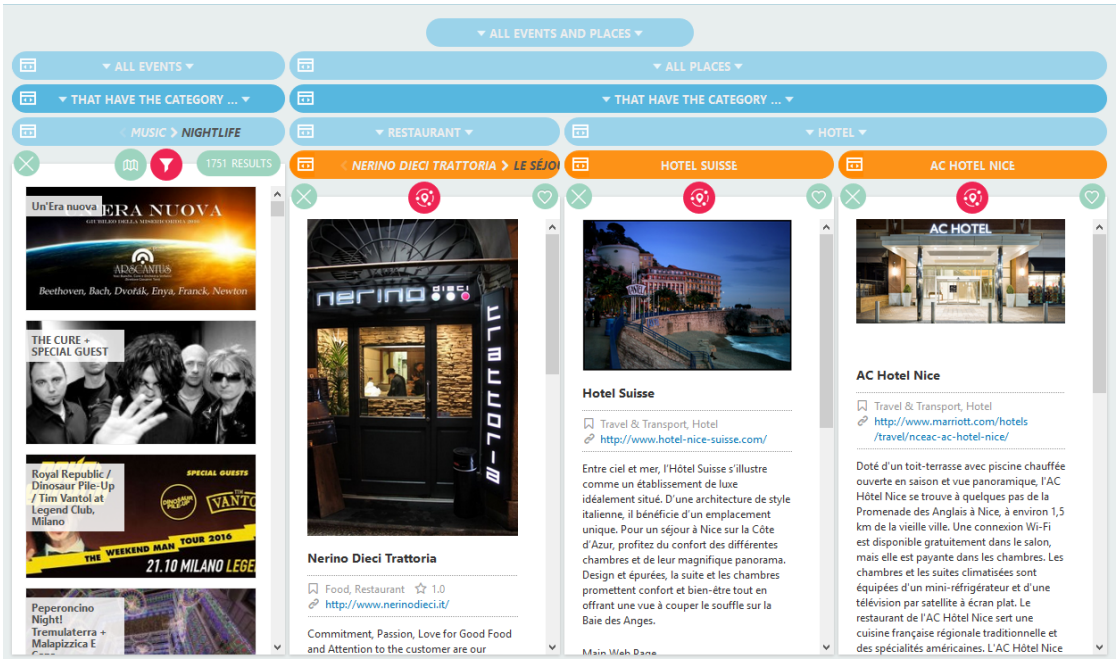

Abbildung 3.9: Zweite Strategie für das Zusammenfassen von Knoten: Priorisierung der unteren Knoten

Abbildung 3.10 (a) benötigt sieben Spalten um vollständig dargestellt zu werden. Die berechnete, momentan verfügbare Fensterbreite lässt jedoch nur die Darstellung von vier nebeneinanderliegenden Knoten zu, daher müssen im Optimalfall drei Spalten eingespart werden. Ein gieriger Algorithmus würde die Knoten aufsteigend, nach der Anzahl ihrer Kinder, solange zusammenfassen, bis der Explorationsbaum höchstens vier nebeneinanderliegende Knoten enthält. Das führt zu einer Darstellung wie in Abbildung 3.10 (b), in der beide zur Auswahl stehenden Knoten zusammengefasst wurden, da nach dem ersten Iterationsschritt die Bedingung nicht erfüllt wurde. Mehr als drei Spalten zu verwerfen wäre jedoch schade, weil Inhalte, die theoretisch Platz auf dem Bildschirm hätten, nicht angezeigt werden. Ein optimaler Algorithmus würde die in (c) dargestellte Lösung liefern, hierfür müssen aber – insbesondere in Hinblick auf komplexere Beispiele – alle Kombinationen der Mengen an Knoten in Betracht gezogen werden, die zusammenfassbar sind. Um die Zusammenfassung unnötiger Knoten zu vermeiden, entschloss ich mich für den Algorithmus, der die optimale Lösung liefert und dieses Problem mithilfe dynamischer Programmierung zu lösen.

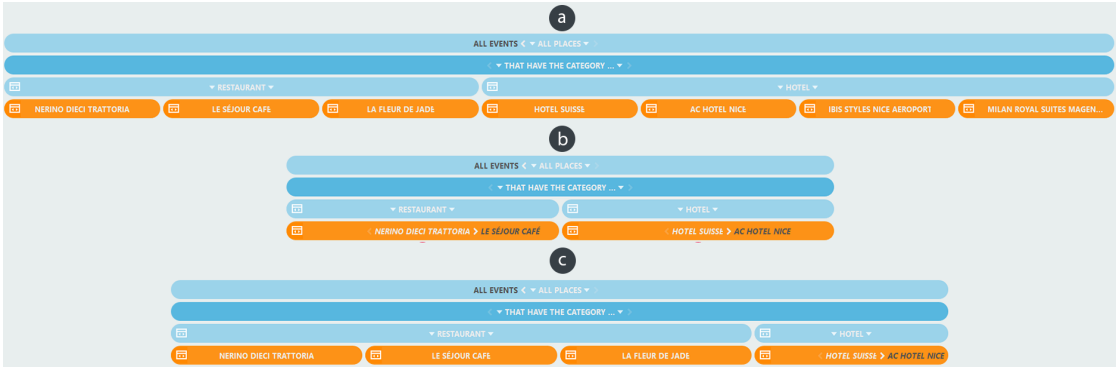

Abbildung 3.10: Algorithmen für die Zusammenfassung von Knoten: die Kopfzeilen des initialen Explorationsbaums (a); Zusammenfassung mithilfe eines gierigen Algorithmus (b); optimale Lösung für die Zusammenfassung (c)

3.6.4 Wie wurde das neue Design implementiert?

Wie in der Abbildung 3.11 zu sehen ist, hat jeder Knoten zu jedem Zeitpunkt einen von drei möglichen Zuständen, die ich mit drei Farben kodiert habe.

1. Rote Knoten beschreiben Knoten in der Einzelansicht
2. Gelbe Knoten beschreiben Knoten in der Parallelansicht
3. Pinke Knoten beschreiben Knoten in der automatisch zusammengeklappten Ansicht

Um die bisher beschriebenen Interaktionsmöglichkeiten, insbesondere das Zusammenfassen mehrerer Knoten der PE-Benutzerschnittstelle zu implementieren, wurden folgende Regeln herausgearbeitet, die dies bewerkstelligen:

Ein Knoten ist genau dann *fokussiert*, wenn der Benutzer die „Parallelansicht“ oder die „Einzelansicht“, die ihm manuelles Zusammenfassen ermöglichen, darauf aktiviert. Zu einem gegebenen Zeitpunkt ist stets genau ein Knoten fokussiert, und zwar der, auf dem zuletzt entweder Parallel- oder Einzelansicht aktiviert wurde. Im Initialzustand ist der Wurzelknoten fokussiert.

Ein Knoten gehört dann zur *roten* Kategorie, wenn er fokussiert ist. Zusätzlich sind alle Knoten zwischen dem Wurzelknoten (a) und dem aktuell fokussierten Knoten (b) werden ebenfalls rot. Wird der Fokus auf einen anderen Knoten umgesetzt, werden zunächst alle roten Knoten gelb, und die Knoten zwischen dem Wurzelknoten

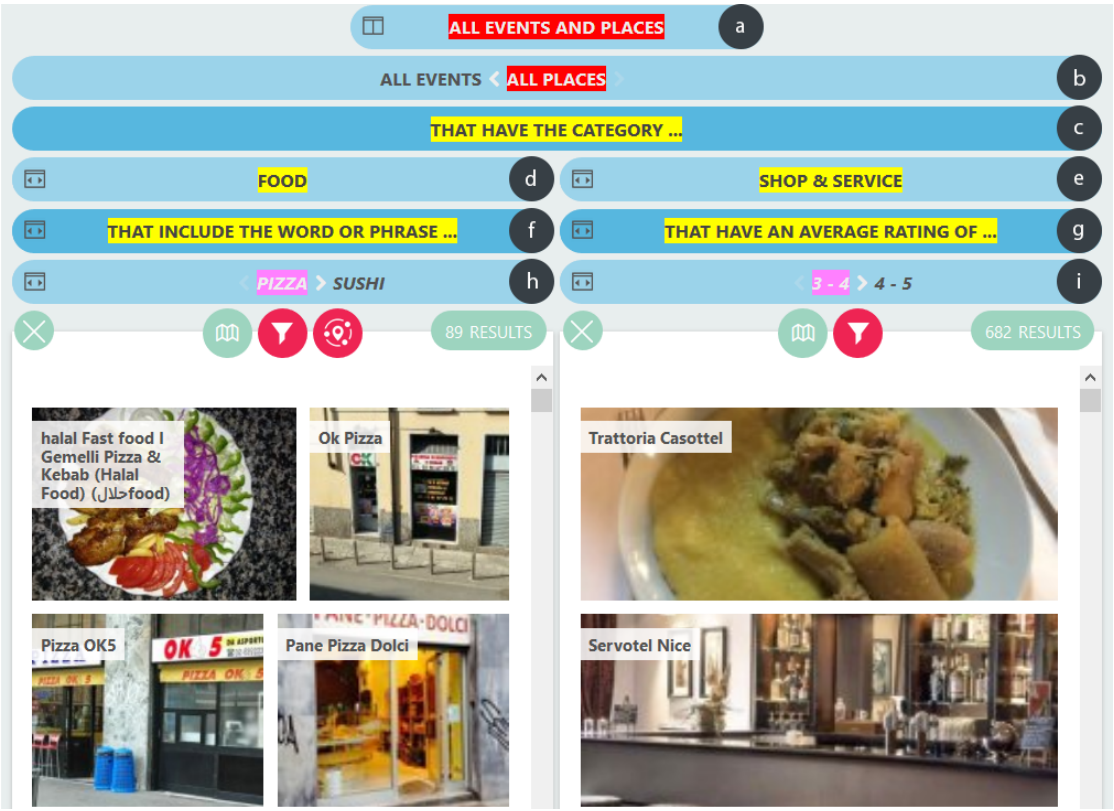

Abbildung 3.11: Färbung unterschiedlicher Knotenansichten

und dem neuen Fokus anschließend rot gefärbt. Rote Knoten verwenden stets die maximale Bildschirmbreite (a) (b), daher kann es niemals vorkommen, dass zwei Rote nebeneinanderstehen. Ein roter Knoten kann mehrere Knoten zusammenfassen (b), zwischen denen man hin- und herschalten kann.

Gelbe Knoten sind alle Knoten unterhalb dem gesetzten Fokusknoten (c) - (g), insofern sie nicht – wie im nächsten Abschnitt erklärt – pink gefärbt wurden. Gelbe Geschwisterknoten werden niemals zusammengefasst, sondern parallel nebeneinander angezeigt (d) (e).

Pinke Knoten entsprechen den automatisch zusammengefassten Knoten, die im Abschnitt 3.6 erläutert wurden. Sie entstehen immer dann, wenn mehrere gelbe Geschwister mehr als die aktuell verfügbare Bildschirmbreite ausnutzen. Diese fassen mehrere gelbe Knoten zusammen (h) (i), zwischen denen man wechseln kann. Durch diese Zusammenfassung kann es nicht passieren, dass pink gefärbte Geschwisterknoten nebeneinander angezeigt werden — man beachte, dass (h) und

Abbildung 3.12: Nummerierung der inversen Knotentiefe anhand eines Beispiels

(i) keine direkten Geschwisterknoten sind.

Bei der Erstellung eines Knotens hängt dessen initiale Färbung vom Elternknoten ab, die geerbt wird. Die einzige Operation, mit der der Benutzer die Färbung sonst noch steuern kann, ist das Setzen eines „Fokus“ auf einen bestimmten Knoten. Dabei wird der Fokusknoten, und alle Knoten oberhalb von ihm rot gefärbt — alle unterhalb gelb. Immer dann, wenn gelbe Knoten aus der Bildschirmbreite herausragen würden, werden diese pink gefärbt und zusammengefasst. Das sorgt dafür, dass der Benutzer niemals horizontal scrollen muss.

Ich werde nun auf die Implementierung eingehen, um die zusammenzufassenden Knoten zu bestimmen. Der Hauptalgorithmus wird unter 3 beschrieben, dieser greift auf zwei Hilfsalgorithmen zurück: (a) die Berechnung der umgekehrten Knotentiefe (Algorithmus 1) und die Bestimmung der maximalen Breite des Baumes (Algorithmus 2).

Bestimmung der inversen Knotentiefe Dieser Algorithmus ist notwendig, um die Reihenfolge der Knoten zu erfassen, in der diese automatisch zusammengefasst werden. Als Resultat dieses Algorithmus erhalten terminale Knoten einen Wert von 0 und der Wurzelknoten stets den höchsten Wert. Es wird jeweils die höchste umgekehrte Tiefe eines Knotens berechnet, da ein Knoten mehrere, unterschiedliche Pfade zu terminalen Knoten haben kann. Auf Abbildung 3.12 ist eine solche Beispielkonfiguration eines Baumes mit inverser Höhe erkennbar, wobei die Zahlen die inverse Tiefe widerspiegeln.

1. Das Attribut „inverseDepth“ aller Knoten im Baum „T“ wird auf -1 gesetzt (Zeile 1-3)

Algorithmus 1 : Bestimmung der inversen Knotentiefe

Data : T representing the tree, a directed graph containing unlabled nodes
Result : Each node in T is labled with the attrbute inverseDepth

```
1  foreach node ∈ T do
2  |  node.inverseDepth ← −1
3  end
4  S ⟵ ∅;
5  foreach node ∈ T do
6  |  if node.children = ∅ then
7  |  |  S.add(node);
8  |  end
9  end
10 while S ≠ ∅ do
11 |  foreach node ∈ S do
12 |  |  node.inverseDepth++;
13 |  |  if node.parent then
14 |  |  |  S.add(node.parent);
15 |  |  end
16 |  |  S.remove(node);
17 |  end
18 end
```

2. Die Menge „S“ wird mit terminalen Knoten gefüllt. Das entspricht der Menge von Knoten ohne Kinder (Zeile 4-9)

3. Solange die Menge „S“ nicht leer ist, wird das Attribut „inverseDepth“ für jeden Knoten erhöht und der Knoten durch seinen Elternknoten in S ersetzt (Zeile 10-18). Da es sich bei „S“ um keine Liste, sondern eine Menge die keine Duplikate zulässt, handelt, werden Elternknoten mit mehreren Kindern nicht mehrfach hinzugefügt. Die Anzahl der Elemente in „S“ verringert sich in jedem Iterationsschritt, da jeder Knoten nur höchstens einen Elternknoten besitzt.

4. Da der Wurzelknoten keinen Elternknoten hat, wird in diesem Fall kein weiterer Knoten hinzugefügt (Zeile 13).

Die maximale Breite des Baumes erfolgt durch die Bestimmung der Anzahl terminaler Knoten, da diese nebeneinander am unteren Ende des Baums stehen und dadurch den größten horizontalen Platzverbrauch haben.

Die Breite wird wie folgt bestimmt: Zunächst wird die Breite auf 0 gesetzt. Anschließend werden alle Knoten des Baumes durchiteriert und die Variable jedes Mal inkrementiert, wenn ein Knoten gefunden wird, der sichtbar ist und keine Kinder hat. Algorithmus 2 zeigt eine Implementierung dessen in Pseudocode.

Algorithmus 2 : Bestimmung der maximalen Breite des Baumes

Data : T representing the tree, a directed graph
Result : The maximum width of T

```
1 treeWidth ⟵ 0;
2 foreach node ∈ T do
3 |   if node.children = ∅ then
4 |   |   if node.visible then
5 |   |   |   treeWidth++;
6 |   |   end
7 |   end
8 end
9 return treeWidth;
```

Bestimmung der zusammenzufassenden Knoten Der Grundgedanke dieses Algorithmus ist es, anhand der gegebenen Fenstergröße und des Platzverbrauchs des Explorationsbaums, diejenigen Knoten zu bestimmen, die automatisch zusammengefasst werden müssen, um horizontales Scrollen zu vermeiden.

