Herausforderungen der kombinierten Verwendung von On-Board- und Off-Board-Sprachdialogsystemen in Telematikeinheiten im Automobil

Maria Schmidt[1,2], *Steffen Werner*[2], *Tobias Heinroth*[2]

[1]*Universität Tübingen,* [2]*Daimler AG*
maria2.schmidt@student.uni-tuebingen.de
{*steffen.s.werner, tobias.heinroth*}*@daimler.com*

Kurzfassung: Die Erwartungen von Autofahrern an ein integriertes Infotainment-System im Fahrzeug nehmen stetig zu. Neben klassischen Funktionen wie Radio, Telefon und Navigation werden auch Inhalte aus dem Internet sowie die Vernetzung mit Social Media-Angeboten erwartet. Apps auf mobilen Endgeräten unterstützen diesen Trend, weshalb zunehmend nicht mehr nur Endgeräte an sich, sondern auch einzeln downloadbare Apps integriert werden. Eine Bedienung per Sprache ist dabei nicht nur komfortabel, sondern auch sicherer und weniger ablenkend als andere Bedienmodalitäten. In diesem Beitrag werden Herausforderungen diskutiert, welche sich aus der kombinierten Verwendung von Sprachdialogsystemen (SDS) ergeben, die sich einerseits lokal im Fahrzeug und andererseits off-board (z. B. auf einem Mobilgerät) befinden. Die Themen „Intelligentes Lernen: das On-Board-SDS lernt vom Off-Board-SDS“ und „Natürlichsprachlichkeit“ werden herausgegriffen und am Beispiel der Verkürzung von Sprachausgaben näher diskutiert. Des Weiteren stellen wir ein entsprechendes Implementierungskonzept für dieses Beispiel vor. Abschließende Ergebnisse sowie eine entsprechende Evaluation folgen in weiterführenden Arbeiten.

1 Einleitung

Die Erwartungen an Infotainment-Systeme im KFZ steigen, da viele Menschen häufig Zeit im Auto verbringen (vgl. [3]) und das Internet fast überall zur Verfügung steht. Diese Systeme sollten intuitiv und per Sprache bedienbar sein, was sowohl die Ablenkung des Fahrers gering hält als auch den Bedienkomfort erhöht. Infotainment-Systeme (auch Telematik-Einheiten oder Head Units genannt) müssen auch in der Lage sein, online verfügbare Informationen zu verarbeiten sowie dem Nutzer seine alltäglich verwendeten Apps im Auto zur Verfügung zu stellen. Beinhalten diese Apps bereits eine Sprachbedienung, soll auch diese dem Fahrer zur Verfügung stehen. Gleichzeitig müssen aber die damit verbundenen Implikationen für das bestehende On-Board-Sprachbediensystem berücksichtigt werden.

In dieser Arbeit werden Herausforderungen der kombinierten Verwendung von beiden Systemen ausführlich diskutiert sowie verschiedene Teilaspekte erörtert. Wir unterscheiden terminologisch wie auch analytisch aufgrund der örtlichen Implementierung des verwendeten SDS zwischen *On-Board*, *Off-Board* und der *Kombination von On- und Off-Board*. *On-Board-SDS* steht für das interne Sprachdialogsystem (SDS) der Telematik-Einheit, *Off-Board-SDS* steht für externe, unabhängige SDS (z. B. in Apps auf dem Smartphone). Die Kombination von *On-Board*- und *Off-Board*-Systemen bezeichnet dementsprechend die Interaktion des internen SDS mit externen SDS.

Das Ziel dieser und weiterführender Arbeiten ist eine intelligente, möglichst stimmige Verknüpfung von externen SDS mit dem internen SDS sowie eine Bedienung, die natürlichsprach-

liche Aspekte umsetzt. Einige der sich daraus ergebenden Aufgaben werden in Abschnitt 3 diskutiert. Die für die Arbeit herausgegriffene Umsetzung der automatischen Verkürzung von Sprachausgaben durch das Lernen des internen SDS von externen ist in Abschnitt 4 beschrieben.

2 Aktueller Status

In diesem Abschnitt betrachten wir zuerst den aktuellen Status der zugrunde liegenden Telematik-Einheit inkl. ihres SDS. Im Anschluss werden die Herausforderungen, welche sich aus der kombinierten Verwendung von On-Board- und Off-Board-SDS ergeben, diskutiert.

