hochschule mannheim

Fakultät für Informatik

Entwicklung und Evaluation eines Sprachassistentensystem auf Basis von Open-Source-Komponente für Raspberry Pi

Kolloqium der Bachelor-Thesis

Stella Neser

Erstkorrektor: Prof. Dr. Oliver Hummel, Hochschule Mannheim

Zweitkorrektor: Theresa Sick, SPACE; StartupLab@HSMA

Mannheim, 02.07.2021

Gliederung

1. Aufgabenstellung
2. Warum Entwicklung eines Berta Sprachassistenten?
3. Verwante Systeme
4. Grundlagen
5. Systemkonzept
6. Implementierung
7. Testen des Systems und Evaluation
8. Zusammenfassung & Ausblick
9. Demo
10. Fragen

Aufgabenstellung

Ausgeschriebenes Thema:

Entwicklung einer eigenen "Offline-Alexa" auf Basis eines Raspberry Pi und Open-Source-Komponenten, im Rahmen des StartupLab@HSMA.

Behandeltes Thema:

Development and evaluation of a voice assistant system based on open source components for Raspberry Pi

Namesgebung für den Sprachassistenten:

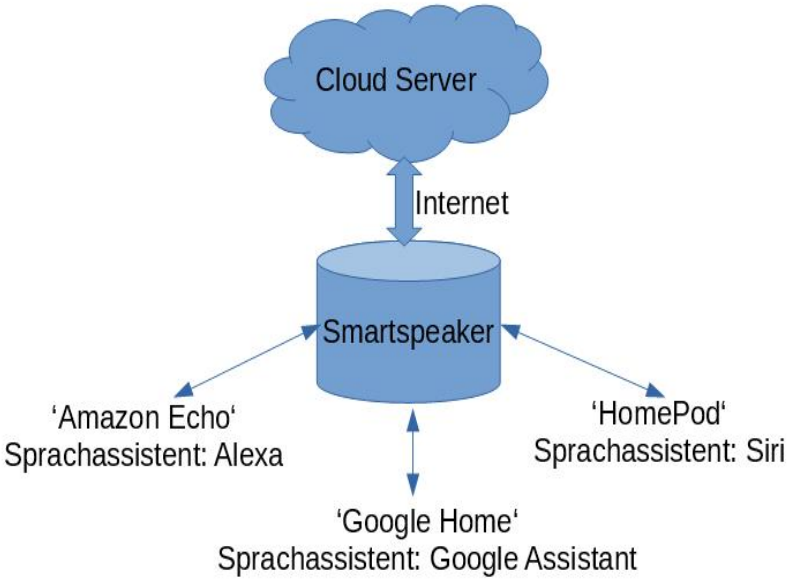
- Berta → in Anlehnung an Berta Benz

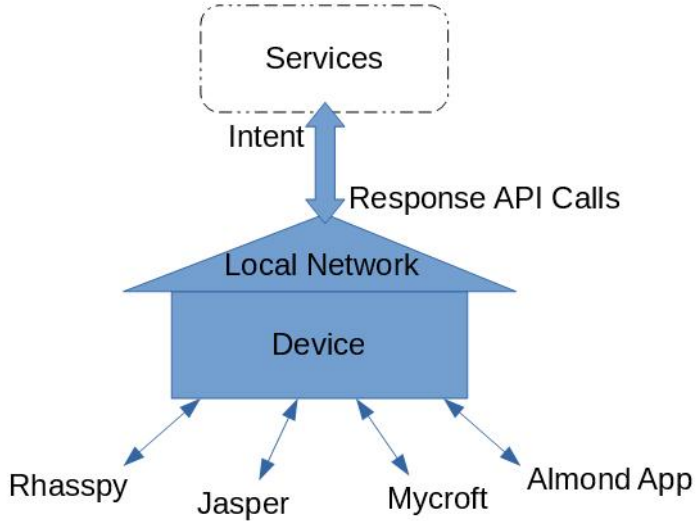
Warum Entwicklung eines Berta Sprachassistenten?

RQ1: Kann ein Sprachassistent mit Open-Source-Komponenten gebaut werden und in ähnlicher Weise funktionieren
ähnlich wie bekannte Systeme arbeiten?

RQ2: Kann ein Sprachassistent mit Open-Source-Komponenten gebaut werden, um offline zu argumentieren
ohne persönliche Daten zu verwenden?