Zunächst erhalten alle Knoten als Ausgangswert die Parallelansicht (Zeile 1-3), die den größten Bedarf an Bildschirmplatz in Anspruch nimmt. Ausgehend davon, wird im Verlauf des Algorithmus durch die beschriebenen Operationen Platz eingespart, um dafür zu sorgen, dass bei einer Änderung des aktuellen Platzverbrauchs oder der Bildschirmbreite die optimale Lösung gefunden wird.

Solange die Breite des Baumes größer als die verfügbare Bildschirmbreite ist, wird eine Schleife durchlaufen. Dabei liefert der Algorithmus 2 die maximale Breite des Baumes in Spalten, die durch den Quotienten der Bildschirmbreite und der minimalen Breite eines Knotens bestimmt wird (Zeile 4).

In dieser Schleife werden in jedem Schritt jene Knoten mit der niedrigsten inversen Tiefe, die mindestens zwei Kinder haben, ermittelt und der Menge „collapsableSet“ hinzugefügt (Zeile 8-14). Dabei werden erst die Knoten mit einer inversen Tiefe von 1 – und nicht 0 – betrachtet, da terminale Knoten ohnehin keine Kinder zum Zusammenfassen haben (Zeile 7).

Mithilfe der Potenzmengenbildung werden alle möglichen Kombinationen von „collapseableSet“ berechnet (Zeile 15). Anschließend werden jene ausgewählt, die mit minimaler Einsparung einen gültigen Zustand – bei der die verfügbare Breite größer oder gleich der Verbrauchten ist – erreicht und in „winningCombination“ zwischengespeichert (Zeile 28). Dabei müssen alle Kombinationen von Teilmengen berechnet werden und jene ausgesucht, die am wenigsten Platz einsparen, sodass die Bedingung noch erfüllt wird. Wenn durch Zusammenfassen aller betroffenen Knoten das Ziel nicht erreicht wurde, werden alle Knoten der aktuellen inversen Tiefe zusammengefasst und automatisch die nächste Ebene der inversen Tiefe untersucht (Zeile 31-35). Sonst wird nur die gewinnende Kombination von Knoten zusammengefasst (Zeile 36-40) und der Algorithmus terminiert, da die Bedingung der While-Schleife nicht mehr erfüllt ist (Zeile 6).

Algorithmus 3 : Bestimmung des automatischen Zusammenklappens

```
foreach node ∈ T do
    node.setParallelView();
end
availableWidth ⟵ windowWidth/nodeWidth;
currentInverseDepth ⟵ 1;
while treeWidth(T) > availableWidth do
    collapsableSet ⟵ ∅;
    foreach node ∈ Tree do
        if node.inverseDepth = currentInverseDepth then
            if node.children.size() > 1 then
                collapsableSet.add(node);
            end
        end
    end
    allCombinations ⟵ powerset(collapsableSet);
    winningCombination ⟵ collapsableSet;
    winningValue ⟵ false;
    foreach combination ∈ allCombinations do
        saving ⟵ 0;
        foreach node ∈ combination do
            if node.inverseDepth = currentInverseDepth and
             node.children.size > 1 then
                saving ⟵ saving + subtreeWidth − combination.size();
            end
        end
        currentValue ⟵ treeWidth(T) − saving − availableCols;
        if currentValue ≤ 0 and currentValue > winningValue then
            winningValue ⟵ currentValue;
            winningCombination ⟵ combination;
        end
    end
    if not(winningValue) then
        foreach node ∈ collapsible do
            node.setSingleView();
        end
    end
    else
        foreach node ∈ winningCombination do
            node.setSingleView();
        end
    end
    currentInverseDepth + +;
end
```

4 Neue Funktionalität für die Parallele Exploration

Dieses Kapitel beschäftigt sich mit der zweiten Forschungsfrage, die in Abschnitt 1.4 vorgestellt wurde: „Welche neue, allgemeine Funktionalität kann der PE hinzugefügt werden?“. Ich werde erläutern wie die verwandten Arbeiten und die im Abschnitt 3.1 erhaltenen Rückmeldungen, die wir zu PE erhielten, in Erweiterungen umgesetzt wurden. Dabei verwende ich den gleichen Aufbau wie im vorigen Kapitel, um die jeweiligen Erweiterungen zu beschreiben.

Zusammen mit Kai-Dominik Kuhn [19] wurden die in dieser Arbeit vorgestellten Erweiterungen realisiert. Dabei habe ich das Frontend – die UI-Elemente – realisiert, während er die Anfrageebene entworfen hat. An den entsprechenden Schnittstellen, an denen unsere Komponenten zusammenarbeiten, verweise ich auf die relevanten Kapitel seiner Ausarbeitung.

Das Grundkonzept wurde von mir in Zusammenarbeit mit Prof. Dr. Anthony Jameson ausgearbeitet. Anschließend wurden Ideen bezüglich der grafischen Umsetzung, wie beispielsweise die Farbgebung sowie die Gestaltung der Formen für die Benutzerschnittstelle von Data-Moove bereitgestellt, die das Erscheinungsbild der Benutzerschnittstelle verbessern konnten. Danach habe ich den Entwurf und die Implementierung unter Berücksichtigung der in dieser Arbeit besprochenen Entscheidungen realisiert. An den Stellen, die ich explizit erwähnen werde, wurden die Erweiterungen vom 3CIXTY-Unterauftragnehmer Deip Designs angepasst, wie beispielsweise die Verwendung einer Datenbank anstatt von Konfigurationsdateien um die Skalierbarkeit und Effizienz einzelner Komponenten zu steigern.

4.1 Globale Suche

Ein Suchfeld für einen schnellen Einstieg ist intuitiv und weit verbreitet in Suchmaschinen, Content Management Systemen und diversen Suchsystemen, daher sind Benutzer solche Volltextsuchen gewohnt. Es gibt ihnen nach Eingabe eines

Stichwortes einen schnellen und oft nützlichen Einstieg, von dem aus sie weiter erkunden können.

4.1.1 Welche Ergebnisse zur Bedienbarkeit und zur Nützlichkeit lagen dem Design zugrunde?

Diese Erweiterung wurde direkt durch Teilnehmer der Benutzerstudie, die im Rahmen von Edit Kapcaris Masterarbeit [18] (Abschnitt 5.4, Seite 54) durchgeführt wurde, vorgeschlagen. Sie stellte fest, dass die Teilnehmer eine globale Stichwortsuche in der PFB-Benutzerschnittstelle vermissten, um eine schnelle Stichwortsuche, ohne einzelne Knoten und Filter öffnen zu müssen, durchzuführen. Dadurch sollte die Lernzeit für das System insgesamt verkürzt werden, da die Verwendung einer Stichwortsuche für jeden Nutzer intuitiv erscheint. Außerdem ist es förderlich, von Nutzern bekannte Elemente der PFB-Benutzerschnittstelle hinzuzufügen, damit diese vertrauter aussieht.

4.1.2 Wie sieht die gewählte Lösung für den Benutzer im Detail aus?

Wie in dem linken Teilausschnitt von Abbildung 4.1 zu sehen ist, kann der Benutzer in einem Suchfeld in der oberen Leiste Eingaben tätigen und dadurch automatisch einen Explorationsbaum erstellen lassen, der seine Textsuche beantwortet. Dabei werden zwei Stränge im Explorationsbaum aufgebaut: es wird nach Veranstaltungen und Orten parallel gesucht und getrennt voneinander angezeigt, wie auf dem linken Teilausschnitt ersichtlich ist. Dies verdeutlicht neuen Benutzern die Möglichkeit, mehrere Ergebnismengen bilden und diese auch getrennt voneinander betrachten zu können. Die globale Volltextsuche lässt sich sowohl im initialen Zustand, als auch zu jedem späteren Zeitpunkt der Exploration durchführen; im letzteren Fall wird der Explorationsbaum um die fehlenden Knoten ergänzt und die neuen Suchergebnisse erweitern die bisherigen Ergebnisse der Erkundung.

4.1.3 Was sind die grundlegenden UI-Designentscheidungen?

Eine grundlegende UI-Designentscheidung beim Entwurf der globalen Suche war es, ob nach einer Suche unter den Suchergebnissen das Kategoriemenü, das dem Filtrieren nach Kategorie entspricht, angezeigt werden sollte. Auf der einen Seite könnten sich möglicherweise zu viele Knoten auf einmal öffnen, auf der anderen

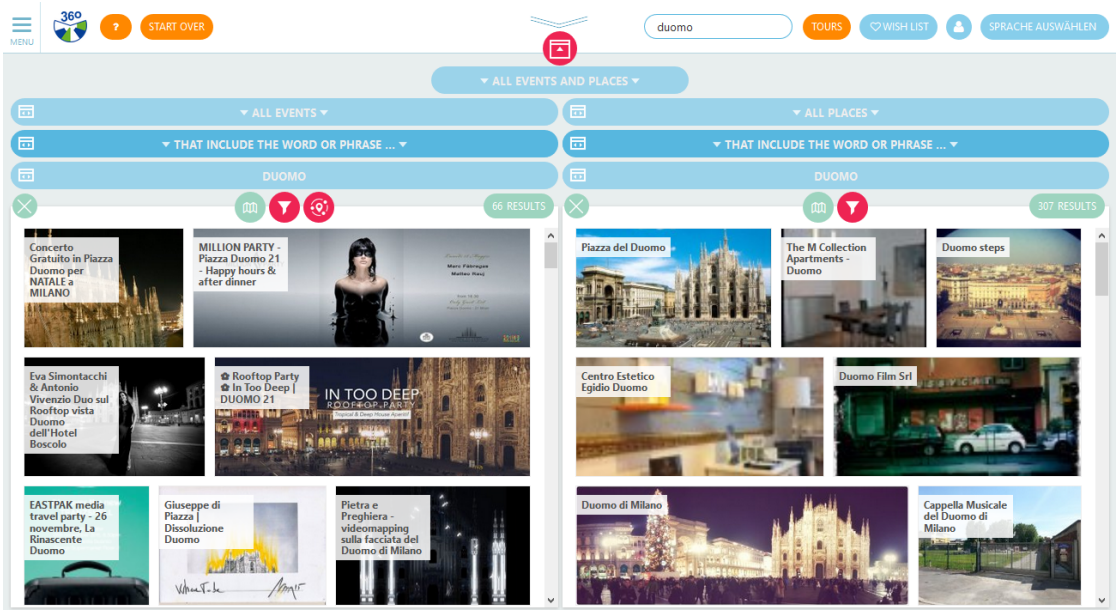

Abbildung 4.1: Realisierung der Volltextsuche, die auf das Paradigma der Parallelen Exploration angewandt wurde

ermutigt es den Benutzer auf einfache Weise, seine Präferenzen genauer zu spezifizieren und animiert ihn weiter zu erkunden. Gleichzeitig hilft es ihm die Ergebnisse besser einschätzen zu können, da es ihm sofortiges Feedback über die Kategoriezugehörigkeit der Ergebnisse liefert, ohne viel manuell spezifizieren zu müssen und nimmt ihm dadurch Arbeit ab. Ein Nachteil wäre, dass der Benutzer in der aktuellen Benutzerschnittstelle nur das Kategoriemenü sehen würde und nicht die Ergebnisse selbst, da man nur die Kopfzeile bei nichtterminalen Knoten sehen kann (siehe Abbildung 4.2). Diese Art der Ergebnisdarstellung könnte für neue Benutzer verwirrend wirken — genau den Benutzern, für die diese Erweiterung ursprünglich implementiert wurde. Um dies zu vermeiden, habe ich mich entschlossen, das Kategoriemenü nicht automatisch einzublenden.

4.1.4 Welche Rückmeldung bekamen wir?

Aus der Expertenmeinung von ErgoSign stellte sich heraus (Abschnitt 3.1), dass es für den Benutzer keinen einfachen Weg gibt eine Suche zurückzusetzen, nachdem dieser eine Suche durchgeführt hat und eine weitere machen möchte. Bei dem ursprünglichen Ansatz hatte ich versucht, die Suchergebnisse in das Paradigma von

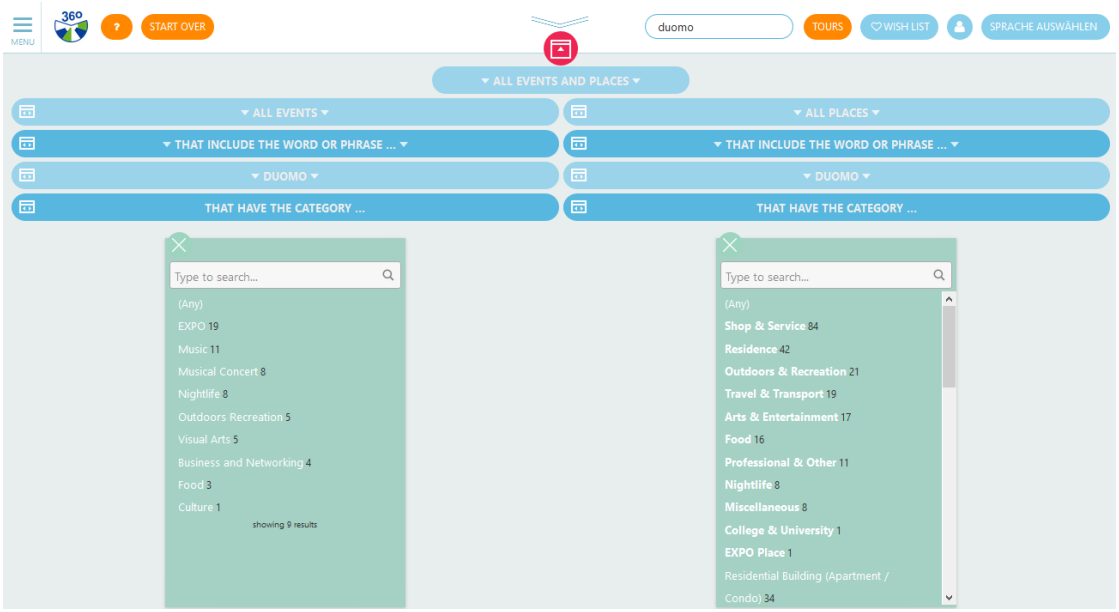

Abbildung 4.2: Anschließendes Öffnen des Kategoriemenüs nach Anwendung der globalen Suche

PE einzubauen, sodass der Benutzer Ergebnisse mehrerer Volltextsuchen angezeigt bekommt. Zwei Möglichkeiten stehen zur Verfügung, dieses Problem zu lösen: entweder ersetzt eine neue Suche die vorige oder es wird dem Benutzer ein „Start Over“-Knopf angeboten, der den ursprünglichen Explorationsbaum mit den beiden Knoten „Veranstaltungen“ und „Orte“ wiederherstellt. Da durch die erste Variante bestimmte Vergleiche nicht mehr möglich wären, habe ich den Knopf „Start Over“ implementiert, der rechts oben in Abbildung 4.2 zu sehen ist.

4.1.5 Wie wurde es implementiert?

Aus technischer Sicht entspricht die globale Volltextsuche für Veranstaltungen dem automatischen Erstellen eines Kindsknoten unter dem Wurzelknoten für das Anzeigen der Veranstaltungen. Dabei wird zunächst geprüft, ob dieser Knoten bereits vorhanden ist oder neu erstellt werden muss. Linear darunter wird ein Knoten hinzugefügt, der die Volltextsuche ausführt. Der gleiche Vorgang wird analog dazu mit den Orten durchgeführt.

[1] Auf multiples Pivoting, bei dem man von einer Menge von Elementen ausgehend nach verwandten Dingen suchen kann, wird in Abschnitt 4.4 näher eingegangen.

