Gruppenprojekt Komputer & Creativität

Wishing Well

Elias Dellago David Gergess Yevhen Prots Selina Weber

Wintersemester 2020/21
Technische Universität München

Inhaltsverzeichnis

1 Einleitung

Das Projekt *Wishing Well* ist das Ergebnis der Projektphase des Kurses *Komputer & Creativität*. Zielvorgabe war die Planung und Umsetzung eines Grußkartengenerators. Aspekte wie die Entwicklung kreativer Software, das Design einer Web-App und ein angenehmes Nutzererlebnis standen dabei im Fokus. Die Umsetzung von *Wishing Well* basiert auf einem hybriden Ansatz unterschiedlicher Arten der Textgeneration, bei dem Anlass, Stimmung und Zusatzinformationen berücksichtigt werden. Die Sprache, in der das Projekt realisiert wurde, ist Englisch, um die Umsetzung des Programms zu erleichtern und um ein breites Spektrum an potentiellen Nutzern anzusprechen.

Die nachfolgende Dokumentation soll einen Überblick über *Wishing Well* bieten und die wichtigsten technischen und kreativen Aspekte näher beleuchten.

2 Technischer Aufbau

2.1 Systemarchitektur

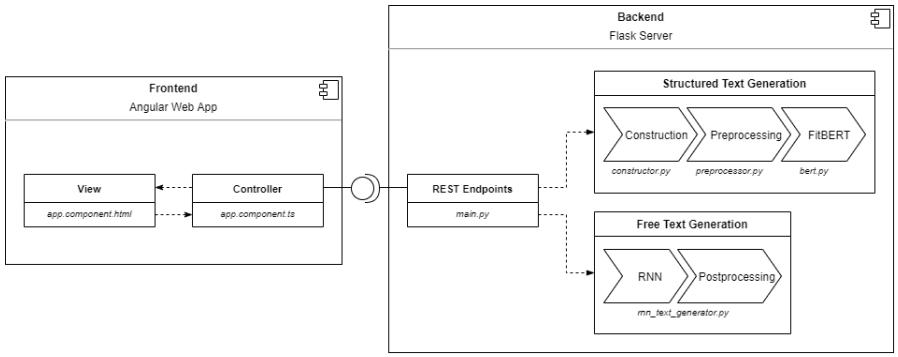

Abbildung 1: Deployment Diagram

Das Projekt *Wishing Well* wurde als interaktive Web-Anwendung konzipiert. Die Anwendung wurde in zwei Module unterteilt: Frontend und Backend. Das Frontend stellt eine in Angular implementierte graphische Benutzeroberfläche bereit. Das Backend wurde in Python entwickelt und umfasst verschiedene Methoden zur Textgeneration, mithilfe von Neuronalen Netzen.

Angular folgt der klassischen *Model-View-Controller* Architektur und stellt Möglichkeiten zur Synchronisierung des HTML View mit einem Typescript Controller bereit. Das Backend basiert auf einem Flask Server und ist funktional aufgebaut. Die Kommunikation zwischen Front- und Backend passiert mittels einer REST-Api, sodass ein Client-Server Informationsfluss entsteht.

Das Backend lädt zu Startup-Time alle benötigten Neuronalen Netze in den Arbeitsspeicher und reagiert auf Anfragen des Frontend, indem entsprechende Prozeduren ausgeführt werden und das dabei produzierte Artefakt zurückgegeben wird.

2.2 Graphical User Interface

Das Graphical User Interface (GUI) stellt einen kritischen Aspekt bei der Benutzung unseres Programmes dar, da dies die visuelle Repräsentation unseres Projektes darstellt. Für eine gelungene User Experience müssen zwei wesentliche Merkmale unterschieden werden:

1. Die unbedingt benötigten Funktionialitäten
2. Das Design

Notwendige Funktionalitäten

Zu den unbedingt erforderlichen Funktionalitäten der Webseite gehört vor allem die Eingabe der Variablen und Parameter, welche das Programm im Backend für die Textgenerierung benötigt. In einer frühen Phase des Projektes wurde ein Prototyp (damals noch ohne Angular) für die Webseite entwickelt, welcher die ersten wesentlichen Funktionalitäten bereits beinhaltete. Diese waren die Eingabe des Anlasses, der Stimmung und optionale anlassspezifische Angaben, für die das Programm einen Grußkartentext entwickeln sollte.

Abbildung 2: Erster Prototyp

Beim Foranschreiten des Projektes kamen neben neuen Inputparametern vor allem noch die Feedback Funktionalitäten hinzu. Die ursprünglichen 3 Eingabeparameter wurden mit der Angabe zu der Erzählperspektive und der Auswahl eines RNNs für die Generierung der Beispielsätze ergänzt. Die Eingabe der

Erzählperpektive soll es dem Programm ermöglichen, möglichst persönliche und authentische Texte zu generieren. Zu den Feedback Funktionalitäten gehören die Funktionen, welche die Wortwahl ändern, den Text neu generieren lassen und neue Satzvorschläge erstellen. Um dem Benutzer den Umgang mit dem Programm zu vereinfachen, wurde die tatsächliche Implementation aller Funktionalitäten zum größten Teil mit Buttons und Radio Buttons realisiert.