2.1 Telematik-Einheit

In den Telematik-Einheiten ist die Sprachbedienung als eine mögliche Bedienmodalität *„on board“* integriert. Dies erfordert eine embedded Text-to-Speech Engine sowie eine Automatische Spracherkennung, welche es dem Benutzer erlaubt, spezifisches Vokabular zu sprechen. Wie in Abb. 1 dargestellt, ist die Telematik-Einheit primär in der Lage, die internen Module wie `Navigation`, `Radio` etc. nach Eingabe von gesprochenen Benutzer-Kommandos zu steuern. In dieser Illustration werden nur die Verbindungen vom SDS zum jeweiligen Modul gezeigt, die Verbindungen der Module untereinander sind außen vor gelassen.

Speziell entwickelte Apps erweitern den Funktionsumfang der Telematik-Einheit und sind entweder „on board“ fest integriert oder lassen sich herunterladen. Auch diese Apps sind heutzutage sprachbedienbar (z. B. Google Voice Search auf Android Smartphones). In Grafik 1 sind diese als internes Modul (`Apps`) dargestellt. Auch ist es denkbar, externe, „fremde“ Apps, die sich beispielsweise auf dem Smartphone des Users – also *„off board“* – befinden, mit dem On-Board-System zu verbinden. Dadurch ist die haptische und sprachliche Steuerung und somit die Benutzung dieser Apps in Verknüpfung mit dem internen System möglich. Da immer mehr Smartphones und die entsprechenden Apps auf den Markt drängen und deren Entwicklungszyklen wesentlich kürzer sind als im Automobilbereich, entstehen zwangsläufig Herausforderungen durch eine kombinierte Verwendung von *On-Board-* und *Off-Board-SDS*. Diese werden in Abschnitt 3 diskutiert.

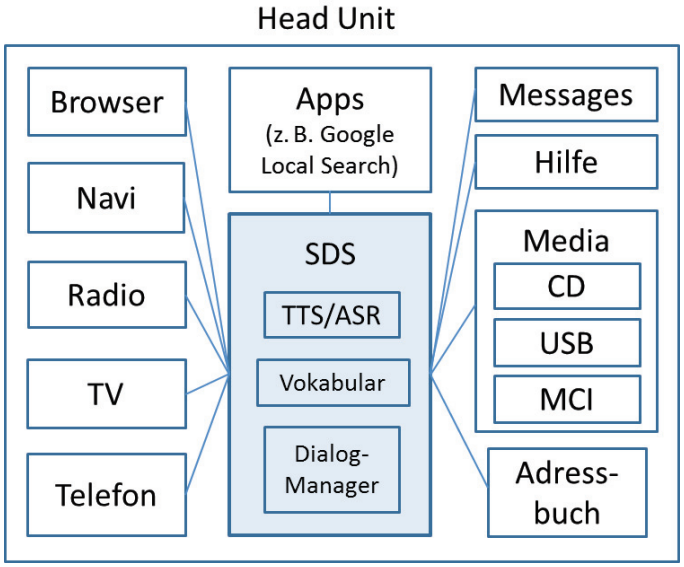

Abbildung 1 - Das zugrunde liegende Telematik-System im Automobil.