Verwante Systeme





Cloud basierte bekannte Sprachassistenten Systeme

'Persönliche' Sprachassistenten Systeme zum teilweise offline verwenden und selbst konfigurieren

Grundlagen: Auswahlkriterien

Die folgenden vorgestellten Technologien wurden nach den folgenden Kriterien ausgesucht:

- Lizenz
- Raspberry Pi kompatible
- Ob die Softwarekomponente die Fähigkeit hat, seine Datenverarbeitung nur auf dem Pi durchzuführen (offline-ability)

Grundlagen: Engines

Picovoice Porcupine engine

Wake word Engine, trainiert in real-worl Umgebungen mit deep neural network.

- Frei nutzbar mit Demo Wörtern
  → “bumblebee“ und “computer“
- Benutzerdefiniertes
  → “wake word“ Testversion nur 30 Tage
  → “wake word“ mit kosten verbunden für distribution
  → “wake word“ Raspberry Pi Testversion nur über Enterprise console
- Raspberry Pi kompatibel

Grundlagen: Engines

DeepSpeech von Mozilla

Sprach-zu-Text Engine

Modell trainiert mit maschinellem Lernen auf Grundlage von Baidus Deep-Speech-Forschungsarbeit.

- Frei erhältliches vortrainiertes Englisches model
- Benutzt Google‘s TensorFlow um implementation einfacher zu gestalten
- Kommt als .pbmm oder .tflite package/ model type
  → Rasberry Pi nur TensorFlow Lite fähige Pakete veröffentlicht
- Entwickler geben Möglichkeit ein eigenes Model zu trainieren

Grundlagen: Engines

SVOX Pico Voice

Text-zu-Sprach Engine

Open-Source Anwendung mit geringem Platzbedarf

- Engine wurde verwendet in Android 1.6 “Donut“
- Support für sechs Sprachen
- Angenehme Stimme
- Rasperry Pi kompatibel

Grundlagen: Webframework

Flask

Python Webframework, sog. Microframework

- Keine Struktur Vorgabe wie Applikation aussehen soll
- Keine Datenbank Vorgabe
- Unterstützt Erweiterungen
  → Flask-WTF für CSRF handling mit SECRET_KEY
- Sehr guter Community Support mit Vielzahl an Erweiterungen
- Schnell einzulernen
- Nicht alleine geeignet für Produktionsumgebung

Grundlagen: Software

NGINX

Etablierter Open-Source Webserver

- verwendet als Reverse Proxy; Public facing Webserver
- leistungsstarke Anwendungsbereitstellung für Microservices
- kann für zukünftige Arbeiten verwendet werden.

Gunicorn mit Supervisor

Webserver Gateway-Schnittstelle und Monitoring Tool

- Gunicorn verwendet als Application Webserver
- Supervisor verwendet für monitoring und controlling Gunicorn

Grundlagen: Datenbank

SQLite3

Prozessinterne Bibliothek, die ein relationales Datenbanksystem enthält.

- Implementiert eine konfigurationsfreie, transaktionale SQL-Datenbank engine
- Serverloses Design und Benutzerfreundlich
- Performance auch gut bei geringem Speicher Umgebungen
- Verwendbar auf einer Vielfalt von Maschinen → Rasperry Pi kompatibel

Grundlagen: Datenbank

SQLAlchemy

Objektrelationale Abbildung (englisch object-relational mapping, ORM)

- Ermöglicht Anwendungen eine Datenbank zu verwalten
  → für Berta SQLite Datenbank
- Mit Klassen, Objekten und Methoden anstatt mit Tabellen und SQL
- Liefert Daten aus der Datenbank als Python Objekt.
- Support für einige Datenbank engines