Design

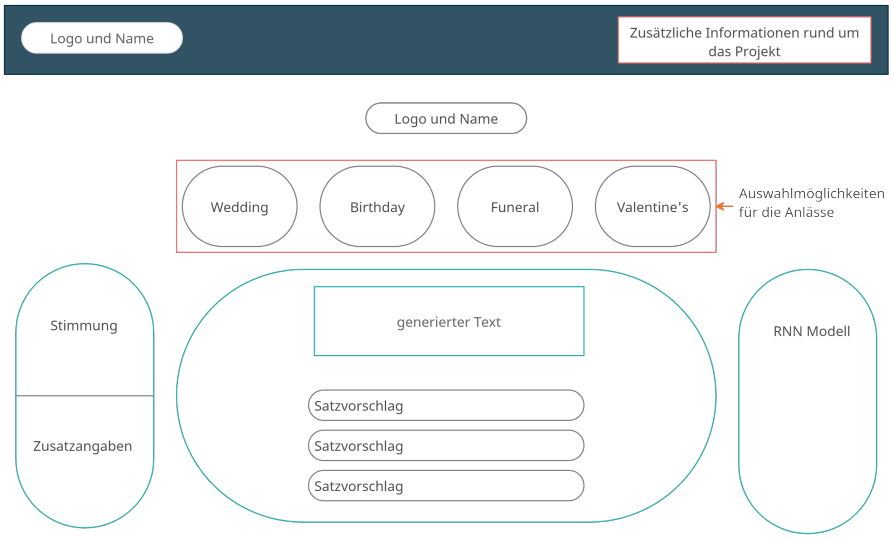

Abbildung 3: Grundlegendes Design

Parallel zu der Implementierung der Funktionalitäten wurde das Design der Website weiterentwickelt. Der Fokus hierbei war stets dem Benutzer eine angenehme Atmosphäre zu bieten. Ein Aspekt, welcher hier eine zentrale Rolle gespielt hat, war die Farbauswahl. Mit viel Feingefühl wurde versucht, passende Farben für die gesamte Seite, als auch für die Anlässe zu finden. Ebenfalls wurden die auf eine intuitove Anordnung der Buttons geachtet.

Zuletzt wurde noch eine Navigationsleiste hinzugefügt, welche dem User genauere Informationen zu der Bedienung der Webseite, über die Funktionsweise des Programms und über das Team bzw. das Projekt liefert.

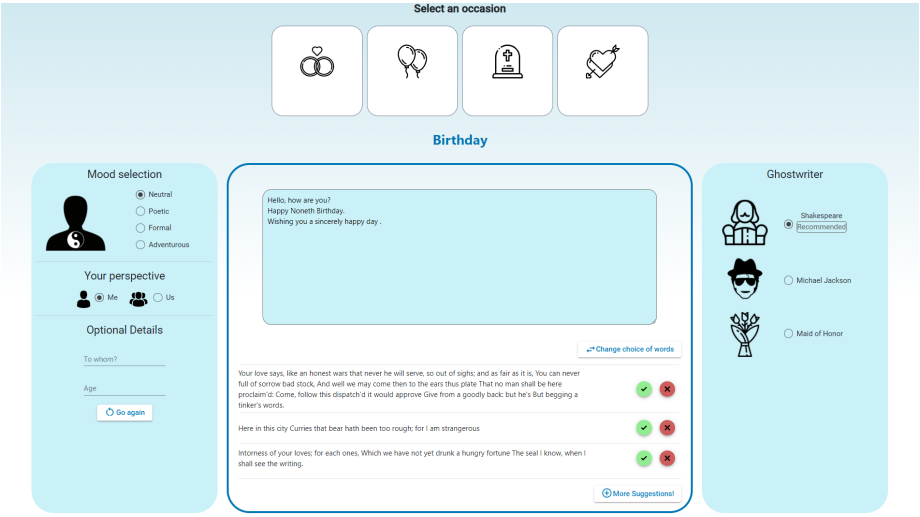

Abbildung 4: Finales Design der Website (Ausschnitt)

2.2.1 Nutzerinteraktion

Wie oben bereits erwähnt, lag bei der Implementierung der Webseite der Fokus auf einfacher Handhabbarkeit. Um alle Funktionen des Programms ausschöpfen zu können, wird lediglich der Anlass und die Stimmung für die Textgeneration benötigt. Zusätzlich kann der Nutzer Informationen angeben, welche anschließend mit berücksichtigt werden. Sollte der Text nicht zufriedenstellend sein, so kann der User ihn neu generieren lassen, die Wortwahl im Text ändern oder Satzvorschläge zum Text hinzufügen. In der folgenden Abbildung wird dieses Konzept graphisch dargestellt.