2.2 Beschreibung von Sprachdialogen

Um diese Herausforderungen angehen zu können, muss zuerst eine einheitliche Beschreibung des Dialogablaufs gefunden werden. Abbildung 2 ist ein schematischer, beispielhafter Dialogablauf, der sich aus unterschiedlichen Elementen zusammensetzt. Zum Einen hat jeder Dialog einen *Start State* als Anfangspunkt sowie einen *Final State* als Endpunkt ähnlich einem endlichen Zustandsautomaten. Wir nehmen an, dass der Benutzer aus einem anderen Dialog in den dargestellten „hereinspringt“ und bereits eine Adresse zur Zielführung eingegeben hat. Nun fragt das System explizit in der *System-Sprach-Ausgabe* (hier als Sprechblase dargestellt) nach „Ist die folgende Adresse korrekt?“. Daraufhin folgt eine *Benutzer-Sprach-Eingabe* „Ja!“ bzw. „Nein!“ (symbolisiert durch eine Ellipse), aufgrund derer ein *Entscheider* (Raute) den Zweig des weiteren Dialogablaufs bestimmt. Wenn der Benutzer die Adresse als richtig bestätigt hat, wird mit Hilfe einer weiteren *System-Ausgabe* mitgeteilt, dass die Zielführung nun gestartet wird. Anschließend wir die entsprechende Funktion im Modul `Navi` aufgerufen (dargestellt als Parallelogram). Wenn der Benutzer hingegen die Adresse ablehnt, wird ein *anderer Dialog*, z. B. ein Korrektur-Dialog zur Änderung der Adressangaben, aufgerufen (Rechteck). Hat er den abgebildeten Dialog vollständig bis zu dessen *Final State* durchlaufen, kehrt der Ablauf zum bisherigen Dialog-Schritt zurück, das System meldet, dass die Adresse erfolgreich geändert worden sei, und ruft wie im ersten Fall die entsprechende Funktion im `Navi`-Modul auf.

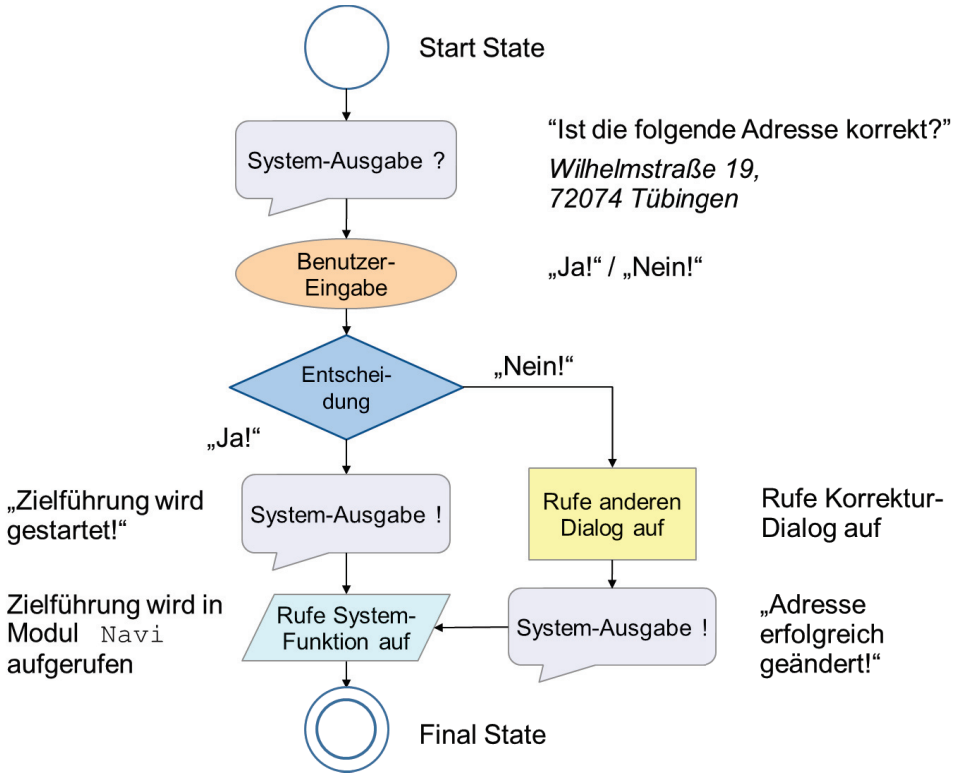

Abbildung 2 - Beispielhafter Ablauf eines Navigationsdialogs.

Wir nehmen den in Abb. 2 gezeigten Dialogablauf als allgemein gültig an, d. h. sowohl das On-Board-SDS als auch die Off-Board-SDS können so dargestellt bzw. modelliert werden. Infolgedessen ist eine kombinierte Verwendung der Systeme theoretisch auch sehr fein abgestimmt möglich, z. B. indem nach einem abgeschlossenen On-Board-Dialog – oder sogar bereits nach einem Entscheidungselement – ein entsprechender Off-Board-Dialog(-Pfad) verwendet wird.

3 Herausforderungen

Um die in Abschnitt 1 genannten Ziele zu erreichen, müssen die folgenden Punkte adressiert werden. Wir beschreiben diese Herausforderungen separat, auch wenn sie ineinander greifen.