Flask-SQLAlchemy

Erweiterung für Flask Anwendungen mit SQLAlchemy unterstützung

- Vorteile durch Konfigurationsschlüssel
  → SQLALCHEMY_DATABASE_URI

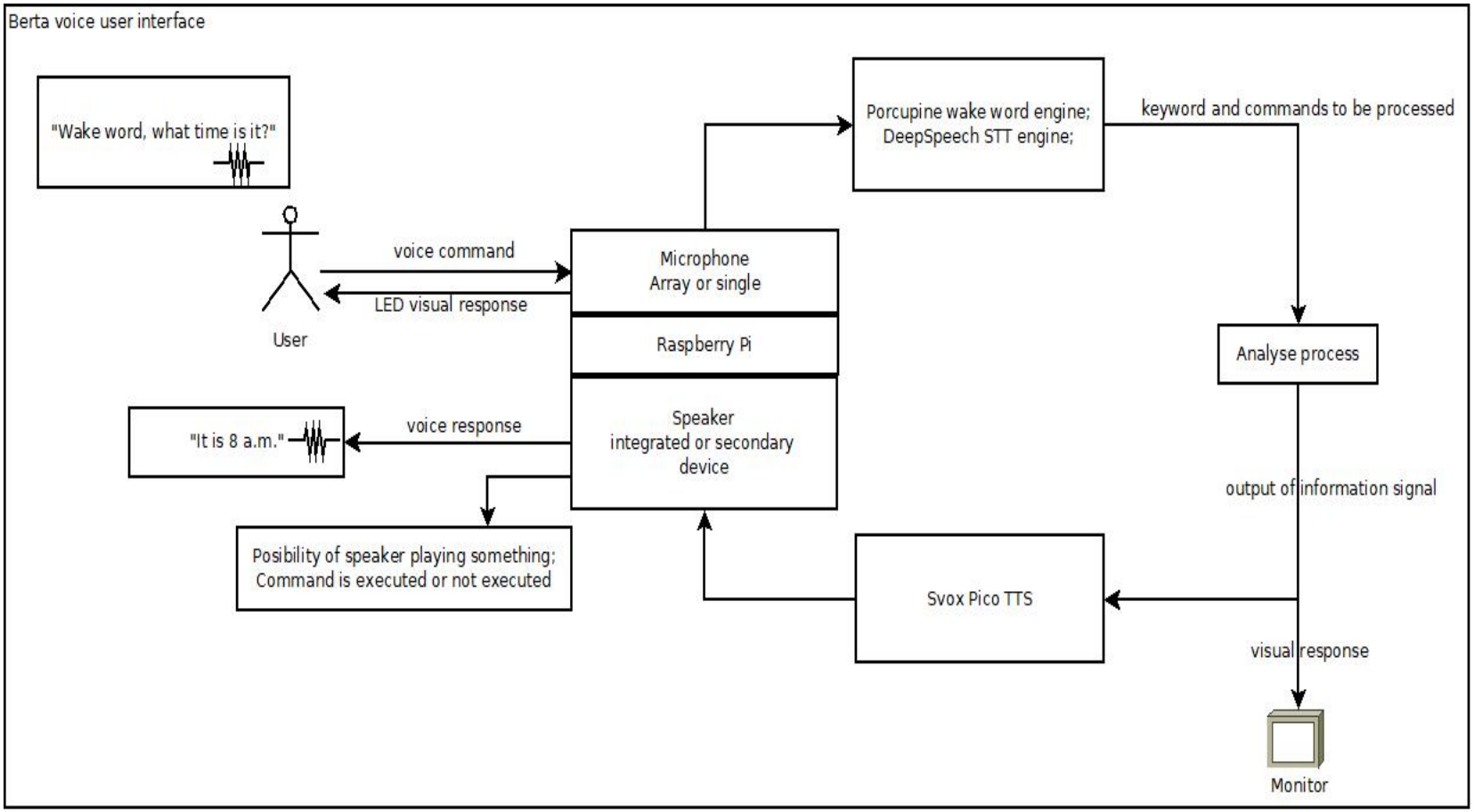
Grundlagen: Konzept

Duck-Typing

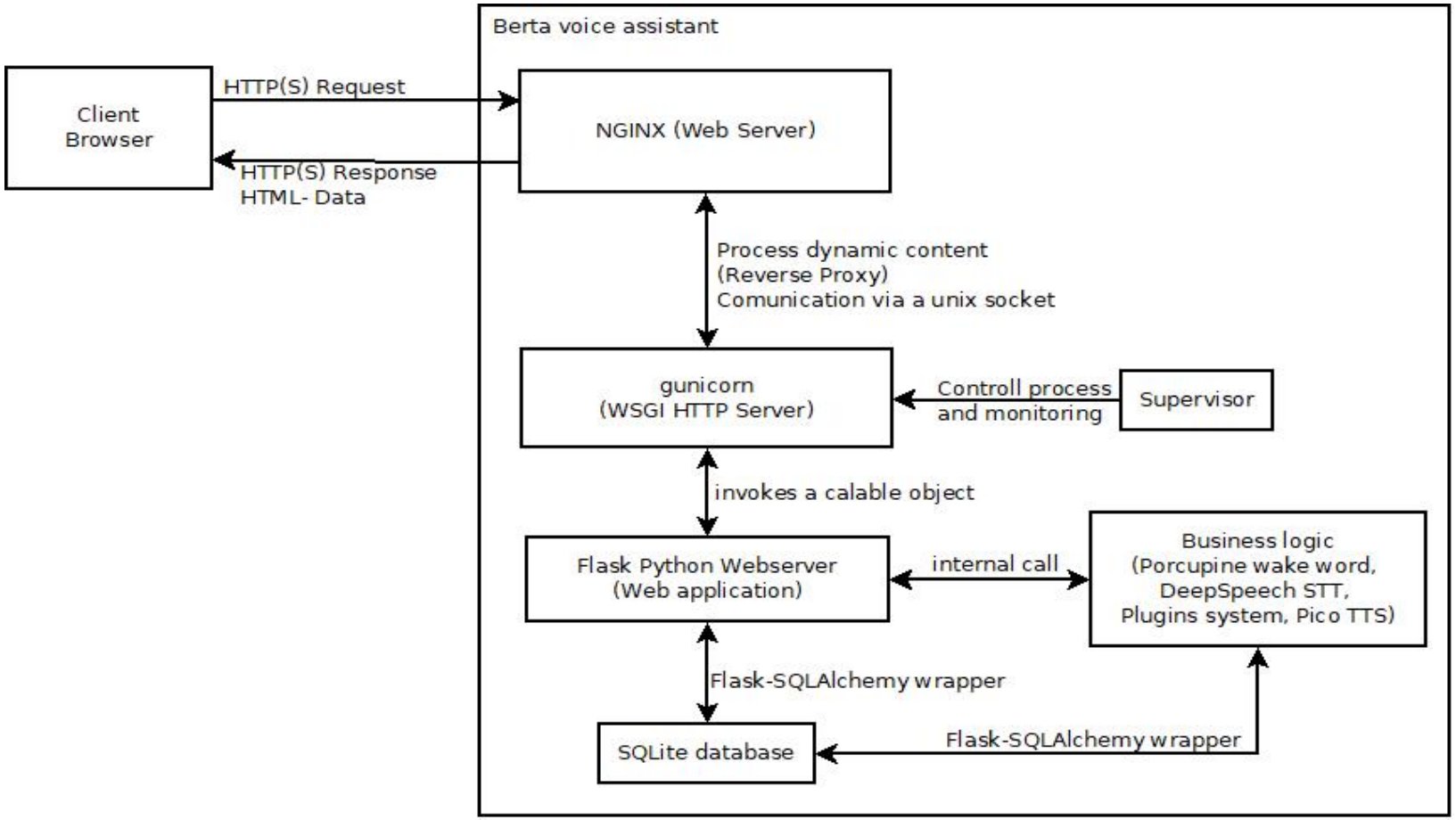
“If it walks like a duck, and it quacks like a duck, then it must be a duck.”

- Konzept aus der objektorientierten Programmierung
- Eine Klasse hat Methoden und von dieser können Objekte erzeugt werden. Wird ein Objekt erzeugt werden nur Methoden dieser Klasse betrachtet, ob diese vorhanden sind oder nicht, der Typ des Objekts ist zweitrangig.