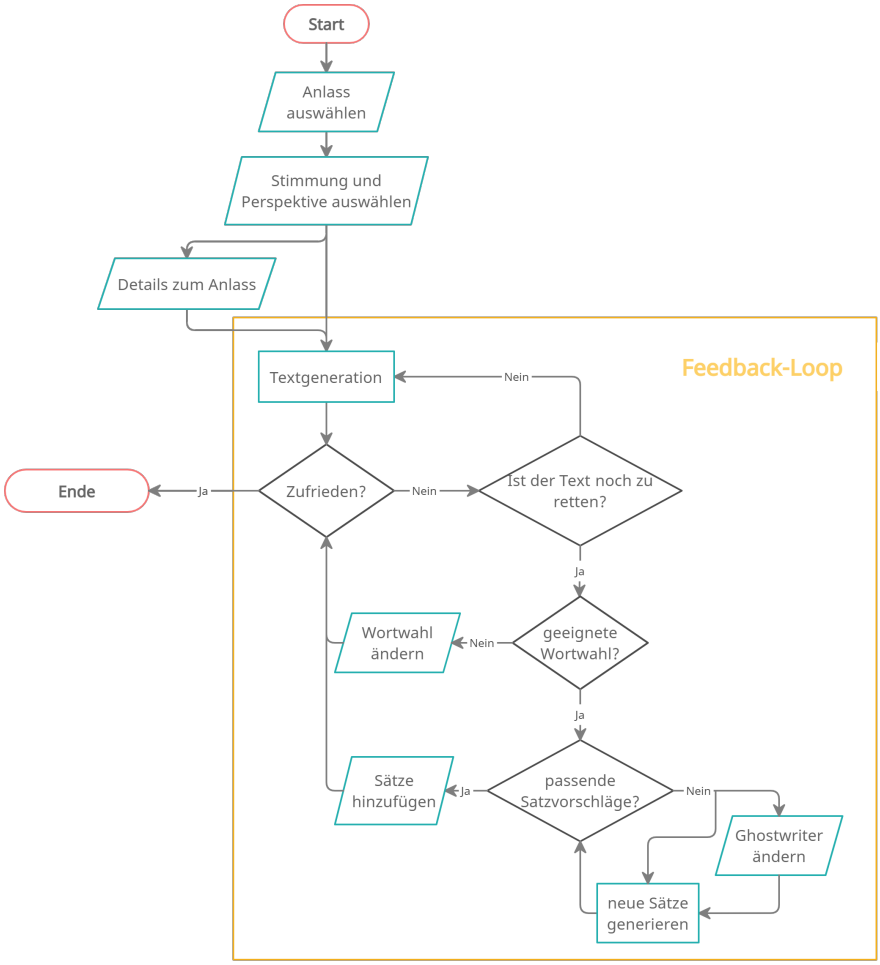

Abbildung 5: Bedienung der Webseite

2.3 Strukturierte Textgeneration mit BERT

2.3.1 Satzgeneration nach Kochrezept

Eine der Erkenntnisse aus unserem Ideenfindungsprozess war, dass es bei allen Arten von Grußkarten eine klar erkennbare Struktur geben sollte. Wir haben deshalb den Ansatz verfolgt, generische Syntax-Blöcke zu erstellen, die im Folgenden als *Templates* bezeichnet werden. Diese wurden erarbeitet, indem in verschiedenen Glückwunsch-Sätzen austauschbare Variablen identifiziert wurden. Ein solches Template hat beispielsweise folgende Form:

```
"<name> <sender> wish you <wish_happy> for your
    wedding and a <positive> time together"
```

Für jede solche frei wählbare "Lücke"wurden Wörterbücher angelegt, kategorisiert nach der Stimmung, die die jeweiligen Wörter vermitteln. Beim Erstellen der Templates wurde darauf geachtet, dass neben diesen generischen Lücken auch anlassspezifische Informationen vorkommen, um dem Benutzer weitere Anpassungsmöglichkeiten zu geben. So sollte es etwa möglich, aber nicht verpflichtend sein, beispielsweise das Alter des Geburtstagskinds oder die Namen des Hochzeitspaars zu spezifizieren.

Im Erstellungsprozess wird zunächst aus verschiedenen Templates ein parametrisierter Gesamttext konstruiert, der eine klare Grußkarten-Struktur besitzt, unter anderem auch eine Grußformel am Anfang. Variablen, welche sich auf spezifische Informationen oder Syntax beziehen, werden anschließend regelbasiert ersetzt. Stimmungsgebende Variablen werden in einem weiteren Schritt mithilfe von FitBERT verarbeitet.

2.3.2 Intelligente Lückenfüllung mit BERT

Die Generation von Sätzen, welche nur auf vorgegebenen Templates basiert, ist eine Lösung, die leider stark die Auswahl an möglichen Varianten limitiert und die Kreativität in diesem Prozessanteil nur der Randomisierung überlässt. Daraus entstehen folgende Probleme:

1. Die Kreativität der Software steht unter Zweifel
2. Die Qualität der Endergebnisse kann nicht garantiert werden, da zufällig ausgewählte Wörter den Sinn des Satzes gefährden können
3. Zufällig ausgewählte Wörter können wegen unterschiedlichem Satzbau grammatikalisch inkorrekt eingesetzt werden.

Um diese Risiken auszuschließen, wurde BERT in der Endsoftware verwendet.

Was ist BERT?

Bidirectional Encoder Representations from Transformers oder einfach *BERT* ist eine Machine Learning Technik für Natural Language Processing, welche von Google entwickelt wurde.[3]

Als ein auf einem Transformer basierendes Modell erkennt BERT kontextuelle Beziehungen zwischen Wörten im Text. Im Gegensatz zu Richtungsmodellen, die den Text sequentiell von links nach rechts oder von rechts nach links bearbeiten, lesen Tansformer die ganze Sequenz von Wörten auf einmal, daher der Name bidirektional. Diese Eigenschaft ermöglicht es dem Modell, den Kontext eines Wortes anhand seiner gesamten Umgebung (links und rechts vom Wort) zu lernen. Die allgemeine Funktionsweise von einem Transformerencoder ist wie folgt: Der Input, eine Sequenz von Tokens, wird im NN verarbeitet. Der Output, eine Sequenz von Vektoren, weist jedem Vektor einen Inputtoken mit dem gleichen Index zu.

Beim Training von Sprachmodellen besteht die Herausforderung darin, ein Vorhersageziel zu definieren. Viele Modelle sagen das nächste Wort in einer Sequenz voraus (e.g. "The child came home from ___"), ein Richtungsansatz, der das Lernen im Kontext einschränkt.

Um diese Herausforderung zu bewältigen, verwendet BERT zwei Trainingsstrategien:

1. Masked LM(MLM): vor dem Abschicken von Wortsequenzen in BERT werden 15% der Wörter in jeder Sequenz durch ein MASK-Token ersetzt. Das Modell versucht den ursprünglichen Wert des MASK-Wortes vorherzusagen, basierend auf dem Kontext aller anderen Wörter.

2. Next Sequence Prediction(NSP): das Model nimmt Paare von Sätzen als Input und lernt vorherzusagen, ob der zweite Satz des Paares der nachfolgende Satz im Originaldokument ist.

Beim Training des BERT-Modells werden MLM und NSP zusammen trainiert, um die kombinierte Lossfunktion beider Strategien zu minimieren.