3.1 Handhabung von Start und Ende der SDS-Sessions sowie von kritischen Timing-Situationen

Um eine reibungslose, kombinierte Benutzung von unterschiedlichen SDS zu gewährleisten, muss festgelegt werden, wo eine Dialogsession beginnt und wo sie endet. Darüber hinaus muss der Wechsel vom einen zum anderen SDS klar definiert sein. Ein Beispiel hierfür ist der Applikationswechsel (z. B. vom On-Board-Radio zur Off-Board-Navigation). Auch kritische Timing-Situationen, wie Signalabbrüche in sog. Funklöchern, müssen während der Verwendung von Off-Board-Dialogen berücksichtigt werden, bspw. bei der Übertragung des Sprachsignals zum externen Erkenner.

3.2 Vokabulardisambiguierung

Durch die steigende Anzahl von Apps wird die Möglichkeit erhöht, dass ambiges oder zum internen SDS konkurrierendes Vokabular vorhanden ist. Bringt der Benutzer beispielsweise eine Navigationsapp mit, die genau wie die intern vorhandene Anwendung heißt, so besteht Konfliktpotenzial, wenn der Fahrer „Navigation“ sagt. – Welche der beiden Anwendungen soll geöffnet werden? Über den Applikationsnamen hinaus kann auch das dialogspezifische Vokabular mehrdeutig sein. Wenn globales Vokabular, das der Benutzer quasi zu jedem Zeitpunkt sprechen kann, ambig ist, entsteht das gleiche Problem wie bei mehrdeutigen Applikationsnamen.

3.3 Verbesserung der Natürlichsprachlichkeit

Heutige SDS sind i. d. R. performant und erfüllen den vorgegebenen Zweck. Häufig sind diese Systeme eher Nutzer getrieben ausgelegt, d. h. der Nutzer initiiert den Dialog selbstständig und nicht das System. Um dem Benutzer eine intuitivere Bedienung zu ermöglichen, sollten auch Aspekte der Natürlichsprachlichkeit – wie eine gemischte Dialog-Initiative, Adaptivität oder Antwortüberladung (nach Berg [1] und Berg et al. [2]) – umgesetzt werden. Bei der Kombination von mehreren SDS müssen aber auch unterschiedliche oder sogar widersprüchliche Dialogeigenschaften beachtet werden. Somit könnte ein SDS mit gemischter Dialoginitiative auf ein System mit Nutzerinitiative treffen. Eine rein technische Kombination ist sicherlich machbar, allerdings ist der Einfluss auf die User Experience nicht abschätzbar.

3.4 Intelligentes Lernen im On-Board-Sprachdialogsystem

Um ein intelligentes, gesamtheitliches Bedienkonzept zu erzielen, obwohl ein heterogenes System aus On-Board-SDS und Off-Board-SDS zugrunde liegt, muss eine Anpassung beider Systeme erfolgen. Im Rahmen dieser Arbeit soll sich das On-Board-SDS (ohne Beschränkungen der Allgemeingültigkeit) durch Lernen von Eigenschaften an die Off-Board-SDS anpassen, da die Off-Board-SDS durch ihre meist unabhängige Entwicklung nur mäßig bis gar nicht kontrollierbar sind. Das in Abbildung 3 schematisch dargestellte Konzept zeigt unseren Ansatz, das „Intelligente Lernen“: Das On-Board-SDS lernt bspw. Vokabular, Erkennungsergebnisse oder Entscheidungen im Dialog von den Off-Board-SDS. Aufgrund derer passt das On-Board-SDS die eigenen Elemente, wie Vokabular, Sprachausgaben und Dialogabläufe, selbst an. Dadurch und durch die Realisierung von Aspekten der Natürlichsprachlichkeit (s. o.) soll das On-Board-SDS flexibler und intuitiver werden.