4.2 Einfaches Pivoting

Die Motivation dieser Erweiterung ist es, dem Benutzer die Möglichkeit zu geben, vom einem Element ausgehend nach verwandten Dinge suchen zu können.[1] Die Grundidee von Pivoting wurde bereits in den verwandten Arbeiten (Abschnitt 2.4.1) anhand von Parallax vorgestellt.

4.2.1 Warum ist diese Erweiterung wichtig?

Pivoting erlaubt es dem Benutzer, komplexe Anfragen zu formulieren und den meisten Nutzen aus einer heterogenen Wissensdatenbank herauszuholen. Beispielsweise kann er ausgehend von einer Veranstaltung zu Restaurants navigieren, die in der Nähe des Veranstaltungsortes sind. Diese Art zu Navigieren unterscheidet sich vom normalen Filtrieren: die Zielmenge muss weder den gleichen Entitätstypen haben wie das Quellelement, noch ist sie in der Regel eine Teilmenge davon. Diese Art von Anfragen können üblicherweise von Benutzern nur mit Anfragesprachen realisiert werden oder sind in bestehenden Systemen bereits fest vordefiniert – beispielsweise das Anzeigen von Rezensionen zu einem Produkt – und dadurch nur eingeschränkt nutzbar.

Unsere Implementierung von Pivoting soll generisch sein, sodass ein Benutzer, je nachdem, welche spezifische Relation für Pivoting verwendet wird, von einer Entität zu einer anderen Entität navigieren kann. Obwohl in der aktuellen Version der Benutzerschnittstelle nur gewisse Entitätswechsel durch Pivoting durchführbar sind, ist das allgemeine Prinzip sehr weitläufig und eröffnet viele Möglichkeiten der Exploration. Pivoting kann verwendet werden, um ähnliche Elemente zu einem Gegebenen zu finden oder entlang beliebiger Beziehungen zu verknüpfen, beispielsweise kann der Benutzer von Veranstaltungen direkt zu deren Veranstaltungsorten wechseln.

4.2.2 Welche Ergebnisse zur Bedienbarkeit und zur Nützlichkeit lagen dem Design zugrunde?

Bereits aus der Benutzerstudie in meiner Bachelorarbeit [26] (Kapitel 5.4, Seite 41) ging hervor, dass Pivoting von einigen Nutzern als Erweiterungsmöglichkeiten genannt wurde.[2] In diesem frühen Prototyp von PFB konnten Benutzer lediglich

[2] Es ist anzumerken, dass die Benutzer selbst auf die Idee gekommen sind; es wurde nicht ausdrücklich danach gefragt.

Elemente einer Entität erkunden, nämlich Lebensmittel. Es wäre demnach hilfreich, diese Lebensmittel mit anderen Entitäten, beispielsweise Rezepten, in Verbindung zu bringen und diese ebenfalls zu erkunden. Spätere Rückmeldungen bei öffentlichen Vorführungen von PE (Abschnitt 3.1) bestätigten dieses Anliegen der Benutzer und es wurde mitgeteilt, dass diese Funktion für die Benutzerschnittstelle einen Mehrwert darstellt.

4.2.3 Was sind die grundlegenden UI-Designentscheidungen?

Um Redundanzen zu vermeiden werde ich bei der Beantwortung dieser Frage gleichzeitig auch auf die Frage eingehen, wie die Erweiterung für den Benutzer aussieht. Es gibt zwei unterschiedliche Möglichkeiten der Interaktion, wie der Benutzer Pivotingaktionen in der Benutzerschnittstelle durchführen könnte. Er kann zwei voneinander getrennte Knoten erzeugen, beispielsweise ein Hotel und alle Parkanlagen. Im nächsten Schritt kann er diese Knoten miteinander verbinden und eine Beziehung angeben, wie zum Beispiel „innerhalb von 500 Meter“. Bei diesem Ansatz müsste ein geeigneter Platz auf dem Bildschirm gefunden werden, wo die resultierende Ergebnismenge anschließend angezeigt wird — beispielsweise unter dem Quellknoten mit nur einem Element.

Um die grundlegenden Interaktionstechniken von PE möglichst konsistent zu gestalten, habe ich mich für folgenden, alternativen Ansatz entschieden, bei dem die Interaktion analog zu der beim Filtrieren ist; lediglich die Abbildung auf eine neue Menge ist anders. Diese Art der Interaktion führt nur minimal neue Bedienkonzepte ein, an die sich der Benutzer gewöhnen muss, sondern greift auf bestehendes Wissen zurück.

Abbildung 4.3 zeigt, wie ein Benutzer Pivoting in der PE-Benutzerschnittstelle anwenden könnte. Ausgehend von einem Element – in diesem Fall der Mailänder Dom – (a) kann er über das Pivotingmenü auswählen, was er in der Nähe erkunden möchte (b). Dabei stehen ihm mehrere Möglichkeiten zur Verfügung: Elemente in einem bestimmten Umkreis mit einer angegebenen Anzahl von Metern Luftlinie (c); oder er gibt die Minuten an, die er bereit ist mit der U-Bahn zu fahren. Anschließend erhält er eine Ergebnismenge, die seinen Kriterien entspricht. Auf (d) werden alle Restaurants angeboten, die höchstens 500 Meter Luftlinie von dem Mailänder Dom entfernt sind. Obwohl in dem Menü (b) nur Hotels und Restaurants für eine Schnellauswahl zur Verfügung stehen, ist das System in der Lage, Pivotingoperationen zu beliebigen Entitätstypen durchzuführen. Der Benutzer kann angeben, nach nahegelegenen Orten zu suchen (e) und danach nach einer beliebigen Kategorie filtrieren. Bildschirmabzug (f) zeigt die anschließende Filtrierung nach Parkanlagen.

Abbildung 4.3: Pivoting mithilfe der PE-Benutzerschnittstelle

4.2.4 Welche Möglichkeiten gibt es, diese Entscheidungen technisch umzusetzen; und was sind deren Vor- und Nachteile?

Da die Implementierung der beschriebenen Menüs aus technischer Sicht unkompliziert ist, betrifft die Hauptimplementierung von Pivoting die Anfragen, die durch die Benutzerschnittstelle an die Wissensdatenbank geschickt werden. Diese

wurden ausführlich in der Masterarbeit von Kai-Dominik Kuhn [19] (Kapitel 4.6) behandelt. Die Anfrageebene erhält von der Benutzerschnittstelle die erforderlichen Daten für das Pivoting in Form eines JavaScript-Objekts und formuliert diese in eine gültige SPARQL-Anfrage, die von der Wissensbasis beantwortet wird. Die daraus resultierenden Daten werden von der Benutzerschnittstelle angezeigt. Dieses Prozedere ist analog zu den Filteranfragen, die durch die UI gehandhabt werden.

4.2.5 Wie wurde es implementiert?

Es lassen sich neben den normalen, bekannten Filteroperationen innerhalb der Benutzerschnittstelle auf leichte Weise zusätzliche Interaktionsmöglichkeiten einbauen, wie beispielsweise das Pivoting. Mithilfe von Vererbung konnten weitere Knotentypen erstellt werden, beispielsweise „PivotingNodes“, die grundlegende Eigenschaften von normalen Knoten erben, aber auch neue Attribute in der Anzeige einblenden, wie beispielsweise das Anzeigen des Abstands eines Elements zum Quellelement nach der Durchführung einer Pivotingaktion. Hierbei werden folgende Attribute erhoben und in Form eines Arrays an die Anfrageebene weitergegeben: die Identität des Quellelements, die Entität der Zielmenge und die einschränkenden Attribute der Pivotingaktion, beispielsweise „500 Meter“.

4.3 Manuelle Selektion

Ich gehe in diesem Abschnitt auf die Erweiterung „Manuelle Selektion“ ein, die es dem Benutzer ermöglicht, jede beliebige Untermenge von Elementen eines Knotens zu erstellen.

4.3.1 Warum ist diese Erweiterung wichtig?

Das Bilden von beliebigen Untermengen kann für den Benutzer in vielen Situationen hilfreich sein. Durch die manuelle Selektion kann er spezifische Elemente auswählen und sich diese auf einer Karte anzeigen lassen um ein räumliches Verständnis für die Beziehung der Elemente zu entwickeln. Auch bei der Planung eines Aufenthalts kann er mithilfe dieser Erweiterung eine Tour zusammenstellen, indem er die für ihn interessanten Orte in einem Knoten zusammenfasst.

Die manuelle Selektion findet auch als eine Erweiterung des bereits im Abschnitt 2.1 vorgestellten Prinzips von „Winnowing“ Anwendung: sie stellt eine weitere Ebene dar, die dem Benutzer ermöglicht, eine – möglicherweise unüberschaubar – große

Menge von Elementen zu reduzieren und die neu erstellte Menge zu betrachten. Somit ist der Benutzer nicht zum Anwenden eines Filters gezwungen, wenn er die Anzahl der Elemente in der Quellmenge reduzieren möchte, sondern kann auf eine feingranulare Selektion zurückgreifen, die ihm entscheidungsunterstützend bei der Anwendung der Winnowing-Strategie hilft.

Wie wir in dem vorigen Abschnitt gesehen haben, ist das einfache Pivoting für den Benutzer nützlich. Möchte man Pivoting jedoch auf eine Menge anwenden – wie in der verwandten Arbeit mit Parallax 2.4.1 bereits gezeigt wurde – ist es sinnvoll, beliebige Mengen von Elementen bilden zu können. Daher diente als weitere Motivation für die manuelle Selektion das multiple Pivoting, auf das ich in Abschnitt 4.4 näher eingehen werde: der Benutzer muss in der Lage sein, aus einer Liste von Elementen diejenigen auszuwählen, von denen aus er weiter erkunden möchte (beispielsweise das Auswälen von fünf ansprechenden Hotels aus einer Liste mit 30 Hotels). Er sollte nicht gezwungen sein, multiples Pivoting auf einer sehr großen Menge anzuwenden, wenn er nur an einigen Elementen als Quelle für die weitere Erkundung interessiert ist.

4.3.2 Wie sieht die gewählte Lösung für den Benutzer im Detail aus?

Abbildung 4.4 veranschaulicht, wie ein Benutzer die Manuelle Selektion anwenden kann: er wählt zunächst über das Filtermenü diese Funktion aus (a). Anschließend kann er beliebige Elemente für die neue Ergebnismenge selektieren (b). Bevor er sich entscheidet, ob er das jeweilige Element zur Selektion hinzufügen möchte, kann er nähere Informationen zu einem Element in einer Detailansicht erhalten (c). Danach kann er mit den ausgewählten Objekten eine Untermenge erstellen und von dort aus weiter erkunden (d).

4.3.3 Was sind die grundlegenden UI-Designentscheidungen?

Bei der Platzierung des Auslösers für „Manuelle Selektion“ in der Benutzerschnittstelle standen zwei Möglichkeiten zur Verfügung. Man könnte dies einerseits über einen zusätzlichen Umschaltknopf in der Kopfzeile des Knotens realisieren, der bei Aktivierung die manuelle Selektion ermöglicht — im deaktivierten Zustand hätte man die Standardansicht der Knoten. Dies hätte den Nachteil, dass eine weitere sichtbare Schaltfläche für jeden Knoten erforderlich wäre, obwohl es keine so häufig verwendete Funktion wie Filtrieren oder Pivoting ist. Der alternative Ansatz wäre, manuelle Selektion über eine weitere Filtermöglichkeit im Menü zu

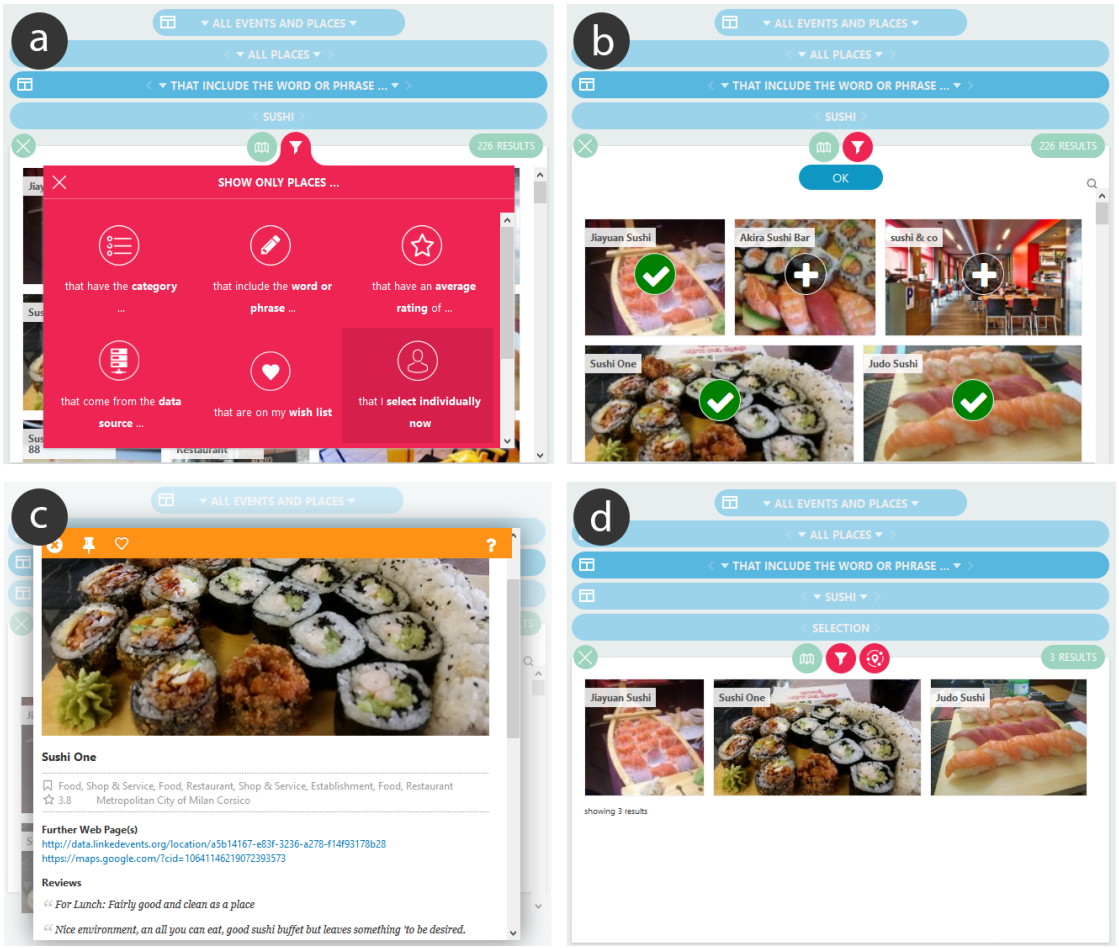

Abbildung 4.4: Manuelle Selektion: Möglichkeit der Erstellung jeder beliebigen Untermenge

realisieren. Das Prinzip der manuellen Selektion passt besser in das Konzept des Filtrierens, da es sich um ein Filtrieren nach ID handelt; somit ist die Entscheidung bei der Realisierung zu Gunsten dieser Variante gefallen.

Anschließend musste eine geeignete Darstellungsform gefunden werden, um einzelne Elemente auszuwählen. Einerseits könnte man dem Benutzer Checkboxen anbieten, um das Auswählen zu ermöglichen, andererseits könnte man wie Abbildung 4.4 (b) zeigt, große „Plus“-Knöpfe über die einzelnen Elemente darstellen. Obwohl der erste Ansatz den Vorteil hat, dass Checkboxen gängig in Benutzerschnittstellen sind und die Benutzer mit ihrem Erscheinungsbild sehr vertraut sind, habe ich mich für die zweite Option entschieden. Der Benutzer kann leichter erkennen, welche Elemente

ausgewählt wurden und ein größeres Element lässt sich gemäß Fitts Gesetz [21] schneller auswählen. Dieses Gesetz besagt, dass die benötigte Zeit um ein Objekt auszuwählen, nicht nur von dem Abstand, sondern gleichermaßen von dessen Größe abhängt.