FitBERT

Um die Verwendung vom BERT im Rahmen dieses Projektes zu vereinfachen, wurde entschieden, eine Bibliothek zu verwenden, die die Notwendigkeit eliminiert, BERT komplett aufzubauen. Eine andere Notwendigkeit für die Bibliothekenverwendung ist die Interaktion mit BERT möglichst schnell und flexibel zu machen. Um dies zu erreichen wurde FitBERT bei Qordoba entwickelt.[1]

FitBERT steht für "*Fill in the Blanks, BERT*"; und nutzt die Eigenschaft, dass BERT(Masked Language Modeling) sich sehr gut zum Füllen von Textlücken

eignet.

Nachdem FitBERT instanziiert wurde, übergeben wir einen Satz mit einer Lücke zum Ausfüllen und eine Liste von Wörtern, die diese Lücke potentiell ersetzen können.

```
#Input
masked_string = "Why Bert, you’re looking ***mask*** today!"
options = [’buff’, ’handsome’, ’strong’]

#Output >>> "Why Bert, you’re looking handsome today!"
```

FitBERT kann sogar einige potentiell falsche Wörter korrigieren, durch die Anpassung des Wortes an die Satzstruktur: Singular, Plural, Form usw.

```
#Input
unmasked_string = "Why Bert, you’re looks handsome today!"

#Output >>> "Why Bert, you’re looking handsome today!"
```

Diese beiden Eigenschaften machen FitBERT extrem nützlich für unsere Anwendung.

Der übergebene Text muss zunächst vorbereitet werden: die Satzzeichen werden von Wörtern und MASKs getrennt, New-Line Charakters werden gelöscht, um die weitere Erkennung von MASKs fehlerfrei gewährleisten zu können. Der Hauptteil besteht darin, bei jedem Durchlaufen zuerst die Anzahl an MASKs zu zählen. Gibt es nur ein einziges MASK im Satz darf FitBERT sofort angewendet werden. Falls nicht, soll der Satz zuerst gekürzt werden.

Da BERT den Kontext analysieren und bearbeiten kann, soll diese Verkürzung so verlaufen, dass soviel wie möglich im Satz enthalten bleibt. Aus diesem Grund, bleibt alles bis zum nächsten MASK im Text und nur der Teil ab dem nach-

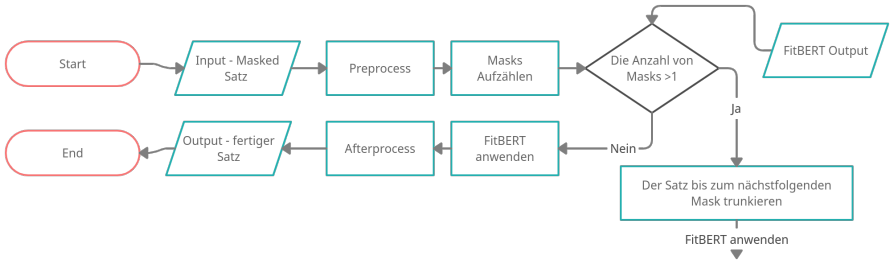

Abbildung 6: Hauptteil FitBERT

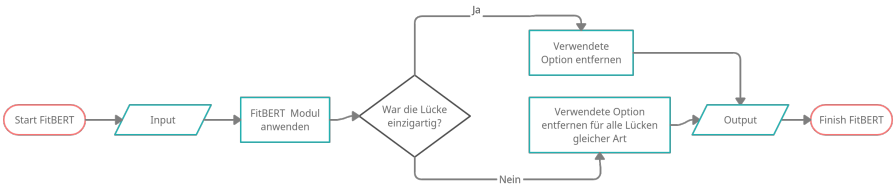

Abbildung 7: Mask Ersetzung

folgenden MASK wird gekürzt. Dies führt nicht nur zu einer reibungsfreien Anwendung von BERT, sondern dient auch dazu, dass die Vorteile der Bidirektionalität von BERT genutzt werden können. Ab diesem Punkt kann FitBERT verwendet werden.

An die FitBERT Funktion wird ein Satz mit einem MASK und ein Wörterbuch mit mehreren Optionen für das MASK übergeben. Danach wird mit Hilfe des FitBERT Moduls dieses MASK mit der passendsten Option ersetzt. Die gewählte Option wird anschließend aus dem Wörterbuch entfernt und zusätzlich auch aus allen weiteren Wörterbüchern, die für andere Lücken herangezogen werden, um Tautologien zu verhindern. Dieser Schritt ermöglicht beim Feedback, dass die Lücke mit einer anderen Option ersetzt werden kann, wenn dem Benutzer das Ergebnis nicht gefällt. Der Satz mit dem ersetzten MASK wird zurück an den Hauptteil übergeben, zusammen mit dem reduzierten Wörterbuch.

Wenn alle MASKs im ursprünglichen Satz ersetzt wurden, wird der ganzen Text nachbearbeitet: Satzzeichen und New-line Charakters werden entsprechend zurück eingesetzt. Der fertige Satz wird zusammen mit dem Wörterbuch an das Frontend zurückgegeben, wo der Benutzer Feedback geben kann.

2.4 Freie Textgeneration mit RNNs

Der zweite Textteil, der nicht über die strukturierte Textgeneration erzeugt wird, basiert auf der Verwendung von rekurrenten neuronalen Netzen. Für das Projekt werden von uns trainierte Modelle benutzt. Modelle wurden aus drei Kategorien trainiert: Shakespeare, Michael Jackson und Maid of Honor. Für das Shakespeare Modell wurde ein fertiger Datensatz von Github für das Training verwendet, der anschließend noch leicht modifiziert wurde;[2] für Michael Jackson und Maid of Honor wurden die Datensätze selbst erstellt. Für Michael Jackson wurden, bis auf ein Lied, alle Songtexte verwendet, mit kleinen Korrekturen für oft wiederholten Text. Die Daten des Maid of Honor Datensatzes bestehen aus verschiedenen Hochzeitsreden. Einige unklare Sätze wurde herausgenommen und alle Namen durch die Namen Anna und Todd ersetzt, um die Weiterverarbeitung zu gewährleisten. In den folgenden Absätzen wird dieses

Vorgehen näher erläutert.