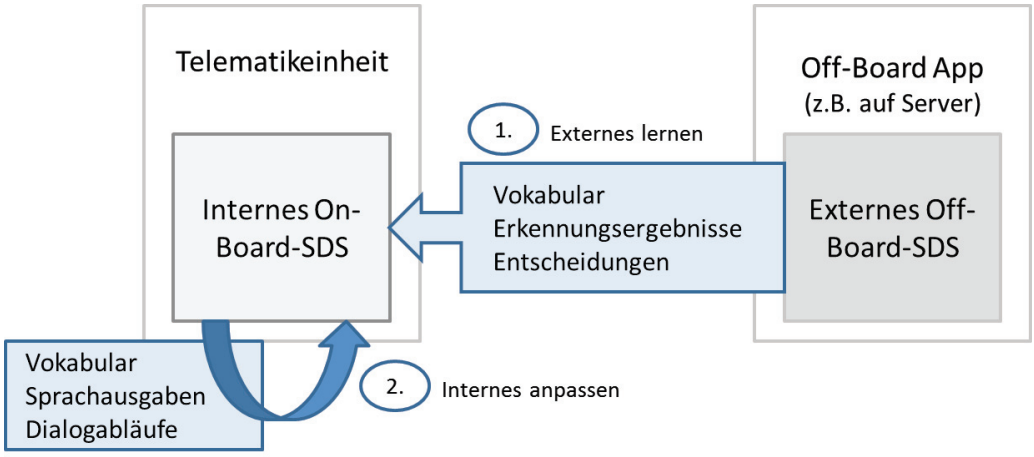

Abbildung 3 - Schematische Darstellung des Lern-Konzepts

Konkrete Möglichkeiten sind z. B. Anpassungen des Dialogflusses durch Adaption oder sogar Übernahme von Dialogen/Dialogpfaden. Des Weiteren ist es auch möglich, die Sprachausgaben des einen, internen SDS an die Sprachausgaben der anderen, externen SDS anzupassen. Wir stellen im nächsten Abschnitt unseren Ansatz und die entsprechende Implementierung zur *Verkürzung von Sprachausgaben* dar.

4 Intelligentes automatisches Lernen: Verkürzung von Sprachausgaben

In der praktischen Umsetzung soll vorrangig das Konzept des intelligenten Lernens behandelt werden. Insbesondere beschäftigen wir uns mit der Verkürzung von System-Sprach-Ausgaben, den sog. System-Prompts. Wie bereits am Anfang von Abschnitt 3 erwähnt, kann man die dort genannten Herausforderungen nicht komplett voneinander isolieren. Eine *Promptverkürzung* hat unweigerlich auch Auswirkungen auf die Natürlichsprachlichkeit, welche beachtet werden müssen. Wir erklären die übereinstimmenden Wörter zur semantischen Minialform des Prompts, auf die er maximal reduziert werden kann, um noch für den User verständlich zu sein. Umgesetzt wird diese Verkürzung durch einen Abgleich, der die Menge der übereinstimmenden Wörter durch Tokenisierung und Lemmatisierung ermittelt.

4.1 Dialogvergleich

Als erster Überblick für eine mögliche *Promptverkürzung* dient ein Dialogvergleich, welcher in Abb. 4 gezeigt ist. In der linken Spalte befindet sich der interne Zieleingabe-Dialog, in der rechten ein vergleichbarer, externer Dialog. Die fett markierten Wörter (die *Token*) stellen die Menge an Übereinstimmungen dar, auf die ein Prompt automatisch maximal verkürzt werden kann, ohne dass die ursprüngliche Semantik der Aussage negativ beeinflusst wird. Ein erster Implementierungsansatz berücksichtigt lediglich diese absolute Form der Verkürzung (also keine graduelle), erlaubt aber auch Übereinstimmungen, die sich erst nach zusätzlich durchgeführter Lemmatisierung der Token ergeben (vgl. *„gestartet“* vs. *„Starten“*). Der Ablauf einer Umsetzung dessen ist in Abb. 5 dargestellt. Um allerdings die entsprechenden, ähnlichen Dialoge (in diesem Fall: *ziel_eingeben*) als Input zur Verfügung zu haben, muss das nötige Preprocessing, die Identifizierung von ähnlichen Dialogen bzw. Dialogschritten, zuerst durchgeführt werden. Als Input hierfür dienen die Mengen von On-Board- sowie Off-Board-Dialogen, D_{on} und D_{off}, welche nach Applikationsmodulen sortiert werden. Diese Sortierung führen wir durch, da wir annehmen, dass Dialoge der selben Domäne den systeminherenten ähnlicher und somit sinnvoller zu adaptieren sind. Folglich erhalten wir für jede interne und jede externe Applikation eine