4.3.4 Wie wurde es implementiert?

Im Prinzip stellt die manuelle Selektion einen Filter nach ID dar, bei dem die Bedingung für die Filteranfrage – in Hinblick auf die Aussagenlogik – aus einzelnen IDs besteht und mit dem Oder-Operator verknüpft sind. Es funktioniert ähnlich wie das Filtrieren nach Kategorie, mit der Ausnahme, dass nicht nur ein einzelner Wert spezifiziert werden kann, sondern eine Menge von Werten. Nach Aktivierung dieses Filters wird mithilfe von jQuery über alle Elemente im aktuellen Knoten „Plus“-Symbole gelegt. Die vom Benutzer selektierten IDs werden in einem Array zusammengefasst, der als Grundlage für den Inhalt des entstehenden Kindsknotens dient und an die Anfrageebene weitergegeben wird.

4.4 Multiples Pivoting

Multiples Pivoting ist eine Verallgemeinerung des im Abschnitt 4.2 beschriebenen, einfachen Pivoting: in diesem Fall navigiert der Benutzer nicht von einem Quellelement sondern von einer Quellmenge zu einer Zielmenge. Manche traditionelle Suchsysteme bieten ein sehr eingeschränktes Pendant zu multiplem Pivoting an: sie erlauben es zum Beispiel, nach Hotels zu suchen, die eine U-Bahn-Station in der Nähe haben. Hierbei wird diese Eigenschaft aber als ein Attribut modelliert, nach dem der Benutzer im eigentlichen Sinne filtrieren kann. Unser Ansatz soll nicht auf eine etwaige Einschränkung beruhen, sondern alle im Backend vorhandenen Relationen zwischen Mengen in der Benutzerschnittstelle abbilden können. Wie in Abbildung 4.5 zu sehen, ist es mit PE möglich, diejenigen Hotels, die in der Nähe von Orten, die etwas mit Sushi zu tun haben, anzuzeigen. Zudem ist eine sehr große Anzahl anderer Szenarien denkbar, die nicht alle gleichzeitig von dem oben beschriebenen, alternativen Ansatz abgedeckt werden können.

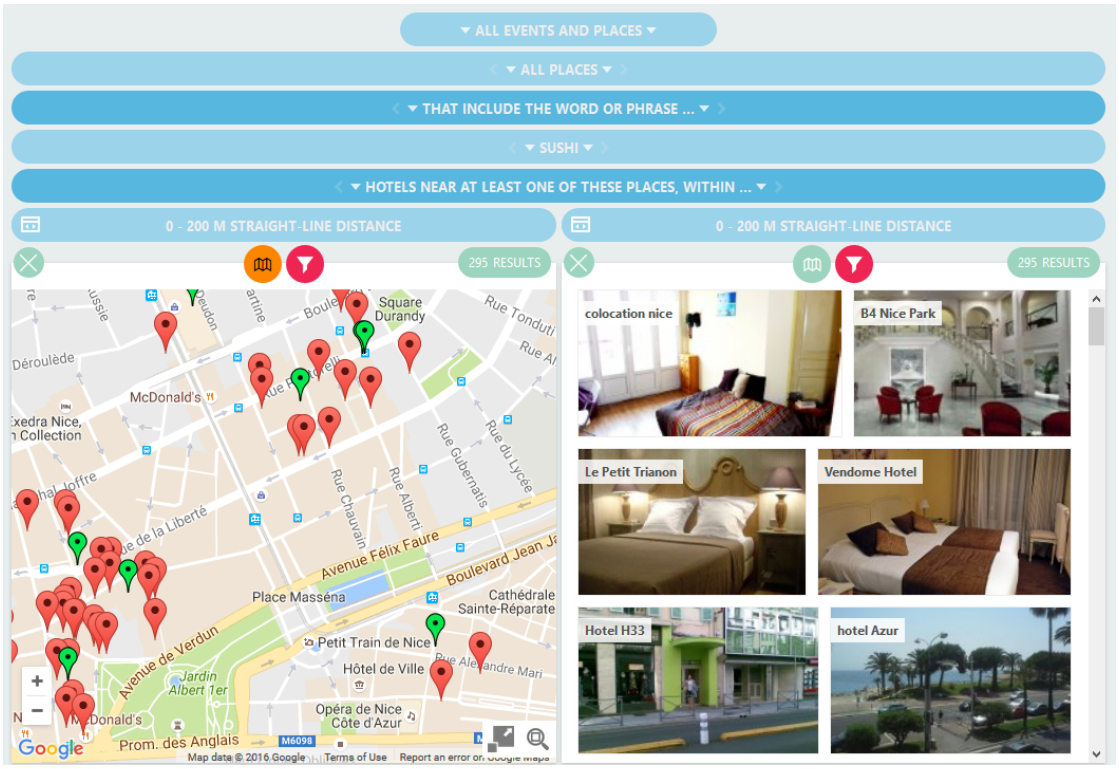

Abbildung 4.5: Multiples Pivoting in der Karten- und Listenansicht: auf der Kartenansicht werden die Orte, in deren Bezeichnung Sushi vorkommt, durch grüne Stecknadeln dargestellt — in der Nähe befindliche Hotels durch rote Stecknadeln.

4.4.1 Wie sieht die gewählte Lösung für den Benutzer im Detail aus?

Bei der Verallgemeinerung von einfachem Pivoting wird dem Benutzer zusätzlich die Möglichkeit gegeben, Pivoting auf Knoten mit mehreren Elementen anzuwenden, anstatt nur auf Elementknoten. In der aktuellen Implementierung ist multiples Pivoting zunächst auf Ergebnismengen mit höchstens 100 Elementen beschränkt — in den anderen Fällen wird der entsprechende Knopf ausgeblendet. Außerdem entsteht eine Problematik bei der Anzeige der Ergebnismenge, nämlich wie die Beziehung zum Quellelement angedeutet werden kann, auf die ich im nächsten Abschnitt genauer eingehen werde. Ansonsten ist aus der Sicht des Benutzers multiples Pivoting konsistent zum normalen Pivoting. In der Abbildung 4.5 wurde multiples Pivoting auf eine ganze Menge von Elementen angewandt.

4.4.2 Was sind die grundlegenden UI-Designentscheidungen?

An dieser Stelle tauchte die Frage auf, wie man Paare passender Elemente sowie deren Abstände andeuten kann. Wie in dem linken Ausschnitt der Abbildung 4.5 zu sehen ist, erscheint auf einer Karte die Antwort auf diese Frage sehr natürlich, da der Mensch die jeweiligen Abstände sehr gut interpretieren kann, aber aus der Liste gehen diese Relationen nicht hervor. Ich habe daher angenommen, dass Benutzer für diese Zwecke die Kartenansicht aufsuchen, da auf Ergebnislisten basierende Lösungsansätze ungeschickt und nicht besonders nützlich sein könnten.

4.4.3 Welche Möglichkeiten gibt es, diese Entscheidungen technisch umzusetzen; und was sind deren Vor- und Nachteile?

In Hinblick auf die technische Umsetzung von multiplem Pivoting ist es naheliegend, dies analog zum einfachen Pivoting zu gestalten. In Bezug auf die Komplexität der Anfragen sieht dies jedoch anders aus, da im schlimmsten Fall mit einer naiven Implementierung, diese linear zu der Anzahl der Quellelemente steigt. Es müssen alle Elemente berücksichtigt, von denen dann entlang einer Relation komplexe Anfragen berechnet werden. Auf diesen Teilaspekt und die Verbesserung der Effizienz von Anfragen multiplem Pivotings werde ich in dieser Arbeit nicht eingehen, da dies hauptsächlich das Backend betrifft und von Kai-Dominik Kuhn in seiner Masterarbeit [19] (Kapitel 4.6.1) bearbeitet wird.

4.4.4 Wie wurde es implementiert?

Aus Sicht der Benutzerschnittstelle wurde das multiple Pivoting – bis auf folgende Ausnahme – analog zum einfachen Pivoting realisiert: damit die Quellmenge von der Anfrageebene nicht unnötigerweise erneut ermittelt werden muss, wird zunächst automatisch ein verborgener, zwischenliegender Knoten erstellt. Dieser hat den gleichen Typ wie die in Abschnitt 4.3 beschriebenen Knoten und enthält höchstens 100 Elemente der vorliegenden Quellmenge. Das tatsächliche Pivoting wird anschließend auf diesen zwischenliegenden Knoten durchgeführt. Somit muss lediglich eine Liste von Quellelementen anstatt einer Beschreibung der Quellmenge an die Anfrageebene gesendet werden, wodurch die Performanz bei der Ermittlung einer Ergebnismenge in Hinblick auf multiplem Pivoting gesteigert wird.

4.5 Lesezeichen

Diese Erweiterung erlaubt es dem Benutzer, einen bestimmten Zustand eines Explorationsbaumes in Form eines Lesezeichens zu speichern. Der Link kann zur späteren Verwendung oder für kollaborative Zwecke dienen. Des Weiteren ermöglicht diese Funktion auch das Speichern von Zuständen als Antworten auf QuickStart-Fragen, die im Abschnitt 4.7 beschrieben werden und ist daher ein grundlegender Baustein für die spätere Implementierung von QuickStart-Applikationen. Dadurch, dass die Lesezeichen einzelne Zustände speichern können, wird eine einfache Implementierung von „Undo“ und „Redo“-Operationen ermöglicht, die den Benutzer Interaktionen zurücknehmen oder wiederholen lassen. Andererseits können durch Lesezeichen auch soziale Aspekte bedient werden: mithilfe von „Share Buttons“ ist es Nutzern möglich, ihre Suche mit anderen zu teilen. Die durch die Erkundung erzielten Ergebnisse können via Facebook, Google Plus, Twitter oder per E-Mail an Freunde oder Bekannte geschickt werden, die ihrerseits Vorschläge geben können oder auf ihren eigenen Interessen basierend das System weiter erforschen können. Ein positiver Nebeneffekt könnte sein, dass sich die betreffende Webapplikation weiter viral verbreiten kann, wenn Benutzer mit dieser Funktion interagieren.

4.5.1 Welche Ergebnisse zur Bedienbarkeit und zur Nützlichkeit lagen dem Design zugrunde?

Sowohl von Geschäftsleuten als auch von typischen Benutzern erhielten wir bei allen Präsentationen in den Jahren 2012 bis 2015 (Abschnitt 3.1) sehr gute Resonanz auf entsprechende Implementierungen von Lesezeichen aus früheren Versionen, die vor dieser Forschungsarbeit entstanden. Die Funktion, einzelne Aktionen rückgängig zu machen, wurde von vielen Benutzern explizit gefordert: insbesondere nach dem unbeabsichtigten Löschen eines Knotens muss diese Möglichkeit zur Verfügung stehen.

Die Möglichkeit einer Speicherung des aktuellen Zustands ist ein wichtiges Grundfeature, welches auch im Rahmen der Evaluation bezüglich der Erlernbarkeit von PFB untersucht wurde [16]. Von 100 getesteten Personen, die eine Abfolge von Interaktionen in PFB durchführten und anschließend den aktuellen Zustand mit einem Lesezeichen abspeichern mussten, schafften es zwar nur 42 Personen das endgültige Lesezeichen zu generieren, aber dies ist auch genau die Anzahl an Personen, die überhaupt zu diesem Schritt gekommen sind. Aus diesem Grund ist davon auszugehen, dass das Setzen eines Lesezeichens für die meisten Benutzer grundsätzlich kein Hindernis darstellt.

Bei anderen Benutzerschnittstellen, in denen Benutzer komplexe Konfigurationen vornehmen können, bestätigte sich die Nützlichkeit, aktuelle Zustände zu speichern. In TrailMap können Benutzer mit unterschiedlichen Interaktionsmöglichkeiten Landkarten und darin erhaltene Plätze erkunden. Die aktuellen Zoom-, Positions- und Filtereinstellungen werden mithilfe eines Lesezeichens für eine spätere Verwendung gespeichert. Die durchgeführte Benutzerstudie [31] konnte zeigen, dass das Speichern der Suchverläufe bei den Benutzern beliebt war.

4.5.2 Wie sieht die gewählte Lösung für den Benutzer im Detail aus?

Für den Benutzer sieht diese Erweiterung wie folgt aus: zu jedem Zeitpunkt kann er auf das Lesezeichen aus der Browserleiste zugreifen, das den aktuellen Zustand eindeutig identifiziert. Dieses Lesezeichen kann anderen Personen für kollaborative Zwecke direkt weitergegeben werden, um nicht auf Methoden wie beispielsweise das Erstellen von Bildschirmabzügen oder manuellen Notizen angewiesen zu sein. Die durch die Lesezeichen abgebildeten Explorationsbäume können durch andere Nutzer weiterbearbeitet oder ergänzt werden, was durch das Erstellen und Verschicken von Bildschirmabzügen nicht möglich wäre. Der Benutzer kann Lesezeichen auch für eine spätere Verwendung abspeichern, die Vorteile sind analog zu denen der kollaborativen Unterstützung. Des Weiteren ist es ihm möglich mit einem aufgespannten Explorationsbaum eine Frage der QuickStart-Applikationen zu beantworten — die Funktionalität, einzelne Zustände abzuspeichern, ist für die Erstellung von QuickStart-Applikationen daher essenziell.

4.5.3 Was sind die grundlegenden UI-Designentscheidungen?

Die grundlegende UI-Designentscheidung bei der Umsetzung der Lesezeichen war, wie man den Benutzer auf die Existenz dieser Funktionalität hinweist. Eine Möglichkeit ist es, einen dedizierten Knopf hierfür einzubauen: dieser würde jedoch die Benutzerschnittstelle weiter überladen. Eine andere Möglichkeit wäre, die Adresszeile des Browsers mit jeder Benutzerinteraktion zu ändern und diese synchron mit dem Inhalt zu halten. Dabei kann der Benutzer erkennen, dass der Zustand durch den Link identifiziert wird, ebenso sind Benutzer damit vertraut, dass die Adresszeile des Browsers stets den aktuellen Inhalt widerspiegelt. Aus diesen Gründen habe ich mich bei der Implementierung für die zuletzt genannte Lösung entschieden.

4.5.4 Welche Möglichkeiten gibt es, diese Entscheidungen technisch umzusetzen; und was sind deren Vor- und Nachteile?

Art der Speicherung Für die Speicherung gibt es zwei Möglichkeiten: (a) Materialisierung des aktuellen Zustands, bei der die Einzelheiten des aktuellen Explorationsbaums gespeichert werden und (b) inkrementelles Speichern der Informationen über die Benutzeraktionen, die zum aktuellen Zustand führten.

Für eine Analyse im Rahmen einer Evaluation sind die Informationen, wie ein bestimmter Zustand erreicht wurde, aufschlussreicher als welche Zustände erreicht wurden. Des Weiteren gehen bei der Möglichkeit (a) die Informationen verloren, wie ein bestimmter Zustand erreicht wurde. Man kann davon absehen, dass die Performanz bei der schrittweisen Rekonstruktion etwas schlechter ist, als die eines materialisierten Zustandes, da die frühere Implementierung der PFB-Benutzerschnittstelle bereits ausreichend performant war. Daher wurde – wie bei dem PFB-Prototyp aus der Bachelorarbeit – auch im Rahmen der Implementierung von ExplorMI 360 die Methode (b) gewählt.