Mit Werten von bis zu 80 Epochen und unterschiedlichen Freiheitsgraden (sogenannten Temperaturen), was die Exaktheit des Textes im Vergleich zum Trainingstext angeht, wurden die Modelle trainiert. Die hier verwendeten Modelle sind ausgewählt, weil sie sich als bestes Netz in einer Gruppe von Netzen gezeigt haben, die jeweils auf den gleichen Datensatz trainiert wurde. Insgesamt wurden über 40 Modelle trainiert. Die Datensätze hatten in den finalen Version folgende Größe:

- Shakespeare: 1.035.721 Zeichen
- Michael Jackson: 34.510 Zeichen
- Maid of Honor: 128.928 Zeichen

Für die Erzeugung der Modelle wurde ein Algorithmus verwendet, der sich sehr an der RNN Textgeneration von Tensorflow orientiert.[2] Von unserer Seite wurden kleine Änderungen hinzugefügt, um das Netz für unseren Zweck entsprechend benutzen zu können. Auch wurde der Teil zum Speichern und Laden der Modelle von uns übernommen. Der Ansatz, der hier verwendet wird, basiert auf der Vorhersage des jeweils nächsten Buchstabens. Der Text wird für die Erstellung der Trainingseinheiten in Blöcke von 100 Zeichen Länge unterteilt. Ein aus drei Layern (Embedding Layer, Gated Recurrent Unit und Dense Layer) aufgebautes Modell wird darauf trainiert, zu jedem der 100 Zeichen langen Eingabedaten die ensprechenden Ausgabedaten zu finden; die Zieldaten sind die gleichen wie die Eingabedaten, nur um einen Buchstaben verschoben. Wenn die Eingabe "Hallo zusammen, ein sehr schöner Ta"heißt, dann lautet der Zielstring ällo zusammen, ein sehr schöner Tag". Das Modell ruft sich also immer wieder selbst auf und gibt den nachfolgenden Buchstaben aus. Bei dem Training werden so vom Modell Wörter und auch Sätze erlernt, ebenso wie die Struktur des Textes.

Für die Generation von Text, der am Ende auf der Website angezeigt werden soll, wird das entsprechende Modell als Parameter übergeben. Die aufgerufene Funktion generiert einen Text mit bis zu 1100 Zeichen. Diese hohe Anzahl an Zeichen wird benötigt, da es sonst zu Fehlern in der Weiterverarbeitung kommt. Zudem möchten wir verhindern, dass ein generierter Satz abgeschnitten und unvollständig wird. Um nun nur einen Satz an das Frontend zurückzugeben, wird der Text je nach Art weiter bearbeitet. Bei Shakespeare wird entweder nach einem Punkt, einem Ausrufezeichen oder einem Fragezeichen abgeschnitten, je nachdem welches Satzzeichen als erstes vorkommt. Spätestes aber wird nach einer Leerzeile abgeschnitten, da in Original Shakespeare Texten nach einer Leerzeile ein neuer Sprecher die Bühne betritt. Wir möchten also nicht mehr als maximal einen Absatz an generiertem Text zurückgeben und dem Benutzer zur Auswahl stellen. Dieser Ansatz basiert auf der entsprechenden Feedback Funktion. Wie im Teil der Nutzerinteraktion bereits beschrieben, kann ein Satz zum

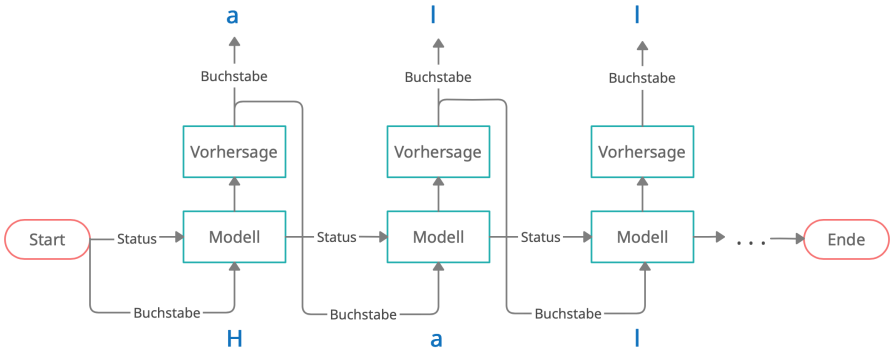

Abbildung 8: Textgeneration im RNN

Text hinzugefügt werden oder kann verworfen werden.

Für das Michael Jackson Modell, in dem als einziges Satzzeichen Kommas auftreten, wird nach zwei Zeile abgeschnitten und übergeben, um einen gewissen Lesefluss zu gewährleisten. Bei dem Text der Maid of Honor wird eine ähnliche Strategie angewendet wie bei Shakespeare: es wird nach dem ersten Auftauchen eines Punktes, Ausrufezeichens oder Fragezeichens abgeschnitten und übergeben. Zusätzlich müssen für diesen Text einige Namensersetzungen vorgenommen werden. Da der Text bei der Generation einer Hochzeitkarte empfohlen, gibt es entsprechende Felder für die Namen von Braut und Bräutigam. Das Modell wurde auf die Namen Anna und Todd trainiert, ist also in der Lage, nach Generation des Textes diese beiden Namen durch die tatsächlichen Namen zu ersetzen. Zusätzliche Namensersetzungen wurden vorgenommen, falls das Netz außer Anna und Todd Namen erwähnt (tritt in einzelnen Fällen auf und ist auf einen nicht ganz sauberen Datensatz zurückzuführen).