Interner Dialog	Externer Dialog
U: “Ziel eingeben.”	U: “Ziel eingeben.”
S: “**Bitte** sagen Sie den Namen des **Ortes**, der **Straße und** die **Hausnummer**.”	S: “Ort, Straße und Hausnummer, bitte.”
U: “Tübingen, Wilhelmstraße 19. ”	U: “Tübingen, Wilhelmstraße 19. ”
S: “Ist diese **Adresse korrekt**?” *Wilhelmstraße 19* *72074 Tübingen*	S: “Adresse korrekt?” *Wilhelmstraße 19* *72074 Tübingen*
U: “Ja.”	U: “Ja.”
S: “Die Zielführung wird **gestartet**.”	S: “Starten...”

Abbildung 4 - Vergleich zweier Dialoge anhand ihrer System-Prompts.

spezifische Menge an Dialogen, z. B. $D_{navi_{on}}, D_{navi_{off}}$. Diese werden dann auf ähnliche Dialoge untersucht wie z. B. Zieleingabe-Dialoge: $d_{ziel\_eingeben_{on}}, d_{ziel\_eingeben_{off}}$. Diese Dialoge dienen als Eingaben für die Identifizierung ähnlicher System-Prompts.

4.2 Promptverkürzung

Wie in Abschnitt 4.1 diskutiert, kann auf Dialogebene zwischen unterschiedlichen Dialogschritten, d. h. zwischen Benutzereingaben, Systemausgaben, Funktionsaufrufen der Head-Unit-Module und Aufrufen anderer Dialogabläufe unterschieden werden (vgl. Abbildung 2). Aufgrund dessen, dass wir eine Verkürzung der System-Prompts durchführen wollen, wird nun nach ähnlichen Systemausgaben gefiltert, um beispielsweise Paare wie $p_{adresse\_korrekt_{on}}$, $p_{adresse\_korrekt_{off}}$ herauszufinden.

Sind diese gefunden, werden sie analog zu Abb. 4 miteinander verglichen. Als nächstes wird der Text bzw. der jeweilige Prompt *tokenisiert* (d. h. in Wörter aufgespalten) und somit mit Indizes versehen. Nun wird jedes *Token* im On-Board-Dialog-Prompt $t_{i_{on}}$ mit jedem Token im Off-Board-Dialog-Prompt $t_{j_{off}}$ abgeglichen. Wenn die Token übereinstimmen („matchen“), werden die entsprechenden Elemente zu einer Liste von übereinstimmenden Token hinzugefügt. Sollten nicht alle Token des internen SDS-Prompts ein entsprechendes Gegenstück gefunden haben, so wird zusätzlich noch eine Lemmatisierung durchgeführt. D. h. jedes Token bekommt in einer nächsten Ebene seine unflektierte Grundform zugewiesen. Diese Lemmata können dann wiederum miteinander verglichen werden und so erhöht sich aufgrund der Abstraktionsebene die Wahrscheinlichkeit eine Übereinstimmung zu erzielen. Ein Beispiel hierfür sind die jeweils letzten Systemausgaben in Abb. 4: Das Partizip „gestartet“ im internen Dialog wird mit dem Infinitiv „starten“ aus dem externen Dialog semantisch gleich gesetzt. In gewisser Weise könnte man also von einem *semantischen* Abgleich sprechen, der durch die Lemmatisierung möglich gemacht wird. Selbstverständlich ist dies u. a. nur möglich, da wir einen Wort-Vergleich durchführen dessen Semantik sich kompositionell zur Semantik der Aussage zusammensetzen lässt. Dies ist dadurch begründet, dass wir zum Einen nur einzelne Sätze isoliert betrachten, sodass keine Abhängigkeiten zu anderen Aussagen bestehen. Zum Anderen müssen diese Systemaussagen eindeutig sein, damit der User genau weiß, welches Input er dem System zurückliefern soll bzw. was das System aufgrund seiner Aussagen verstanden hat.

Nachdem die Lemmatisierung durchgeführt worden ist, werden die übereinstimmenden Elemente (*Matchende Elemente M* in Abb. 5) wiederum zu der entsprechenden Liste hinzugefügt. So ergibt sich nach Tokenisierung und ggf. Lemmatisierung eine Menge an Wörtern, die für die

Bedeutung des aktuell untersuchten Prompts (in diesem Fall $p_{adresse\_korrekt}$) essentiell sind. Auf diese kann nun der Prompt maximal verkürzt werden.