Eine Erweiterung zur Implementierung der Bachelorarbeit stellte die einstellbare Begrenzungsmöglichkeit dar, bis zu welcher Aktion der Explorationsbaum wiederhergestellt wird. Um das Ganze anschaulich zu machen: Das Lesezeichen „3cixty.com/#bookmark=8gmVZ-17“ spielt die ersten 17 Nutzeraktionen der Session mit der „8gmVZ“ ab. Diese Neuerung ermöglicht eine unkomplizierte Implementierung der „Undo-Funktion“, die es dem Benutzer gestattet, die letzte Aktion rückgängig zu machen. Im oberen Fall würden nach Anwendung der Funktion nur die ersten 16 Nutzeraktionen der Session durchgeführt werden und der Benutzer würde den aufgespannten Explorationsbaum mit dem dazugehörigen Lesezeichen „3cixty.com/#bookmark=8gmVZ-16“ sehen.

Mehrbenutzerunterstützung Ein Lesezeichen muss sich beliebig oft öffnen lassen. Würde das System die Schritte einfach hochzählen, so gäbe es bei simultaner Benutzung desselben Lesezeichens durch mehrere Personen Duplikate in Bezug auf die Session und die Nummerierung der Schritte. Um dieses Problem zu umgehen, wird nach dem Öffnen eines Lesezeichens ein neues Lesezeichen generiert, das alle bisherigen Schritte des alten durchgeht und erneut ausführt. Durch diesen Trick, dass jeder Benutzer an einer Kopie des Lesezeichens arbeitet, wird verhindert, dass mehrere Benutzer dasselbe Lesezeichen editieren.

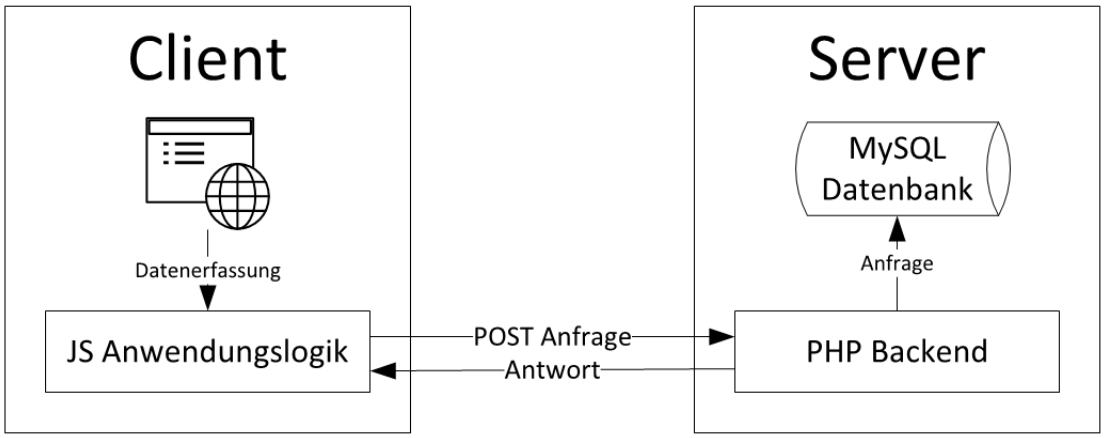

Abbildung 4.6: Architektur der Speicherung von Lesezeichen

4.5.5 Wie wurde es implementiert?

Aus technischer Sicht funktioniert diese Erweiterung – wie man in der Abbildung 4.6 erkennen kann – wie folgt: bei jeder Benutzerinteraktion werden alle für die Rekonstruktion notwendigen Daten per JavaScript erfasst und in einem Objekt gespeichert. Dieses Objekt wird mittels einer POST-Abfrage an das Backend geschickt; die dort empfangenen Daten werden mithilfe von PHP in eine dafür vorgesehene Tabelle der MySQL-Datenbank geschrieben. Ruft ein Benutzer ein Lesezeichen auf, wird eine POST-Anfrage an das Backend geschickt, die die angeforderte Session ID und die Anzahl der zu rekonstruierenden Schritte enthält. Dieses sendet alle notwendigen Informationen zurück, die für die Wiederherstellung des Zustandes bis zum angegebenen Schritt erforderlich sind. Die Benutzerschnittstelle kann nun die Informationen verarbeiten und alle empfangenen Schritte abarbeiten, bis der endgültige Zustand erreicht wurde.

Die MySQL-Datenbank speichert alle für die Rekonstruktion eines Lesezeichens notwendigen Informationen ab. Darüber hinaus werden noch weitere Informationen erfasst, die für spätere Analysen in Hinblick auf Benutzerstudien hilfreich sein können. Um dies zu realisieren, wurden folgende Spalten in der Datenbank modelliert, wobei jede eingefügte Zeile eine einzelne Aktion des Benutzers widerspiegelt.

Spalten, die für die Rekonstruktion notwendig sind

1. **log_id** Fortlaufende Nummerierung der Einträge

2. **bookmark_id** Bezeichner des Bookmarks

3. **step** Fortlaufende Nummerierung innerhalb des Bookmarks

4. **log_type** Art des Logeintrags, beispielsweise CREATE_NODE, DEL_NODE oder MINIMIZE_NODE

5. **p0 - p9** Hier werden detaillierte Informationen gespeichert, Bedeutung hängt von log_type ab

6. **project_version** Versionsnummer der Benutzerschnittstelle

Spalten, die für weitere Analysen hilfreich sein können

1. **user_id** Eindeutige 3CIXTY-Identifikations ID des Benutzers, der dieses Lesezeichen kreirt hat

2. **screen_width** Maximale Bildschirmbreite des Benutzers

3. **screen_height** Maximale Bildschirmhöhe des Benutzers

4. **window_width** Aktuelle Breite des Browserfensters

5. **window_height** Aktuelle Höhe des Browserfensters

6. **timestamp** Erstellungsdatum des Eintrags

4.6 Wunschliste

Diese Erweiterung ermöglicht es dem Benutzer, ausgesuchte Elemente auf seiner eigenen Wunschliste zu speichern. Einerseits hilft es ihm, auf einfache Weise, Veranstaltungen oder Orte, die er als interessant bewertet für ein späteres Wiederfinden zu speichern; andererseits ermöglicht die Wunschliste, Ergebnisse an andere Applikationen weiterzuleiten. Ein Benutzer könnte beispielsweise zuhause auf einem Computer seine Reise mit der Webapplikation von ExplorMI 360 planen, einige Veranstaltungen oder Orte auf seine Wunschliste packen und sich anschließend mit einer anderen – mobilen – Applikation von einem Ort zum nächsten navigieren lassen.

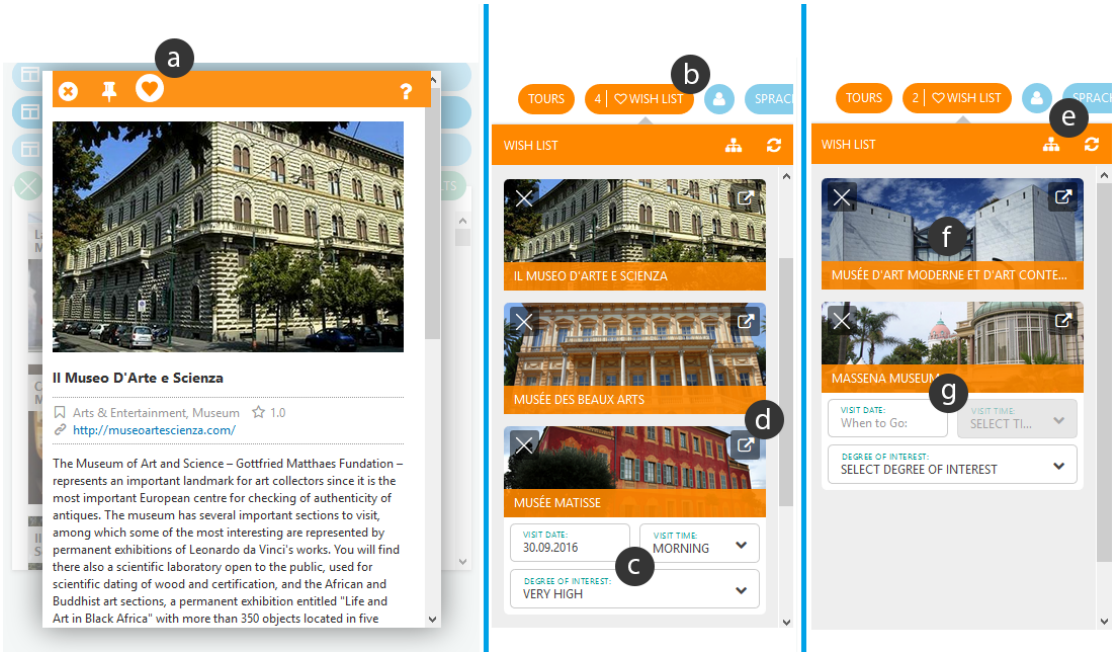

Abbildung 4.7: Darstellung der möglichen Benutzerinteraktionen mit der Wunschliste (Erläuterung im Text)

4.6.1 Welche Ergebnisse zur Bedienbarkeit und zur Nützlichkeit lagen dem Design zugrunde?

Die Wunschliste ist ein Kernteil der Konzeption von ExplorMI 360, da sie in gewisser Hinsicht notwendig ist, um die Explorationsergebnisse in einer geeigneten Listendarstellung speichern zu können und um die Zusammenarbeit zwischen Webapplikation und mobiler App zu ermöglichen. Die Notwendigkeit und Nützlichkeit dieser Erweiterung wurde in den vielen Vorführungen nie angezweifelt.

4.6.2 Wie sieht die gewählte Lösung für den Benutzer im Detail aus?

Alle Elemente, wie beispielsweise Veranstaltungen oder Orte, lassen sich aus dem aufgespannten Explorationsbaum in eine lineare Liste speichern. Wie auf Abbildung 4.7 zu sehen ist, kann der Benutzer in der Detailansicht eines Elements (oder auch in der Schnellansicht), die Schaltfläche mit einem Herz (a) auswählen und somit ein Element entweder auf die Wunschliste setzen oder wieder davon entfernen. Die

Schnellansicht ermöglicht es dem Benutzer, sich in einem überlappenden Fenster einen schnellen Überblick von einem Element zu verschaffen, ohne sofort einen neuen Knoten im Explorationsbaum anlegen zu müssen. Die eigene Wunschliste (b), die sich in rechten, oberen Menü befindet, kann man ein- und ausblenden. Zu jedem einzelnen Eintrag kann der Benutzer ein Datum und eine Zeitspanne (c) angeben, an dem er die Veranstaltung oder den Ort besuchen möchte und wie stark das Interesse für den jeweiligen Eintrag ist. Wenn die Benutzer ausgehend von einem auf der Wunschliste gespeicherten Eintrag weiter erkunden möchten, können sie das machen, indem sie auf die entsprechende Schaltfläche (d) klicken. Alternativ ist es mit einem Klick auf das Baumsymbol möglich, alle Elemente der Wunschliste dem Explorationsbaum hinzuzufügen (e). Anschließend können Benutzer wie gewohnt weitere Pivoting-Funktionen darauf anwenden, beispielsweise naheliegende Restaurants oder Hotels suchen.

Die Wunschliste ist in der 3cixty-Plattform gespeichert und mit dem Benutzerkonto verknüpft. Somit kann von jeder anderen 3cixty-Applikation auf die Wunschliste zugegriffen werden — lesend als auch schreibend. Die beiden Smartphone-Applikationen für Android und iOS, „ExplorMI Mobile“, zeigen die Wunschliste und führen die Benutzer zu den vorgemerkten Veranstaltungen und Orten.

4.6.3 Was sind die grundlegenden UI-Designentscheidungen?

Um eine größere Anzahl an Elementen in der Wunschliste zulassen und verwalten zu können, sollte jedes einzelne Element zwei unterschiedliche Ansichten haben: (f) eine platzsparende Ansicht, bei der nur das Vorschaubild und die Bezeichnung sichtbar sind und (g) eine maximierte Ansicht, bei der alle Details bezüglich der Planung angezeigt werden. Dies ermöglicht feinkörnige Einstellungsmöglichkeiten der Wunschliste.

Ein weiterer Aspekt in Hinblick auf die Wunschliste ist die Möglichkeit, enthaltende Elemente nach gewissen Kriterien wie beispielsweise „Datum der Veranstaltung“, „Datum des Hinzufügens“ oder „Entfernung zum aktuellen Standort“ zu sortieren und nach Entitätstypen zu gruppieren. Da das Backend zur Wunschliste diese Funktionalität nicht unterstützte und auch erforderliche Informationen wie den Standort oder das Datum des Hinzufügens nicht liefert, war eine Implementierung im Rahmen der Masterarbeit nicht mehr möglich, aber sie soll bald erfolgen.

Die Aufgabe, eine im Backend abgespeicherte Wunschliste, bei der man unterschiedliche Elemente hinzufügen und entfernen kann, zu programmieren erscheint trivial. Was soll aber zum Beispiel passieren, wenn der Benutzer nicht eingeloggt ist? Der simple Lösungsansatz wäre, die Funktion in diesem Fall nicht zur Verfügung zu

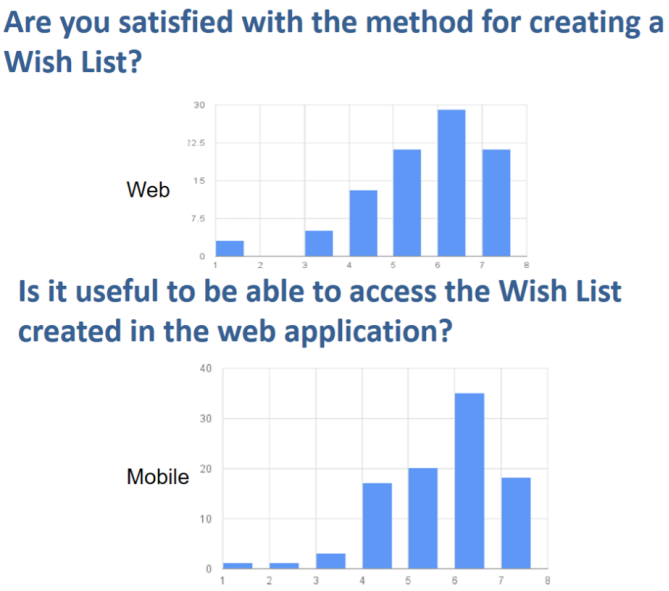

Abbildung 4.8: Einschätzungen in Bezug auf die Wunschliste; x-Achse: Stufe der Akzeptanz auf einer Likert-Skala von 1 (starke Ablehnung) bis 7 (starke Zustimmung), y-Achse: Anzahl der Personen

stellen. Alternativ könnte die Funktion wie gewohnt verfügbar sein und sobald sich der Benutzer anmeldet, werden die Elemente mit den bisherig gespeicherten Elementen zusammengeführt — die Vereinigungsmenge wird gebildet. Außerdem muss bedacht werden, dass die Wunschliste erhalten bleiben sollte, selbst wenn der Benutzer sich nicht angemeldet hat, um bei einem späteren Besuch die gespeicherten Elemente wiederzufinden. Dies könnte im internen Speicher des Browsers, dem „LocalStorage“, gespeichert werden; wenn der Benutzer – oder jemand anders – mit dem gleichen Browser die Applikation wieder aufruft, ließe sich der vorherige Stand der Wunschliste wiederherstellen.

4.6.4 Welche Rückmeldung bekamen wir?

Die durch das PoliMi durchgeführte Benutzerstudie (Abschnitt 3.1) machte deutlich, dass die Wunschliste von den Teilnehmern gut angenommen wurde. Wie auf Abbildung 4.8 zu sehen ist, häuften sich die Antworten der Nutzer auf die Frage bezüglich Nützlichkeit der Wunschliste, auf sechs Punkte auf einer Skala von eins bis sieben, wobei mehr Punkte für eine höhere Zufriedenheit standen. Auch auf die Frage, welche Features beibehalten werden sollten, stimmten 70 von 95 Befragten

für die Wunschliste und niemand befürwortete das Entfernen dieser; von daher deckte sich das Ergebnis der Benutzerstudie sowohl mit den Rückmeldungen aus den Vorführungen als auch mit unseren Erwartungen.