Systemkonzept: Voice user interface



Systemkonzept: Berta Architektur

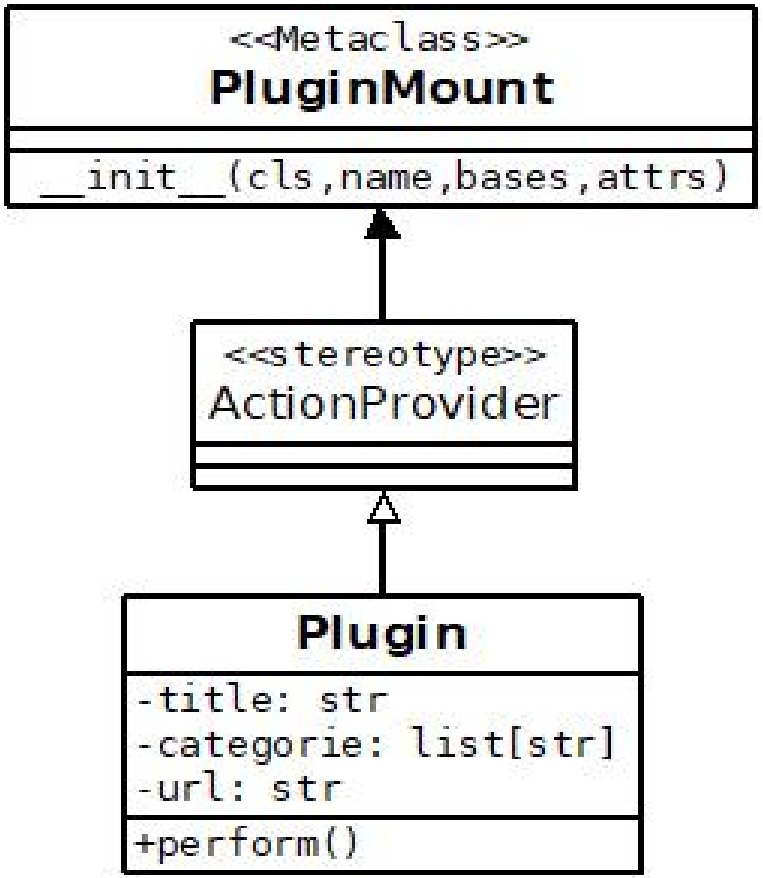


Systemkonzept: Berta Datenbankmodell

users	
*id	integer auto
*username	varchar(64)
*email	varchar(120)
*password_hash	varchar(128)
°about_me	varchar(140)
*last_seen	datetime

deep_speech_logs	
*id	integer auto
*question	varchar(140)
*answer	varchar(140)
*timestamp	datetime

Systemkonzept: Berta plugin class diagram



Implementierung: Generic plugin

```
class GenericPlugin(ActionProvider):
""" Class Documentation String """
# Every plugin must have a title that describes it
title = "My generic plugin"
# Every plugin must define at least one key word it should react upon
categories = ['generic']
#if necessary, provide an URL of a foreign API
url = 'http://some.api.com/news'

def perform(self):
""" Action to be perfomed """
# ...
# API Processing here
# ...
return answer
```

Implementierung: Funktionen

Funktionen die Berta ausführen kann:

- Simple function
  → keyword “simple“
- Time announcement
  → keyword “time“
- Date announcement
  → keyword “date“
- Default function
  → keyword “incomprehensible spoken words“
- Gimmick function
  → keyword “joke“

Testen des Systems und Evaluation

2 Eigenschaften getestet im Verglich zu Google Assistent:

Usability Test:

10 Testpersonen, jeder stellt jeweils 5 Fragen an das System

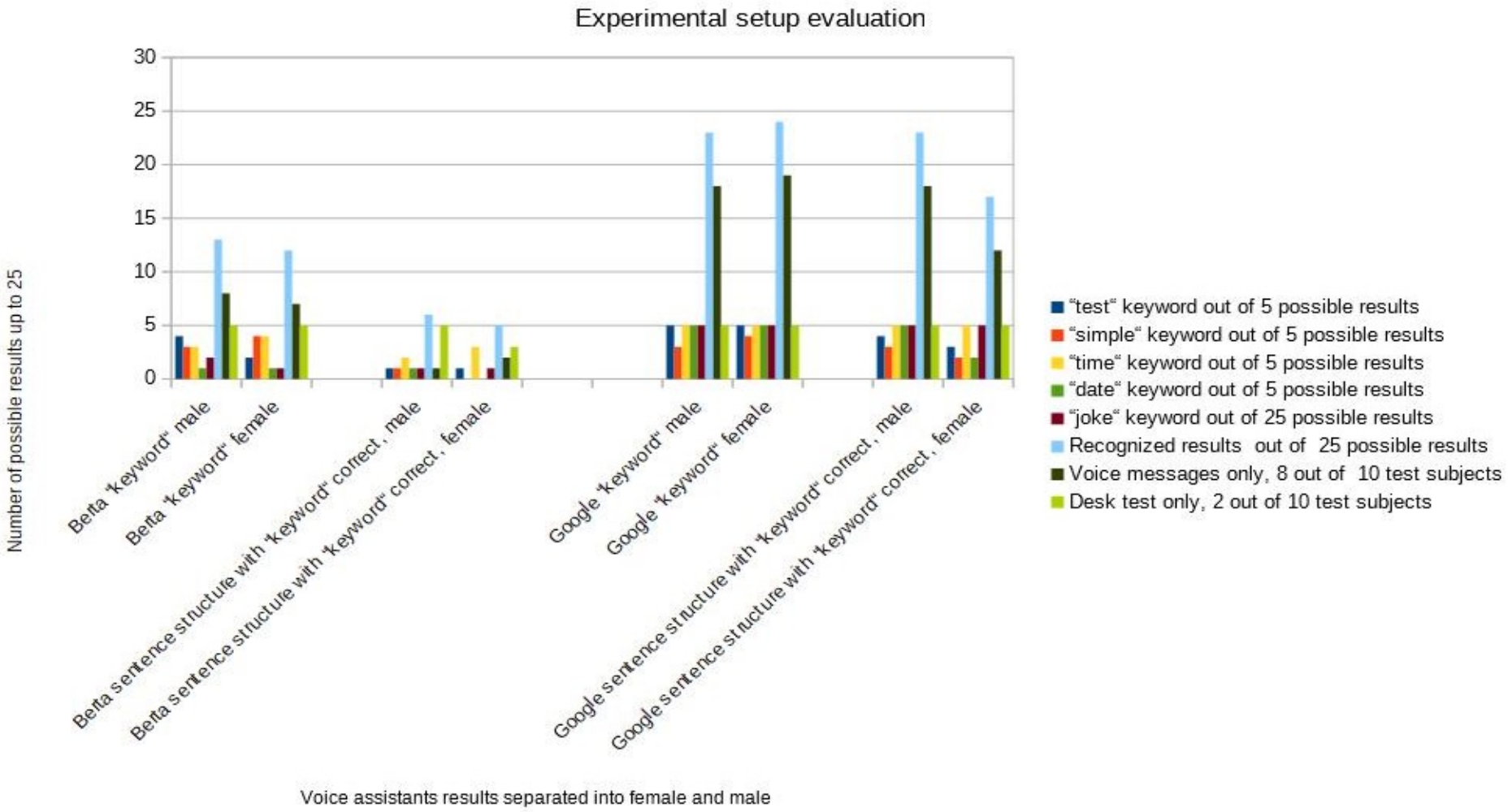
- 2 Schreibtisch Tests
- 8 Sprachnachrichten Tests die Berta vorgespielt wurden.

Ergebnis: Unbefriedigend, weil testen mit Sprachnachrichten nicht gut war.

Offline-ability Test:

Wireshark wurde während Usability Tests im Hintergrund laufen gelassen

Ergebnis: Unbefriedigend, wegen mangelnden Kenntnissen.


Experimental setup evaluation
Number of possible results up to 25
30
25
20
15
10
5
0
Berta "keyword" male
Berta "keyword" female
Berta sentence structure with "keyword" correct, male
Berta sentence structure with "keyword" correct, female
Google "keyword" male
Google "keyword" female
Google sentence structure with "keyword" correct, male
Google sentence structure with "keyword" correct, female
Voice assistants results separated into female and male
"test" keyword out of 5 possible results
"simple" keyword out of 5 possible results
"time" keyword out of 5 possible results
"date" keyword out of 5 possible results
"joke" keyword out of 25 possible results
Recognized results out of 25 possible results
Voice messages only, 8 out of 10 test subjects
Desk test only, 2 out of 10 test subjects

Zusammenfassung & Ausblick

Zusammenfassung

- RQ1 mit Berta Sprachassistent bestätigt
  → mit open-source Komponenten implementiert
  → verhält sich ähnlich zu herkömmlichen bekannten Systemen
- RQ2 mit Berta Sprachassistent bestätigt
  → mit open-source Komponenten implementiert
  → funktioniert offline ohne Verwendung persönlicher Daten

Ausblick

- Weboberfläche upgraden
- DeepSpeech Model ausbauen
- Mehr Funktionalitäten
- APIs über Web-Applikation nutzen
- Berta Service

Demo

Danke für Ihre Aufmerksamkeit

Kolloquium der Bachelor-Thesis
Stella Neser
Mannheim, 02.07.2021

Fragen?