Ein wichtiger Punkt, der bei der Verkürzung von Systemausgaben nicht außer Acht gelassen werden darf, ist die *Höflichkeit*. In [7] wird von Ribeiro und Benest beschrieben, dass das System generell höflich sein muss – insbesondere, wenn es die Aufmerksamkeit des Benutzers auf sich lenken will. Andererseits beschreiben die Autoren aber auch in ihrer Arbeit, dass menschliches (also „natürliches“) Verhalten dadurch nicht vollständig abgebildet ist, da in der zwischenmenschlichen Kommunikation nicht zu jedem Zeitpunkt Höflichkeit essentiell, sondern manchmal sogar unpassend ist. Allerdings muss man diese Entscheidung für oder gegen einen gewissen Grad an Höflichkeit situationsbedingt treffen, sowie insbesondere auf den SDS-Anwenderkreis, seine Gewohnheiten und Erwartungen abstimmen. Weitere Verkürzungsmöglichkeiten sowie Ergänzungen zu diesem Verfahren werden in Abschnitt 5 angesprochen.

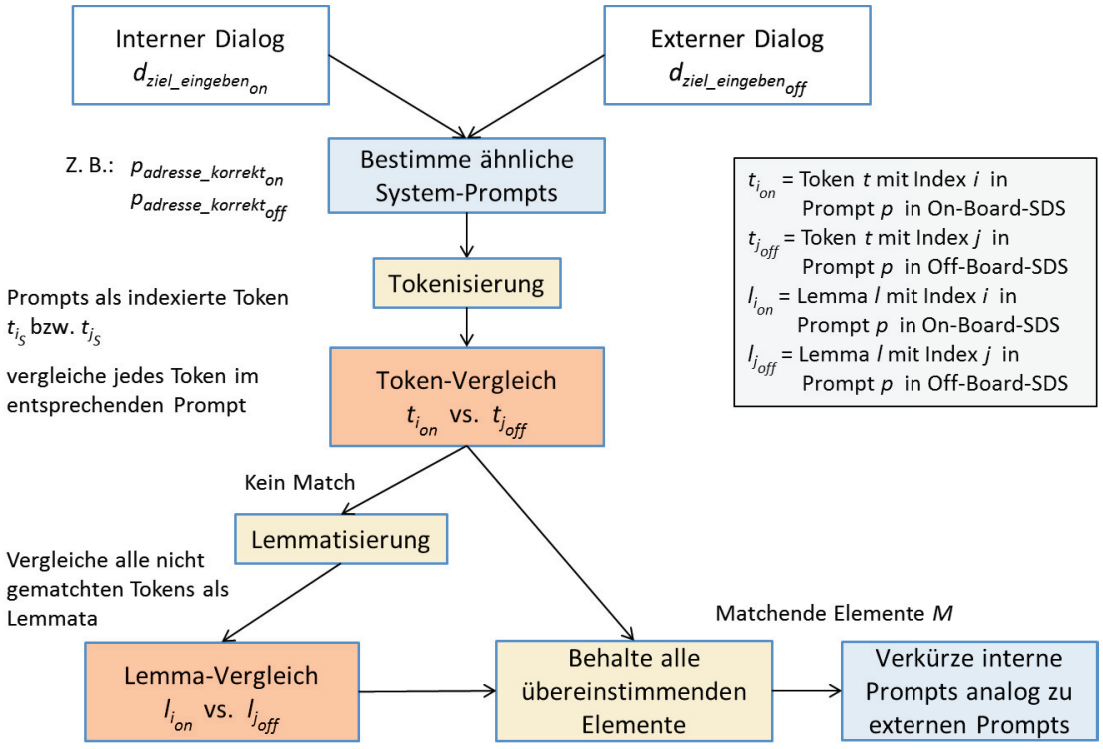

Abbildung 5 - Schematische Prozessdarstellung des Vergleichs von internen und externen Prompts.