Benutzertests in kleinerem Rahmen, die auf usertesting.com durchgeführt wurden, haben gezeigt, dass es hilfreich ist den aktuellen Zustand einzelner Elemente im Explorationsbaum anzuzeigen. Es soll sichtbar gemacht werden, ob sich das Element bereits auf der Wunschliste befindet: ein ausgefülltes Herz weist den Benutzer darauf hin; analog hierzu ein leeres Herz, wenn das nicht der Fall ist. Diese Empfehlung wurde bereits umgesetzt.

Es liegen weitere Rückmeldungen vor, aus denen Ideen für die Zukunft hervorgehen. So ergaben Gespräche mit ErgoSign, dass dem Benutzer die Möglichkeit gegeben werden sollte, benutzerdefinierte Wunschlisten anzulegen, beispielsweise „Sehenswürdigkeiten“, „Shoppingtour“ oder „Für die Kinder“. Dies würde eine strukturiertere Nutzung der Wunschliste ermöglichen und dem Benutzer bei der Planung seines Aufenthaltes unterstützen. Diese Idee wurde von Teilnehmern bei Vorführungen bestärkt und es ist bereits vorgesehen, diese Erweiterung zu realisieren.

Die Wunschliste bietet zwar die Funktionalität einen Aufenthalt zu planen, und zu jedem darauf befindlichen Element ein Datum und eine Uhrzeit anzugeben, aber die Einzelansicht eines Elements könnte ausführlichere und detailliertere Daten anzeigen. Im Moment sind diese Angaben nur sichtbar, wenn sich der Benutzer die Elemente im Explorationsbaum anzeigen lässt.

Es wurde von den Benutzern der PoliMi-Studie, und auch unabhängig davon von ErgoSign angemerkt, dass Wege zur Manipulation bzw. Sortierung der Daten innerhalb der Wunschliste erwünscht wären.

4.6.5 Wie wurde es implementiert?

Das Backend und die API für die Wunschliste wurde von den 3cixty-Teilnehmern des Forschungsinstituts Inria bereitgestellt.[3] Mit JavaScript werden POST-Anfragen an das Backend geschickt, die den aktuellen Inhalt der Wunschliste abfragen, neue Elemente hinzufügen oder bestehende aktualisieren. Diese Aktionen können durch das erforderliche Attribut „action“ beschrieben werden. Je nach ausgewählter Aktion müssen weitere Details angegeben werden, die ID des zu bearbeitenden Elements sowie die Daten die geändert oder hinzugefügt werden sollen. Zusätzlich

[3] `https://api.3cixty.com/v2/tray`

muss der Applikationsschlüssel unter dem Attribut „key“ gesetzt werden, damit die Registrierung bei dem Service erhalten wird.

Um die bereits erwähnte Funktionalität, dass die Wunschliste auch im nicht-eingeloggten Zustand verfügbar ist, zu gewährleisten, wurde ein „Junk Token“ hinzugefügt. Im Grunde verfügt jeder Benutzer über einen Session Token, solange dieser jedoch nicht eingeloggt ist, wird der Session Token nicht mit einem Konto assoziiert und als „Junk Token“ bezeichnet. Erst nach dem Einloggen wird diesem Junk Token ein Benutzer zugewiesen und alle bereits durchgeführten Aktionen werden serverseitig so behandelt, als wäre der Benutzer stets angemeldet gewesen. So gelingt es, die temporäre Wunschliste mit der des eingeloggten Benutzers zusammenzuführen. Die vollständige Dokumentation der API für die Wunschliste wurde von Inria online zur Verfügung gestellt.

4.7 QuickStart-Applikationen

Eine QuickStart-Applikation ist eine Sammlung von Fragen zu einem bestimmten Thema. Als Antwort auf diese Fragen dient ein vordefinierter Explorationsbaum, der in der Form eines Lesezeichens (siehe Abschnitt 4.5) gespeichert wird. Der Benutzer kann sowohl eine bestehende QuickStart-App benutzen als auch eigene Applikationen erstellen und diese für andere Benutzer zur Verfügung stellen.

Ebenso können QuickStart-Applikationen von anderen Webseiten aus verlinkt – und später auch eingebettet – werden. Die Möglichkeit der Erstellung eigener Applikationen fördert somit soziale Aspekte für Nutzer, ein Ökosystem von QuickStart-Applikationen könnte die Verbreitung von PE vorantreiben. Aber auch gewerbliche Nutzer können sich auf der Plattform besser präsentieren, indem sie sich mithilfe einer eigenen Applikation vorstellen und diese in ihre Webpräsenz einbinden.

Da die Möglichkeit für Benutzer der PE-Benutzerschnittstelle, eigene QuickStart-Applikationen zu erstellen, vom 3cixty-Unterauftragnehmer Deip Designs übernommen wurde, gehe ich in dieser Arbeit nicht näher darauf ein, sondern nur auf die von mir geleistete Vorarbeit und die oben beschriebenen Aspekte.

4.7.1 Welche Ergebnisse zur Bedienbarkeit und zur Nützlichkeit lagen dem Design zugrunde?

Die Idee für diese Erweiterung kam bei Benutzertests mit der Applikation auf, als ein Hotelier sagte, dass er gerne einen „Concierge“ hätte, der die häufigsten Fragen

seiner Gäste beantwortet (Abschnitt 3.1). Die bereits beschriebenen Vorteile der Lesezeichen könnten dadurch verstärkt werden, wenn man diese auf einfache Art und Weise verwalten könnte. Mit einer QuickStart-Applikation kann man die Beantwortung einer beschränkten Menge von Fragen mit wenig Lernzeit ermöglichen, indem man selbst definierte Fragen jeweils mit einem erstellten Lesezeichen beantwortet. Diese Fragenkataloge können beispielsweise in Hotels auf einem iPad für Besucher zugänglich gemacht werden. Dadurch können Besucher die am häufigsten gestellten Fragen erkunden, und unter Umständen ihre Anliegen gleich beantworten.

Aus den Rückmeldungen, die wir über die Benutzerschnittstelle erhalten haben, wie die Studie in meiner Bachelorarbeit [26] und die Veröffentlichung über die Erlernbarkeit von PFB [16], geht hervor, dass Benutzer eine gewisse Zeit benötigen, die neuartigen Funktionen zu erlernen. Die grundlegende Überarbeitung des Interaktionsdesigns, bei der PE an die facettierte Suche angenähert wurde (Abschnitt 3.4), konnte bereits einige dieser Schwächen ausgleichen. Da Benutzer mit einigen neuartigen Interaktionen, wie beispielsweise die nebeneinanderliegende Anzeige mehrerer Ergebnismengen oder multiplem Pivoting, nicht vertraut sind, kann es passieren, dass sie diese nicht sofort entdecken und somit gar nicht erkennen, welche Vorteile ihnen PE bringen kann. QuickStart-Applikationen setzen genau an diesem Punkt an und versuchen den Benutzern mithilfe von ausformulierten Fragen auf die komplexe Funktionalität von PE hinzuweisen — für neue Benutzer ist es nämlich leichter, einen vorgegebenen Explorationsbaum zu verstehen als selbst einen zu erstellen. Dadurch erkennen die Benutzer nicht nur, was mit PE realisiert werden kann, sondern sie werden vertrauter im Umgang mit den neuartigen Erweiterungen.

4.7.2 Wie sieht die gewählte Lösung für den Benutzer im Detail aus?

Abbildung 4.9 zeigt den Aufbau von QuickStart-Applikationen. Der Benutzer erhält die Übersicht der unterschiedlichen Applikationen (a) beim ersten Besuch der Webapplikation ExplorMI 360 und kann sie im späteren Verlauf immer wieder über die Schaltfläche „QuickStart“ aufrufen. Nachdem der Benutzer eine Applikation ausgewählt hat, stehen ihm unterschiedliche Fragen zur Auswahl, die in Sektionen unterteilt sind (b) — jede dieser Fragen ist mit einem Lesezeichen versehen, das als Antwort darauf dient. Nach dem Auswählen einer Frage erscheint ein vordefinierter Explorationsbaum (c). Kontextspezifische Tooltipps sind eine weitere Hilfe, die dem Benutzer bei der Interpretation der angebotenen Antwort unterstützt, auf die ich in Abschnitt 4.8 näher eingehen werde. Eine Applikation ist flexibel konfigurierbar: der Titel, das Titelbild, die Beschreibung und die Titel der Sektionen inklusive der

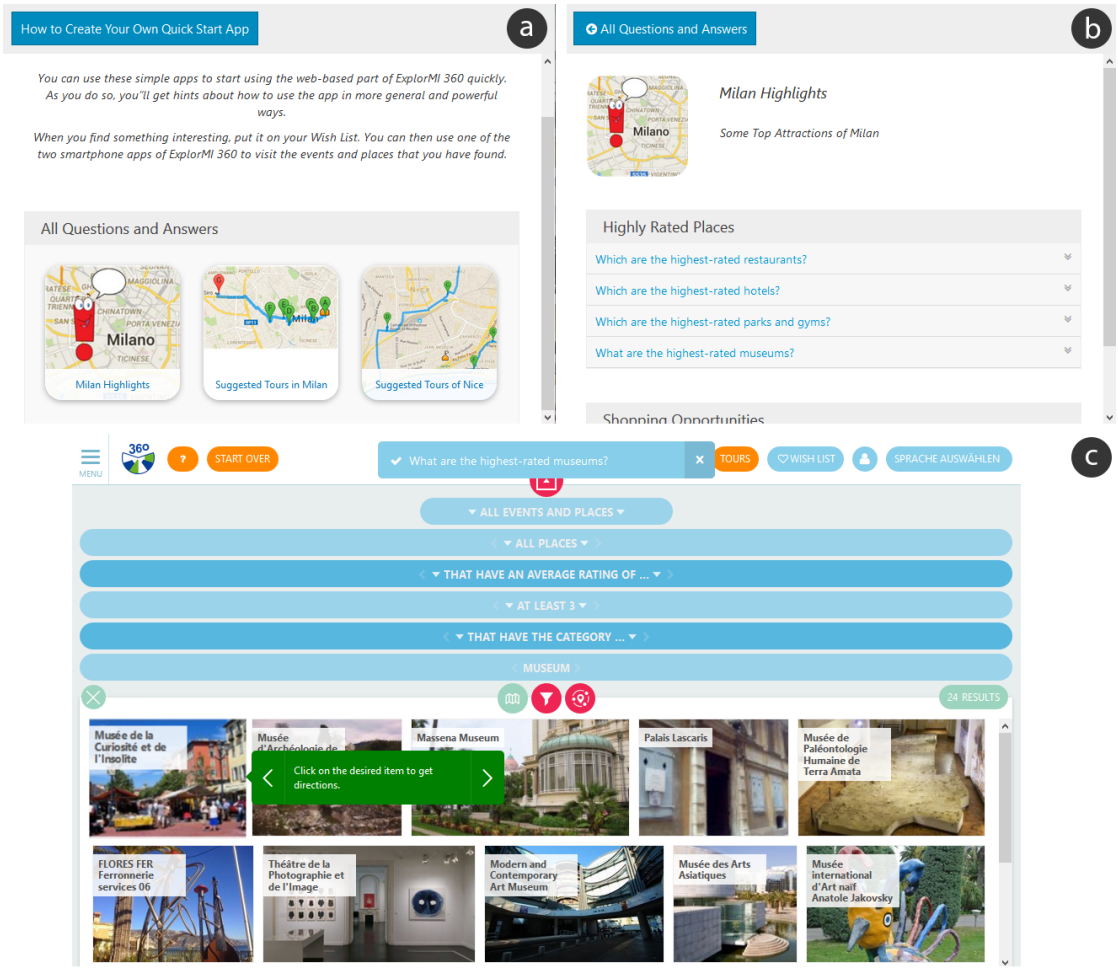

Abbildung 4.9: Funktionsweise von QuickStart-Applikationen

beinhaltenden Fragen und Bookmarks sind beliebig einstellbar. Eigene Applikationen können auch gelöscht werden und sind für den Ersteller jederzeit editierbar, sodass sich diese auch im Nachhinein erweitern lassen.

4.7.3 Was sind die grundlegenden UI-Designentscheidungen?

Es stehen zwei Möglichkeiten zur Verfügung, die Sammlung von QuickStart-Apps zu verwalten: (a) statisch, bei dem lediglich vorgegebene, Applikationen zugelassen sind oder (b) dynamisch, bei dem Benutzer selber Applikationen vorschlagen können. Während bei der ersten Lösung die Applikationen sorgfältig entworfen und

ausgewählt werden, muss bei Letzterem eine Moderation implementiert werden, um böswillige Applikationen zu vermeiden. Gleichzeitig ermöglicht (b) ein Gedeihen des Ökosystems von Applikationen und fördert Diversität, da verschiedene Benutzer unterschiedliche Interessen haben. Die Einschränkung von (a), dass Benutzer keine eigenen Applikationen entwerfen können hat uns dazu bewegt, die zweite Variante zu bevorzugen.

Eine weitere UI-Designentscheidung war, ob sich ein antwortendes Lesezeichen in einem neuen Fenster öffnen sollte. Einerseits sind mehrere offene Fenster schwerer zu überblicken, andererseits wird der aktuelle Zustand der Benutzerschnittstelle nicht durch das neu geöffnete Lesezeichen überschrieben. Da man nach dem Öffnen eines Lesezeichens dem Benutzer die Möglichkeit geben kann, zum vorigen Zustand zurückzukehren, habe mich für das Öffnen im gleichen Fenster entschlossen.

Wenn sich der Benutzer für eine Applikation entschieden hat und anschließend nochmal QuickStart öffnet, kann er entweder zu der Applikationsübersicht gelangen oder zur zuletzt ausgewählten Applikation geleitet werden. Da die Applikationen themenspezifisch sind und sich der Benutzer bereits für eine Applikation entschieden hat, ist es sinnvoll, ihm die ausgewählte App anzuzeigen. Über den Knopf „All Quick Start Apps“ (siehe Abbildung 4.9 (b)) kann er jederzeit zum Auswahlmenü gelangen, wenn er andere Applikationen auswählen möchte.

4.7.4 Welche Rückmeldungen erhielten wir nach der Implementierung dieser Erweiterung?

Die ersten Implementierungen der QuickStart-Applikationen wurden vielen Leuten vorgestellt, darunter ErgoSign und der CeBIT Innovation Award Jury (siehe Abschnitt 3.1). Die allgemeine Rückmeldung war positiv, aber die Realisierung des Systems sollte vereinfacht werden.

Ein weiterer Aspekt wäre eine semi-automatische, templatebasierte Erstellung von Applikationen. So müsste ein Hotelbesitzer seine App nicht von Grund auf neu schreiben, sondern die am häufigsten verwendeten Fragen wären schon voreingetragen. Der Besitzer hätte die Möglichkeit, die Fragen weiter aufzubereiten, zu ändern und neue hinzuzufügen. Mit diesem Ansatz ließen sich Applikationen vollautomatisch erstellen, die grundlegende Aspekte abdecken, zum Beispiel die eines Hotels. Die anschließende Aufbereitung durch den Ersteller ergänzt die fehlenden Informationen, sodass die gewünschten Details mithilfe einer Applikation abgebildet werden können.