3 Projektorganisation

3.1 Arbeitsaufteilung

Der Ansatz unserer Arbeitsaufteilung orientiert sich an den Präferenzen des Einzelnen. Jeder sollte die Möglichkeit bekommen, sich mit einer Aufgabe zu befassen, die ihn interessiert. Die grobe Aufteilung lautet wie folgt:

- Elias: Erstellung der Templatesätze, REST-API, Web-App Dynamik
- David: Erstellen und Gestalten der Web-App
- Yevhen: Intelligentes Lückenfüllen der Templates mit BERT
- Selina: Freie Textgeneration mithilfe eines RNN

Wichtige Entscheidungen wie das Design der Website, die Art der Texte und die Feedback Funktion wurden in gemeinsamen Sessions erarbeitet. Auch überschneiden sich einzelne Aufgaben oder es wurde zusammen an Details gearbeitet. Die genannte Aufgabenverteilung stellt also nur die Hauptbereiche der einzelnen Mitglieder dar.

In regelmäßigen Besprechnungen wurden isolierte Arbeitspakete als Issues identifiziert und auf die Gruppenmitglieder verteilt. Erkenntnisse und Probleme bei der Ausarbeitung wurden im jeweiligen Issue protokolliert, beispielsweise der Trainingsprozess der Neuronalen Netze. Die Issues wurden der aktuellen Projektphase entsprechend Meilensteinen zugeordnet, wodurch eine ausgewogene Verteilung des Zeitaufwands zumindest angepeilt wurde.

3.2 Herausforderungen und Erkenntnisse

Eine wichtige Grundsatzfrage war für uns, in welchem Ausmaß maschinelles Lernen zur Erstellung eines Grußkartentexts verwendet werden sollte. Bei realistischem Entwicklungsaufwand ergibt sich hierbei folgender Tradeoff: Maschinelle Kreativität gegen Anpassungsmöglichkeiten. Bei einem rein auf neuronalen Netzen basierendem Ansatz können kaum Garantien über die Textform gegeben werden und die Steuerbarkeit durch den Benutzer ist stark limitiert. Bei einem rein regelbasierten Ansatz leidet jedoch der Aspekt der maschinellen Kreativität stark. Deshalb haben wir uns für einen hybriden Ansatz entschieden, der ein minimal brauchbares Artefakt garantiert, welches dann erweitert werden kann.

Während des gesamten Entwicklungsprozesses kam es immer wieder zu Problemen mit der Verwendung verschiedener Funktionen, Bibliotheken und Systembausteinen; zu allen auftretenden Schwierigkeiten wurden nach ausführlichen Recherchen Lösungen gefunden.

Bei der FitBERT Verwendung wurde durch ausführliche Tests festgestellt, dass für die Erzeugung bestmöglicher Lösungsqualität auf die Verwendung von Phrasen verzichten und nur auf alleinstehende Wörter begrenzen werden sollte. Ansonsten brauchte das Programm entweder sehr lang, um eine Auswahl zu treffen, reagierte nicht mehr oder gab etwas Falsches aus.

Die bereits erwähnten Tautologien bei benachbarten MASKs mit gleichen Wörterbüchern waren eine Herausforderung, die uns länger beschäftigte. Als Lösung einigten wir uns auf eine gleichzeitige Anpassung aller Wörterbücher.

Andere Probleme bei der Anwendung von FitBERT lagen meistens an Kleinigkeiten: beispielsweise der richtigen Zeichensetzung, korrekter Maskenersetzung oder Maskensuche in Templates und Ähnlichem; alle Punkte konnten durch mehrmaliges Testen gelöst werden.

Bei der Erstellung der Datensätze kam das Problem auf, dass einige der verwen-

deten Texte nicht lang genug waren. Bei gleichen Trainingsparametern funktionierten diese Modelle deutlich schlechter als Modelle, deren Trainingssätze ungefähr 30 Mal so groß waren. Die Lösung, mit der im fertigen Algorithmus auch gearbeitet wird, ist, die Modelle stärker zu trainieren, was zwar die Satzstruktur deutlich verbessert, sie allerdings auch ähnlicher zum Ausgangstext macht. Die jetzigen Modelle stellen daher einen Kompromiss zwischen sinnvollen Wörtern mit (oft) korrekter Grammatik und der Freiheit des Netzes dar, nicht exakt die Trainingsdaten zu reproduzieren. Diese Herausforderung trat vor allem beim Training für Michael Jackson auf, da nur eine begrenzte Anzahl an Songs zur Verfügung stand und wir uns dagegen entschieden, die Texte anderer Künstler zusätzlich zu verwenden.

Auch die bereits erwähnten Ersetzungen von Namen im Modell der Maid of Honor ist ein Problem, das aufgrund der Datensätze auftritt. In einigen Fällen erscheint in dem vom RNN generierten Text mehrere Namen, statt wie vorgesehen nur die Namen Anna und Todd. Dies ist auf einen nicht ganz reinen Datensatz zurückzuführen. Zwar wurden so viele Namen wie möglich schon vor dem Training ersetzt, dennoch kann es vorkommen, dass das RNN weitere Namen generiert. Bei eingehenden Tests wurden manuell die auftauchenden Namen ersetzt, sodass die Gesamtzahl im fertigen Text deutlich reduziert wurde.