4.3 Verwendete Tools

Für die Proof-of-Concept-Umsetzung der Tokenisierung und Lemmatisierung haben wir etablierte Tools verwendet: Für die Tokenisierung sowie die Satzerkennung (*sentence detection*) verwenden wir einen Tokenizer der *Apache OpenNLP*-Bibliothek[1] mit den entsprechenden Modellen. Die Lemmatisierung führen wir mit Hilfe des *TreeTaggers* [8] durch, der von Helmut Schmid am IMS der Universität Stuttgart entwickelt wurde.

Des Weiteren planen wir, *GermaNet* (vgl. Hamp und Feldweg [6]), ein Wortnetz für die deutsche Sprache, für die Extraktion von Synonymen sowie für die Berechnung von Ähnlichkeiten zwischen unterschiedlichen Begriffen zu verwenden. Dies wäre zum Einen von Vorteil, um –

[1] `http://opennlp.apache.org/`

abgesehen von den Prompts – den User auch dann zu verstehen, wenn er Synonyme oder nur ähnliche Begriffe verwendet. Zum Anderen könnte man durch diese Ressource Prompts vorrangig mit Synonymen alternieren und so zu einem *Random Prompting*-Ansatz gelangen, welcher zufällig einen Prompt aus einer Menge an semantisch ähnlichen Prompts an den User ausgibt. Dadurch könnte man einen weiteren Aspekt der Natürlichsprachlichkeit integrieren, der zudem laut [7] sinnvoll ist, da der Nutzer im schlimmsten Falle irritiert und sogar [vom Fahren] abgelenkt wird, wenn stets das Selbe wiederholt wird.

5 Zusammenfassung

Wir haben vier Herausforderungen beschrieben, die durch die kombinierte Verwendung von On-Board- und Off-Board-SDS entstehen: Kritisches Timing sowie Start/Ende von SDS-Sessions, Vokabulardisambiguierung, Natürlichsprachlichkeit und intelligentes Lernen von Off-Board-Charakteristika im On-Board-SDS. Die letzten beiden Herausforderungen beeinflussen die exemplarische Umsetzung der *Promptverkürzung*, welche die internen Prompts an ähnliche externe anpassen soll.

Weiterführende Arbeiten

Abschließende Ergebnisse und eine entsprechende Evaluation folgen in weiterführenden Arbeiten. Zudem wäre es möglich, eine unabhängige, computerlinguistische Promptanalyse mit ähnlichen Mitteln durchzuführen, sollte kein Dialog bzw. Prompt für einen Vergleich verfügbar sein. Allerdings entfällt dadurch die Möglichkeit, das On-Board-SDS durch externe Informationen „intelligenter“ zu machen. Für einen *Random Prompting*-Ansatz können die diversen, möglichen Wort-Reihenfolgen zusätzlich ohne wesentlichen Mehraufwand festgestellt werden.

Literatur

[1] BERG, M.: *Natürlichsprachlichkeit in Dialogsystemen*. Informatik-Spektrum, S. 1–11, 2012.

[2] BERG, M., A. DÜSTERHÖFT und B. THALHEIM: *Adaptation in Speech Dialogues – Possibilities to Make Human-Computer-Interaction More Natural*. Fifth Baltic Conference Human-Computer-Interaction, 2011.

[3] BLEIJENBERG, A.: *Cars and Carbon – The Attractiveness of Car Use*, Kap. 1, S. 19–42. Springer Netherlands, 2012.

[4] CARSTENSEN, K.-U.: *Sprachtechnologie – Ein Überblick*. http://www.kai-uwe-carstensen.de, 2012. Version 2.1.

[5] HAMERICH, S. W.: *Sprachbedienung im Automobil*. Springer Berlin Heidelberg, 2009.

[6] HAMP, B. und H. FELDWEG: *GermaNet – a Lexical-Semantic Net for German*. In: *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, 1997.

[7] RIBEIRO, N. M. und I. D. BENEST: *Invisible but Audible: Enhancing Information Awareness through Anthropomorphic Speech*. In: FAULKNER, X., J. FINLAY und F. DÉTIENNE (Hrsg.): *People and Computers XVI - Memorable Yet Invisible*, S. 17–35. Springer London, 2002.

[8] SCHMID, H.: *Probabilistic Part-of-Speech Tagging Using Decision Trees*. In: *Proceedings of International Conference on New Methods in Language Processing*, 1994.