4.7.5 Welche Möglichkeiten gibt es, diese Entscheidungen technisch umzusetzen; und was sind deren Vor- und Nachteile?

Die Speicherung der Inhalte einer QuickStart-Applikation könnte auf zwei Arten erfolgen: in einer Konfigurationsdatei, die die dafür notwendigen Informationen enthält oder in einer relationalen Datenbank. Der erste Ansatz hat den Vorteil, flexibler zu sein. Änderungen in der Entwicklungsphase können schneller implementiert werden, da hierbei nur einzelne Variablen hinzugefügt, bearbeitet oder gelöscht werden. Andererseits ist die Wartung sowie Performanz bei einer größeren Anzahl an Einträgen besser, wenn eine relationale Datenbank zum Einsatz kommt. Außerdem kann die Erzeugung von QuickStart-Applikationen durch den Benutzer leichter in einer Datenbank gespeichert werden. Da es zunächst wichtig war, eine erste Version von QuickStart-Applikationen zu entwerfen und die Grundfunktionalitäten zu testen, entschied ich mich für den Einsatz einer Konfigurationsdatei. Sobald die benötigte Struktur der Daten klar wurde, habe ich es dem 3CIXTY-Unterauftragnehmer Deip Designs überlassen, die Datenbanklösung zu implementieren.

Listing 4.1: Beispiel einer Konfiguration einer QuickStart-Applikation

```
1  // EXPLORE MILAN
2  var app = new App();
3  app.label = 'Explore Milan';
4  app.description = 'Plan your visit to the city of Milan
       ';
5  app.image = 'img/apps/milano.png';
6
7  var section = new Section();
8  section.label = 'Simple Questions';
9  app.add_section(section);
10
11 var bookmark = new Bookmark('What events and places have
12   something to do with <em>Japan</em>?', 'k9N-16');
13 section.add_bookmark(bookmark);
```

4.7.6 Wie wurde es implementiert?

In der ersten Version der Implementierung wurde der gesamte App Store in einer Konfigurationsdatei zusammengefasst, die alle Attribute der Applikationen modelliert und beinhaltet. Der Codeausschnitt 4.1 zeigt eine Beispielinstanziierung

von „Explore Milan“, in der zunächst eine Applikation mit einem Bezeichner, einer Beschreibung und einem Bild initialisiert wird. Anschließend wird dieser eine Sektion zugewiesen und eine Frage hinzugefügt. In der späteren Version, die vom 3CIXTY-Unterauftragnehmer Deip Designs weiterentwickelt wurde, befinden sich die beschriebenen Attribute in einer Datenbank.

4.8 Kontextspezifische Tooltipps

Einerseits gibt es die Tooltipps, die nach dem Schweben mit der Maus über ein Element erscheinen und dessen Funktionalität erläutern. Diese Art von Tooltipps wurden über alle Versionen von PE hinweg gepflegt, aber da diese nicht innovativ und aus technischer Sicht trivial sind, gehe ich in dieser Arbeit nicht darauf ein. Dieser Abschnitt beschäftigt sich mit der Art von Tooltipps, abhängig vom aktuellen Zustand des Explorationsbaums auf gerade verfügbare Funktionalität hinweist und diese erklärt. Bei webbasierten Anwendungen mit starrem Layout stellt es ebenfalls keine große Herausforderung dar, solche Tooltipps zu integrieren; dafür existieren etliche Programmiergerüste. In unserem Fall finden diese jedoch keine Anwendung, da die PE-Benutzerschnittstelle sehr dynamisch ist: der Benutzer startet typischerweise in einem Zustand mit zwei Ergebnismengen – den Veranstaltungen und Orten einer bestimmten Stadt – in dem nicht alle Funktionen von PE anwendbar oder möglich sind. Außerdem existieren viele denkbare Folgezustände, wie der Benutzer den Explorationsbaum weiter aufbauen könnte, sodass es nicht ganz einfach ist, im Vornhinein die anzuzeigenden Tooltipps zu definieren. Eine weitere Herausforderung ist, dass die Position von zu erklärenden UI-Elementen schlechter vorhersehbar ist als bei typischen Webseiten, für die die typischen Programmiergerüste der Tooltipps gedacht sind.

In diesem Abschnitt beschreibe ich, wie ich auf diese Herausforderungen eingegangen bin. Meine Lösungsansätze könnten auch auf andere webbasierte Systeme mit ähnlichen Eigenschaften Anwendung finden.

4.8.1 Welche Ergebnisse zur Bedienbarkeit und zur Nützlichkeit lagen dem Design zugrunde?

Das Vorgehen bei dem Entwurf der Tooltipps war iterativer Natur: in ersten Versuch zeigte ein – von dem hypothetischen Benutzer Alice – aufgebauter Explorationsbaum mithilfe von einer Sequenz von Tooltipps, welche Schritte Alice machen musste um das Resultat zu erhalten. Informelle Tests mit 3CIXTY-Teilnehmern zeigten, dass

diese Art von Rückblick nicht befriedigend ist — der Benutzer möchte wissen, was er jetzt tun kann und nicht wie der Explorationsbaum, den er sieht, entstanden ist. Es ist jedoch nicht einfach mit Tooltipps zu zeigen, wie ein Explorationsbaum aus dem Nichts aufgebaut wird, da der Benutzer viele Möglichkeiten dazu hat und viele Sonderfälle abgedeckt werden müssen.

Nach Einführung der im Abschnitt 4.7 behandelten QuickStart-Applikationen wurde in einem weiteren Iterationsschritt auf dieses Problem eingegangen. Öffnet der Benutzer ein Explorationsbaum aus einer QuickStart-Applikation, so ist davon auszugehen, dass er die vorliegende Antwort wissen, verstehen und möglicherweise auch weiter ausbauen möchte. An dieser Stelle kann man verfügbare Funktionen erläutern und ist auch nicht auf Tooltipps zu Knoten, die zuerst erstellt werden müssen, angewiesen. Weitere informelle, iterative Tests haben zu Detailverbesserungen geführt, die ich in den kommenden Abschnitten nennen werde.

4.8.2 Wie sieht die gewählte Lösung für den Benutzer im Detail aus?

Die Anzeige von Tooltipps kommt, wie auf Abbildung 4.10 zu sehen ist, als eine Hintereinanderreihung mehrerer Tooltipps nach dem Öffnen eines QuickStart-Lesezeichens, das mit solchen versehen wurde. Die Tooltipps zeigen, wie die Antwort auf die QuickStart-Frage verstanden werden kann und gibt auch Hinweise darüber, wie der Benutzer die Ergebnisse weiter erkunden kann — beispielsweise durch die Anwendung eines Filters oder von multiplem Pivoting.

Der Benutzer hat auch die Möglichkeit, sich zu jedem beliebigen Zeitpunkt Tooltipps anzeigen zu lassen. Er erhält nach einem Klick auf die Fragezeichen-Schaltfläche in der Kopfzeile eine Tooltippsequenz der Funktionen, die für ihn gerade im sichtbaren Bereich sind.

4.8.3 Was sind die grundlegenden UI-Designentscheidungen?

Eine UI-Designentscheidung war die Option, alle Tooltipps gleichzeitig anzuzeigen. Diese Funktion könnte mit einem global erreichbaren Knopf ein- und ausgeschaltet werden, sodass der Benutzer eine schnelle Übersicht der möglichen Handlungen erhalten kann. Das Haupthindernis bei diesem Ansatz wären die Fälle, bei denen sich einzelne Tooltipps überlagern — diese müssten entweder durch den Ersteller der QuickStart-Applikation vermieden werden oder das System müsste diese Fälle durch eine geschickte Anordnung umgehen. Da eine praktikable Lösung für dieses

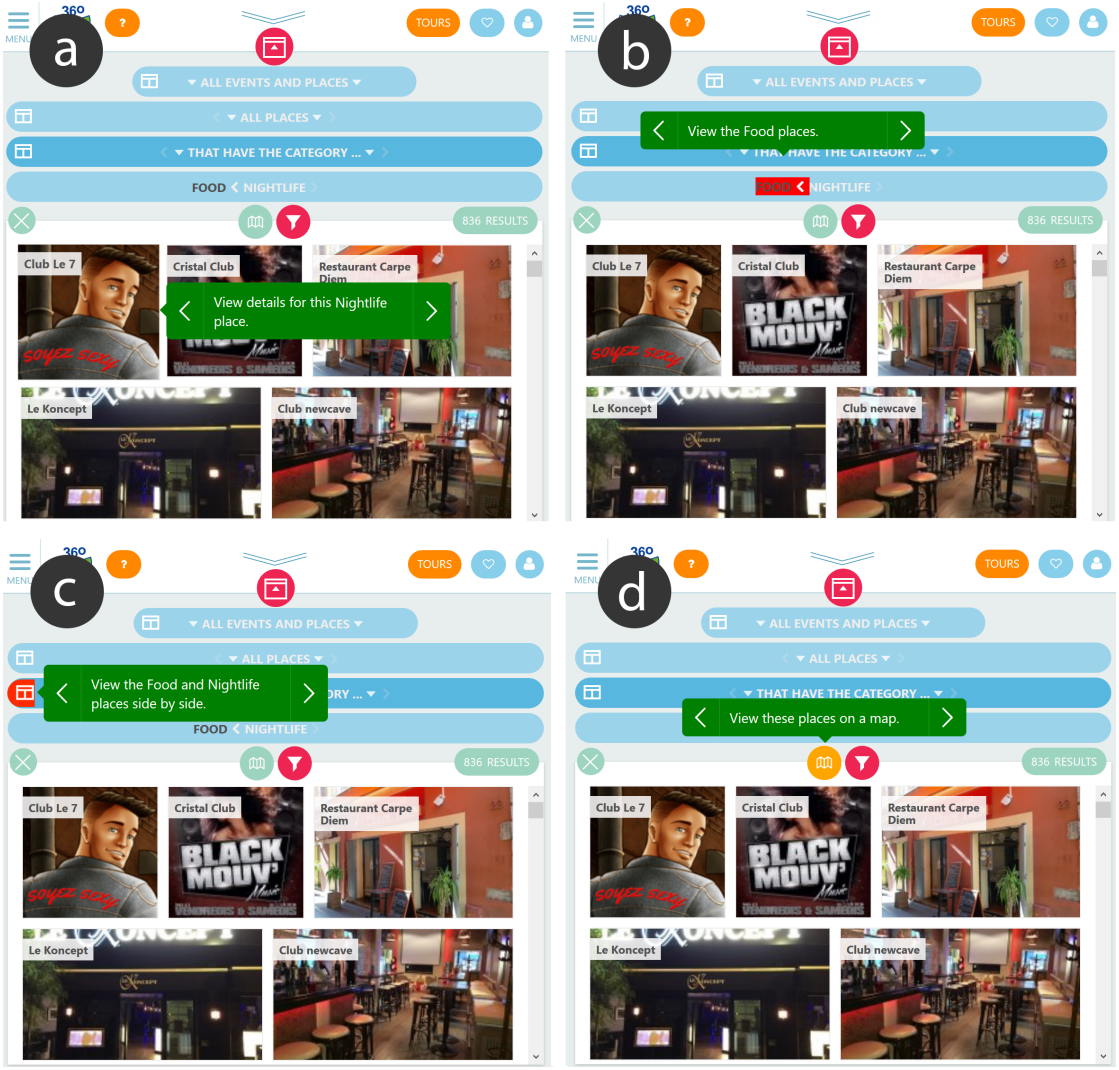

Abbildung 4.10: Eine Sequenz von drei Tooltipps erläutert gängige Aktionen

Problem nicht in der vorgegebenen Zeit gefunden werden konnte, wurde dieser Ansatz in dieser Arbeit nicht weiterverfolgt. Allerdings wurde die dahinterliegende Grundidee aufgenommen und auf der Hilfeseite von ExplorMI 360 eingebaut. Auf dieser sind die gängigsten Funktionen mit Tooltipps beschriftet, sodass der Benutzer auf einen Blick eine Erklärung zu jedem einzelnen bekommen kann. In diesem Spezialfall konnte ich mithilfe einer vorausgesuchten Positionierung der Tooltipps dafür sorgen, dass diese sich nicht überlappen und die bereits erwähnten Komplikationen beim Anzeigen mehrerer Tooltipps einfach umgegangen werden.

Rückmeldungen von Benutzern – und von ErgoSign – zeigten, dass auch auf der Hilfeseite die Anzahl der Tooltipps nicht zu groß werden sollte, da die Seite dann überfüllt wirken könnte. Eine Lösung wäre die Aufteilung der Tooltipps in mehrere Gruppen, die auf Knopfdruck nacheinander dargestellt werden. Die Alternative wäre, eine sequenzielle Darstellung auch auf der Hilfeseite zu verwenden, mit dem Nachteil, dass kein schneller Überblick der Möglichkeiten gegeben ist. Das Fazit, das ich hieraus schließen konnte, ist, dass man die Anzahl der Tooltipps in Grenzen halten muss. Dieses wird durch das neue Design möglich, da weniger Erklärungen erforderlich sind.

Im Abschnitt 3 wurde bereits auf Erweiterungen eingegangen, die die Benutzerschnittstelle intuitiver machen; dadurch sind weniger Anweisungen für das Verständnis notwendig geworden. Ein anschließender Benutzertest im kleinen Rahmen auf „usertesting.com“ nach der Implementierung des erhaltenen Feedbacks ergab, dass die Tooltipps von zwei von drei Benutzern, die die Benutzerschnittstelle vorher nicht kannten, insgesamt als nützlich empfunden wurden. So schrieb einer dieser beiden Personen, auf die Frage hin, ob die Tooltipps dabei geholfen haben, die Benutzerschnittstelle besser zu verstehen, folgendes:

> „Yes, a lot! It shows things in a detailed and very easy way.“

Die dritte Person, die weniger Schwierigkeiten dabei hatte die Benutzerschnittstelle zu bedienen, empfand die Tooltipps als nicht zwingend notwendig, was in Bezug auf die Tooltipps zumindest neutral betrachtet werden kann.

4.8.4 Welche Möglichkeiten gibt es, diese Entscheidungen technisch umzusetzen; und was sind deren Vor- und Nachteile?

Bezüglich der Umsetzung gibt es drei Alternativen:

(a) ein bestehendes Framework für diese Aufgabe zu verwenden

(b) ein eigenes Framework zu schreiben

(c) eine Sequenz vorgefertigter Bilder zur Verfügung stellen, die erklärende Tooltipps beinhalten

Obwohl die erste Lösung im Vergleich zur Zweiten eine große Zeitersparnis darstellen könnte, stellte sich im Verlauf der Implementierung heraus, dass die angebotenen Funktionen des verwendeten Frameworks größere Beschränkungen in Bezug auf

[4] `http://zurb.com/playground/jquery-joyride-feature-tour-plugin`

Parallele Exploration aufwiesen. Diese waren hierfür nicht geeignet, da sie für konventionelle Webseiten ausgelegt sind und mit dynamisch nachgeladenen Inhalten nicht zurechtkommen. Im Speziellen wurde das Framework Joyride[4] verwendet: Tooltipps zu skalierten Elementen werden an falschen Stellen angezeigt. Die Variante (b) ermöglicht schließlich genau das, was die PE-Benutzerschnittstelle erfordert, mit dem Nachteil, dass eine eigene Implementierung mehr Zeit in Anspruch nimmt. Die Möglichkeit (c) ist aus der Sicht der Programmierung die Einfachste — aber auch die mit den meisten Einschränkungen. Hierzu müssen nur Bildschirmabzüge erstellt und mit entsprechenden Tooltipps versehen werden.

Bei jeglichen Änderungen der Benutzerschnittstelle müssten stets alle Bildschirmabzüge neu erstellt werden, was diese Lösung disqualifiziert. Textuelle Inhalte der Tooltipps lassen sich im Übrigen auch durch den „Google Translator“ übersetzen, was ein Vorteil von (a) und (b) gegenüber (c) darstellt. Ich habe mich daher für Option (b) entschieden, da die Flexibilität dieser Lösung in jeder beschriebenen Hinsicht vorzuziehen ist.