Bei der von Tensorflow vorgegebenen Art, trainierte Modelle zu speichern und wieder zu laden, kam es anfänglich zu Schwierigkeiten. Variablen konnten nicht mehr aufgerufen werden oder wurden doppelt abgebildet, was die gespeicherten Modelle unbrauchbar machte. Erst mit der Nutzung einer älteren Funktion von Keras konnten die Modelle schließlich wieder aufgerufen werden und direkt in eigenen Programmen zur Textgeneration verwendet werden.

4 Kreative Aspekte

4.1 Design Thinking

4.1.1 Benutzeroberfläche

Der Schlüssel zu einem brauchbaren Website-Design ist es, der Intuition des Benutzers gerecht zu werden. Der Aufbau der Website ist demnach klassisch gehalten, mit einer Kopfzeile, sowie Input und Output auf gleicher Höhe, um unnötiges Scrollen zu vermeiden. Es wurde versucht, die verschiedenen Einstellungen intuitiv graphisch durch teilweise selbst angefertigte Icons darzustellen. Daneben wird die allgemeine Benutzung der Website auch textuell beschrieben. Bei der Dynamik der Website war es uns wichtig, dem Benutzer Rückmeldung darüber zu geben, in welchem Zustand sich das System gerade befindet, so wurden Ladeanimationen und Error Handling integriert.

4.1.2 Artefakt

Wir haben uns dafür entschieden, unseren skalierbaren Ansatz im Rahmen dieses Projekts nur auf vier beliebte Anlässe anzuwenden. Bereits zu Beginn hatten wir zu jedem Anlass spezifische Informationen identifiziert, die man in einer solchen Grußkarte erwarten würde und die unser Programm anbieten sollte. Wir kamen ebenfalls zum Schluss, dass die Tonart einer Grußkarte stark vom Verhältnis zum Empfänger abhängig ist und durch bestimmte Schlüsselwörter verändert werden kann.

Neben diesen Grundlagen sollte unser System aber auch eine unerwartete Komponente als Mehrwert anbieten: dies wurde durch thematisch ähnliche, freie Textgeneration realisiert. Besonders wichtig war uns dabei, dem Benutzer Handlungsfreiraum zu geben. So präsentieren wir freie Textgeneration als Vorschläge, die übernommen werden, aber auch abgelehnt werden können. Zusätzlich sollte der strukturierte Teil auf mehreren Ebenen korrigierbar sein. So kann der Benutzer einen völlig neuen Satzbau fordern können, nur Stimmungswörter ersetzen oder manuell Änderungen am Text vornehmen.

4.2 Kreativer Prozess der Gruppe

Um einen roten Faden durch die Organisation des Projektes zu erhalten, wurde bereits am Anfang des Projektes vereinbart, dass Änderungen und Entscheidungen das Projekt betreffend, in gemeinsamen Gruppensitzungen diskutiert werden sollten. Um diesen Meetings eine gewisse Struktur zu verleihen, die später eingesehen werden kann, wurden alle Beschlüsse protokolliert.

Die Entwicklung unseres Projektes bestand zu Anfang aus mehreren Sitzungen reiner Ideenfindung, aus denen wir im Anschluss diejenigen auswählten, deren Umsetzung und Originalität uns realisierbar und sinnvoll erschienen. Nachdem der grobe Aufbau des Projektes (Web-App und Generation des Textes) beschlossen war, bekam jedes Mitglied die Freiheit, explorativ die Grenzen des ihm zugewiesenen Arbeitsbereiches zu erkunden.

Dadurch entwickelten wir unsere Ideen konstant weiter: die Anpassung an andere Programmteile, das Design neu hinzugefügter Funktionen und das Erweitern einzelner Teile durch neu entdeckte technische Möglichkeiten. Durch gegenseitiges Feedback konnten nicht nur die rein technischen Aspekte unseres Projektes, sondern auch die Zusammenarbeit und das Miteinander in der Gruppe verbessert werden.

4.3 Kreativer Prozess des Programms

Unser Ansatz, feste Templates, intelligente Lückenfüllung und freie Textgeneration zu kombinieren, führt zu einem Ineinandergreifen verschiedener kreativer Aspekte. Sowohl die Struktur und Zufälligkeit des kreativen Prozesses sind ge-

geben als auch verschiedene Arten von Kreativität.

Die Struktur in unserem Algorithmus ist an fast allen Stellen zu sehen. Am klarsten erscheint sie in den fertigen Templates, die eine relativ strenge Struktur für den ersten Textteil der Grußkarte vorgibt. Der frei generierte Textteil weist weniger Struktur auf, dennoch stehen dahinter fertig trainierte Modelle, die den Stil einer gewissen Person oder eines gewissen Anlassers erlernt haben.

Der Faktor der Zufälligkeit wird ebenfalls in mehreren Funktionen verwendet. So werden die Satzanfänge in allen Modellen zur freien Textgeneration zufällig ausgewählt. Das Ergebnis ist, dass mehr unterschiedliche Sätze erzeugt werden und die Wahrscheinlichkeit sinkt, den selben Satz mehrmals zu generieren. Für die Zufälligkeit der Templates wurden mehrere Vorlagen für den gleichen Anlass geschrieben. Unser Ziel ist, auch bei der festgelegten Struktur Variation einzubringen, um keinen statischen Text zu erzeugen.

Betrachtet man die intelligente Lückenfüllung von BERT, erkennt man den Ansatz der kombinatorischen Kreativität. BERT kombiniert Templates mit Wörtern, die ihm in diesem Kontext am Besten erscheinen. Dabei greift er zwar auf einen deutlich eingeschränkten Wortschatz zurück, trifft dennoch eigenständig selektive Entscheidungen.