4.8.5 Wie wurde es implementiert?

Das Framework zum Anzeigen der Tooltipps funktioniert wie folgt: alle Tooltipps werden am Anfang geladen, mithilfe von CSS werden alle Tooltipps bis auf das Erste ausgeblendet. Navigiert der Benutzer in den Tooltipps vorwärts, wird das Aktuelle aus- und das nächste eingeblendet. Dies ermöglicht auch eine einfache Implementierung eines „Zurück“ Knopfes, da die Tooltipps weiterhin vorhanden, aber versteckt sind. In der ersten Version der Implementierung von Tooltipps wurden einzelne Bookmarks mithilfe einer Konfigurationsdatei angepasst: je nach Lesezeichen wird entschieden, welche Tooltipps angezeigt werden und auf welchen Knoten diese gerichtet sind.[5] So beschreibt, wie im Codeausschnitt 4.2 zu sehen ist, der Befehl „Tooltip.create()“, der zwei Parameter beinhaltet, zuerst einen CSS-Pfad, wo der Tooltipp im HTML-Dokument erscheinen soll. Der zweite Parameter enthält den Text des Tooltipps, der auch HTML-Notationen zulässt.

Listing 4.2: Beispiel einer Konfiguration eines Tooltipps

```
1 Tooltipp.create(
2   $('div[data-nodeid=11]>.main>.body'),
3   'Click on any item to see its details.'
4 );
```

[5] In einer späteren Version wurde vom 3CIXTY-Unterauftragnehmer Deip Designs die Konfigurationsdatei obsolet, da die unterschiedlichen Tooltipps direkt in MySQL gespeichert wurden.

5 Schlussfolgerungen und Möglichkeiten für weitere Forschung

5.1 Rückblick auf die geleisteten Beiträge

Wie wir in dieser Arbeit gesehen haben, bietet Parallele Exploration den Benutzern Vorteile gegenüber traditionellen Suchsystemen, aber es mussten einige Herausforderungen bewältigt werden: durch die Annäherung an die facettierte Suche wurde den Benutzern ein leichter Einstieg mit bekannten Interaktionen gegeben, wobei Möglichkeiten der parallelen Erkundung erst schrittweise eingeführt wurden. Manuelles und automatisches Zusammenklappen einzelner Knoten hilft einerseits Benutzern, den Überblick zu behalten und ermöglicht gleichzeitig, die Benutzerschnittstelle auf Endgeräten jeglicher Bildschirmauflösung anzuzeigen und zu bedienen.

Es wurden neue Funktionalitäten hinzugefügt, die auf unterschiedliche Arten gut mit dem Interaktionsparadigma PE zusammenwirken. Wie wir gesehen haben, kann Pivoting für Benutzer in vielen Situationen nützlich sein; in Zusammenhang mit PE entsteht jedoch eine Synergie: Benutzer können von mehreren Ausgangspunkten aus, gleichzeitig mehrere Pfade verfolgen und verwandte Elemente erkunden — im Gegensatz dazu müssten Backtracking-Strategien angewandt werden, wenn dem Benutzer nur eine Ergebnismenge angezeigt würde.

Die Funktionalität, einen Zustand oder eine Ergebnismenge zu speichern, ist auch in anderen Suchsystemen anzutreffen. In Hinblick auf PE entsteht der zusätzliche Vorteil, dass der in Abschnitt 4.5 besprochene Ansatz von Lesezeichen nicht nur einen Zustand, sondern die gesamte Erkundung in Form eines Explorationsbaums erfasst. Diese Tatsache ermöglicht einem Kollaborationspartner einen Überblick der Suchstruktur und bietet dadurch mehr Hintergrundinformationen zur Suche. Darüber hinaus ging diese Arbeit auf UI-Elemente ein, die zwar in anderen Systemen gängig sind, bei denen aber Wege gefunden werden mussten, diese auf PE zu übertragen. Wie in Abschnitt 4.1 und 4.8 besprochen, mussten Entscheidungen

beim Entwurf und bei der Implementierung der Volltextsuche und der Tooltipps getroffen werden.

Die Beiträge dieser Arbeit dienen als Grundlage für weitere Entwicklungen, bieten die Kernfunktionalität und ermöglichen notwendige Anpassungen und Erweiterungen. Sie wurden im Rahmen von ExplorMI 360 verwendet, welches den ersten Platz bei der „Semantic Web Challenge 2015“ [25] gewann.[1] Dessen Benutzerschnittstelle für PE, wurde für den „CeBIT Innovation Award 2016“ eingereicht und kam unter die sechs Finalisten. Die aktuelle Implementierung, die in dieser Masterarbeit besprochen wurde, hat Interesse bei kommerziellen Partnern in Europa geweckt, mit denen eine Zusammenarbeit angestrebt wird — insbesondere stellt sie eine Kernkomponente vom dem Produkt 3cixty dar, die von der französischen Firma Data-Moove angeboten wird.

5.2 Möglichkeiten für weitere Forschung

Ich habe im Rahmen dieser Arbeit die Parallele Exploration anhand von ExplorMI 360 vorgestellt, es existieren jedoch viele Anwendungsgebiete auf unterschiedlichen Domänen, die gerade mit betreffenden Unternehmen im Gespräch sind: zwei große Messegesellschaften, die Medienbibliothek eines Fernsehsenders und Webshops eines Unternehmens, das Software für den elektronischen Handel verkauft. Neben den notwendigen Anpassungen für jede dieser Domänen, die gemacht werden müssen, könnten weitere Herausforderungen für die Parallele Exploration entstehen.

Gespräche mit zwei dieser Unternehmen offenbarten die Möglichkeit der Integration eines Empfehlungssystems in eine PE-Applikation, um die angezeigten Ergebnisse an die jeweiligen Benutzerwünsche anzupassen. Es kamen Ideen auf, wie Empfehlungstechnologien effektiver in Zusammenhang mit PE verwendet werden könnten. Einerseits könnte die Sortierung einer Ergebnisliste darauf beruhen, dass die meistempfohlenen Ergebnisse zuerst erscheinen. Andererseits wären unterschiedliche Filtermöglichkeiten denkbar, die von dem Empfehlungssystem Gebrauch machen, beispielsweise die Anzeige derjenigen Elemente einer Liste, die einem Benutzer empfohlen werden. Dies könnte etwa gleichzeitig Elemente darstellen, bei denen das System sicher ist, dass der Benutzer sie mögen wird, und Elemente, die Gelegenheiten bieten, neue Sachen auszuprobieren. Solche Filter können sich aber auch auf eine Personengruppe beziehen [15] und die Elemente selektieren, die einzelnen Gruppenmitgliedern oder der gesamten Gruppe empfohlen werden.

[1] `http://challenge.semanticweb.org/2015/`

Obwohl die zu Pivoting relevanten Beispiele, die in dieser Arbeit gezeigt wurden, nur auf räumliche Beziehungen eingingen – beispielsweise Veranstaltungen in der Nähe eines Ortes – ist die Anwendbarkeit von Pivoting nicht darauf beschränkt. Die Art der Pivotingrelation kann dabei beliebig sein, so könnte man zum Beispiel in dem Kontext einer Messe von einer Menge von Ausstellern, zu den von ihnen angebotenen Produkten oder den Vorträgen, die sie halten, wechseln.

Eine weitere mögliche Anwendung der PE neben bestehender Filter- und Pivoting-operationen könnte „Critiquing“ darstellen [22], welches oft mit Empfehlungssystemen verwendet wird. Eine Integration in PE würde eine Art „paralleles Critiquing“ ermöglichen, wodurch der Benutzer die Möglichkeit hätte, mehrere Variationen gleichzeitig zu vergleichen. Critiquing kann in diesem Zusammenhang auf zwei Arten erfolgen: es kann sich auf bestehende Entitäten beziehen, wie beispielsweise bei einer Anfrage nach einem günstigeren Hotel oder einem mit größeren Zimmern. Alternativ kann es auch für die Erzeugung neuer Instanzen dienen: ausgehend von einem Kochrezept können Variationen gebildet werden, bei dem gewisse Zutaten durch andere ausgetauscht werden. So ist es denkbar aus einem Fleischgericht eine vegane Variation zu bilden, bei der das Fleisch durch Tofu ersetzt, und auf gewisse andere Zutaten verzichtet würde und gleichzeitig andere Variationen des Rezepts anzuzeigen.

Die aktuelle Implementierung von PE erlaubt es, mehrere Explorationspfade zu verfolgen und diese immer weiter aufzuspalten. Man könnte es ermöglichen, Pfade wieder zusammenzuführen und somit beispielsweise ein neues Ergebnisfenster mit der Vereinigungsmenge von zwei Knoten zu bilden, von dem man aus weiter erkunden kann. Ein anderer interessanter Aspekt der Zusammenführung mehrerer Explorationspfade wäre das Verschmelzen von Antworten mehrerer Fragen der QuickStart-Applikationen. Die Anzeige der beiden Fragen „Welche interessanten Konzerte finden am Wochenende statt?“ und „Welche Festivals laufen aktuell?“ in einem einzigen Explorationsbaum anzeigen.

Zusammengefasst kann man sagen, dass die dargestellten Ergebnisse dieser Arbeit nicht nur als Lösung eines bestimmten Problems angesehen werden können, sondern auch als ein aus mehreren Einzelteilen bestehendes Programmiergerüst für die Erstellung einer Vielzahl von Anwendungen der Parallelen Exploration.

Literaturverzeichnis

[1] Catalin Barbu, Anthony Jameson, and Adrian Spirescu. An exploration interface for choosing in-car applications. *Deliverable A1308-TD1302 of EIT ICT Labs Activity 13131, Apps for Your Car*, 2013.

[2] Max H. Bazerman, Don A. Moore, Ann E. Tenbrunsel, Kimberly A. Wade-Benzoni, and Sally Blount. Explaining how preferences change across joint versus separate evaluation. *Journal of Economic Behavior & Organization*, 39:41–58, 1999.

[3] Marc Bron, Jasmijn van Gorp, Frank Nack, Maarten de Rijke, Andrei Vishneusk, and Sonja de Leeuw. A subjunctive exploratory search interface to support media studies researchers. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 425–434, New York, 2012. ACM.

[4] Sven Buschbeck, Anthony Jameson, and Tanja Schneeberger. New forms of interaction with hierarchically structured events. In Marieke van Erp, Willem R. van Hage, Laura Hollink, Anthony Jameson, and Raphaël Troncy, editors, *Proceedings of the ISWC 2011 Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web*. CEUR, Aachen, Germany, 2011. Available from http://ceur-ws.org/Vol-779/.

[5] Sven Buschbeck, Anthony Jameson, Tanja Schneeberger, and Robin Woll. A web-based user interface for interaction with hierarchically structured events. In *Demo paper for the 2012 Conference on Intelligent User Interfaces*, 2012.

[6] Sven Buschbeck, Anthony Jameson, Adrian Spirescu, Tanja Schneeberger, Raphaël Troncy, Houda Khrouf, Osma Suominen, and Eero Hyvönen. Parallel faceted browsing. In *Extended Abstracts of CHI 2013, the Conference on Human Factors in Computing Systems (Interactivity Track)*, 2013.

[7] Sven Buschbeck, Anthony Jameson, Raphaël Troncy, Houda Khrouf, Osma Suominen, and Adrian Spirescu. A demonstrator for parallel faceted browsing. In *Proceedings of the EKAW 2012 Workshop on Intelligent Exploration of Semantic Data*, Galway, Ireland, 2012.

[8] Samur F. C. De Araujo and Daniel Schwabe. Explorator: A tool for exploring RDF data through direct manipulation. In *Proceedings of the WWW 2009 Workshop on Linked Data on the Web*, 2009.

[9] Ward Edwards and B. Fasolo. Decision technology. *Annual Review of Psychology*, 52:581–606, 2001.

[10] Marti Hearst. Next generation web search: Setting our sites. *IEEE Data Engineering Bulletin*, 23, 2000.

[11] Marti A. Hearst. *Search User Interfaces*. Cambridge University Press, Cambridge, UK, 2009.

[12] F. David Huynh and R. David Karger. Parallax and Companion: Set-based browsing for the data web. In *Proceedings of the 18th International World Wide Web Conference*, 2009.

[13] Anthony Jameson, Bettina Berendt, Silvia Gabrielli, Cristina Gena, Federica Cena, Fabiana Vernero, and Katharina Reinecke. Choice architecture for human-computer interaction. *Foundations and Trends in Human-Computer Interaction*, 7(1-2):1–235, 2014.

[14] Anthony Jameson and Sven Buschbeck. Interaction design for the exchange of media organized in terms of complex events. In *Proceedings of the Workshop EVENTS 2010 – Recognising and Tracking Events on the Web and in Real Life, hosted by SETN 2010*, Athens, 2010.

[15] Anthony Jameson and Barry Smyth. Recommendation to groups. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 596–627. Springer, Berlin, 2007.

[16] Anthony Jameson, Adrian Spirescu, Tanja Schneeberger, Edit Kapcari, and Sven Buschbeck. Learnability and perceived benefits of parallel faceted browsing: Two user studies. In *Proceedings of the Hypertext 2013 Workshop on Intelligent Exploration of Semantic Data*, Paris, 2013.

[17] Waqas Javed, Sohaib Ghani, and Niklas Elmqvist. PolyZoom: Multiscale and multifocus exploration in 2D visual spaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, 2012.

[18] Edit Kapcari. Parallel vs¸ traditional faceted browsing: Comparative studies and proposed enhancements, 2015. Masterarbeit, Universität des Saarlandes, Fachrichtung Informatik.

[19] Dominik Kuhn. Anfragen für die Parallele Exploration - Ein Programmiergerüst für die Erzeugung inhaltlich verwandter Anfragen an Wissensbasen und Datenbanken, 2016. Masterarbeit, Universität des Saarlandes, Fachrichtung Informatik.

[20] Aran Lunzer and Kasper Hornbæk. Subjunctive interfaces: Extending applications to support parallel setup, viewing and control of alternative scenarios. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 14(4):17, 2008.

[21] I. S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7:91–139, 1992.

[22] Lorraine McGinty and James Reilly. On the evolution of critiquing recommenders. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 419–453. Springer, Berlin, 2011.

[23] Theresa Neil. *Mobile Design Pattern Gallery: UI Patterns for Mobile Applications, Second Edition*. O'Reilly Media, Inc., 2014.

[24] Jan Polowinski. *Widgets for Faceted Browsing*, pages 601–610. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[25] Giuseppe Rizzo, Raphaël Troncy, Oscar Corcho, Anthony Jameson, Julien Plu, Juan Carlos Ballesteros Hermida, and Ahmad Assaf. The 3cixty knowledge base for Expo Milano 2015: Enabling visitors to explore the city. In *Proceedings of the 8th International Conference on Knowledge Capture*, K-CAP 2015, pages 18:1–18:4, New York, NY, USA, 2015. ACM.

[26] Adrian Spirescu. Parallel Faceted Browsing - Entwurf, Implementierung und Evaluation, 2013. Bachelorarbeit, Universität des Saarlandes, Fachrichtung Informatik.

[27] Daniel Tunkelang. *Faceted Search*. Morgan & Claypool, Palo Alto, CA, 2009.

[28] Robert Villa, Iván Cantador, Hideo Joho, and Joemon M. Jose. An aspectual interface for supporting complex search tasks. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 379–386, New York, 2009. ACM.

[29] Ryen W. White and Resa A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*. Morgan & Claypool, San Francisco, CA, 2009.

[30] Wenchang Xu, Chun Yu, Songmin Zhao, Jie Liu, and Yuanchun Shi. Facilitating parallel web browsing through multiple-page view. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2167–2170, New York, NY, USA, 2013. ACM.

[31] Jian Zhao, Daniel Wigdor, and Ravin Balakrishnan. TrailMap: Facilitating information seeking in a multi-scale digital map via implicit bookmarking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3009–3018, New York, 2013. ACM.

Abbildungsverzeichnis

Liste der Algorithmen