Explorative Kreativität kann ebenfalls bei der freien Textgeneration beobachtet werden. Zwar bleibt ein Modell in der Struktur, in der es trainiert wurde, dennoch kombiniert es verschiedene Wörter und Satzteile, welche es erlernt hat. Dadurch entstehen neuartige Kombinationen, die leider auch nicht immer Sinn ergeben, da es an Reflexion fehlt. Dennoch kann ein ganzes Spektrum an neuen Sätzen erzeugt werden. Verbesserungsvorschläge und Erweiterungen werden im Ausblick diskutiert.

5 Ausblick

5.1 Erweiterung der Satztemplates

Die aktuellen Satztemplates folgen einer relativ strengen Form. Als Endresultat der Konstruktion erhalten wir Sätze, bei denen jeweils nur ein bis zwei Stimmungswörter ersetzt werden können, wodurch wir FitBERT wahrscheinlich etwas unterfordern. Neben neuen Templates und mehr Wörterbuch Einträgen wäre eine weitere Verbesserung, die Templates noch variabler zu gestalten, etwa durch Ersetzung auf mehreren Ebenen (Satz, Satzteil, Einzelwort).

5.2 Dynamische Wörterbücher

Eine weitere Option, wie man die Kreativität von Wishing Well verbessern kann, liegt in der Erzeugung von dynamischen Wörterbüchern für die BERT Anwen-

dung. Obwohl BERT selber die Entscheidungen trifft, welche der Optionen des Wörterbuchs am besten zu jeder Lücke passt, ist diese Auswahl begrenzt.

Eine mögliche Lösung dafür wurde in einem Prototyp entwickelt: ein RNN wird für die Generierung der Optionen verwendet. In der Theorie analysiert das RNN den ganzen Satz ohne MASK und generiert pro Iteration eine entsprechende Auswahl an Optionen, die potentiell eingesetzt werden können. Danach führt BERT seine Aufgabe wie gewohnt aus. Dieser Ansatz wurde nicht finalisiert.

5.3 Verbesserung des RNN-Lernprozess

Um die Generation des RNN basierten Textteiles zu verbessern, muss das Training der Modelle überarbeitet werden. Unter diese Kategorie fallen sämtliche Parameter und Rahmenbedingungen des Trainingsalgorithmus.

Eine deutliche Erweiterung der momentan recht geringen Trainingsdaten würde beispielsweise zu besseren Modellen führen. Parameter, wie die Anzahl der Epochen und die Temperatur, wurden für alle getesteten Modelle optimiert; unklar ist allerdings, ob wirklich die besten Kombinationen für die Netze gefunden wurden.

Ebenfalls ein Ansatz, der nicht weiter von uns untersucht wurde, ist die Vorhersagen von Wörtern, anstatt von Buchstaben. Wir entschieden uns für den Ansatz der Buchstabenvorhersage, da eine kurze Recherche darauf schließen ließ, dass Grammatik und Wörter damit leichter erlernbar wären. Für eine Erweiterung des jetzigen Projektes könnte ein alternativer Ansatz allerdings herangezogen und mit dem Momentanen kombiniert werden.

5.4 Selbstverbesserung

5.4.1 Benutzerpräferenzen

Bei Inbetriebnahme unseres Systems wäre es denkbar, dass die Zufriedenheit des Nutzers mit den aktuellen Parametern statistisch ausgewertet wird. Somit könnte man die Standardwerte für die verschiedenen Einstellungen nach Nutzerpräferenzen wählen. Daneben könnte bei der Auswahl der Satztemplates und Wörterbucheinträge eine Gewichtung nach Beliebtheit vorgenommen werden.

5.4.2 Autonome Modelloptimierung

Die momentanen Möglichkeiten des Feedbacks beschränken sich bei der freien Textgeneration auf das Annehmen oder Verwerfen einzelner Sätze. Sobald das Frontend es anfordert, wird ein einzelner Satz vom RNN zurückgegeben. Allerdings wird die Entscheidung nicht an das RNN zurückgegeben, da es solche Informationen nicht verarbeiten kann. Über die Rückgabewerte von Benutzerpräferenzen könnte sich das Modell selbst optimieren. Parameter wie die Anzahl der Trainingsepochen oder die Temperatur könnten vom Netz selbst eingestellt

werden und damit bessere Modelle trainiert werden. Auch der Ansatz eines unabhängigen zweiten Netzes, das die erzeugten Sätze korrigiert (in Anlehnung an das Konzept von GANs) und Verbesserungen am aktuellen Modell vornimmt, wäre denkbar.

6 Schlusswort

Für uns alle war dieses Projekt eine Möglichkeit, über uns hinauszuwachsen und uns neue Fähigkeiten anzueignen. Durch die intensive Zusammenarbeit in der Gruppe haben wir auch potentielle Tandempartner fürs Russisch lernen gefunden...
Stundenlange Meetings und kurzfristige Änderungen haben uns teilweise an unsere Grenzen gebracht und unsere Adrenalinpegel erhöht; aber der Spaß blieb nie auf der Strecke.

Des Weiteren möchten wir unseren Dank an Igor Rhzin aussprechen, welcher das Design unseres Schriftzugs übernommen hat. Unser Dank geht ebenfalls an den Kurs *Komputer & Creativität*, der uns zu diesem Projekt zusammengebracht hat, an Kaffee und die große Knusprigkeit™.

Alles was noch zu sagen bleibt, ist: Knusprig bleiben!

Literatur

[1] Biljana Rolih; Aneta Stal; Qordoba Jenkins; Sam Havens. *FitBERT - Use BERT to Fill in the Blanks*, 2020. `https://pypi.org/project/fitbert/`, letzte Einsicht 2021-02-03.

[2] Tensorflow. *Text generation with an RNN*, 2020. `https://www.tensorflow.org/tutorials/text/text_generation`, letzte Einsicht 2021-02-03.

[3] Jacob Devlin; Ming-Wei; Chang Kenton; Lee Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2019.