A LANGUAGE LEARNING FRAMEWORK BASED ON MACARONIC TEXTS

by

Adithya Renduchintala

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

August, 2020

Abstract

This thesis explores a new framework for foreign language (L2) education. Our framework introduces new L2 words and phrases interspersed within text written in the student's native language (L1), resulting in a *macaronic* document. We focus on utilizing the inherent ability of students to comprehend macaronic sentences incidentally and, in doing so, learn new attributes of a foreign language (vocabulary and phrases). Our goal is to build an AI-driven foreign language teacher, that converts any document written in a student's L1 (news articles, stories, novels, etc.) into a pedagogically useful macaronic document. A valuable property of macaronic instruction is that language learning is "disguised" as a simple reading activity.

In this pursuit, we first analyze how users guess the meaning of a single novel L2 word (a noun) placed within an L1 sentence. We study the features users tend to use as they guess the meaning of the L2 word. We then extend our model to handle multiple novel words and phrases in a single sentence, resulting in a graphical model that performs a modified cloze task. To do so, we also define a data structure that supports realizing the exponentially many macaronic configurations possible for a given sentence. We also explore ways to use a neural cloze language model trained only on L1 text as a “drop-in” replacement for a real human student. Finally, we report findings on modeling students navigating through a foreign language inflection learning task. We hope that these form a foundation for future research into the construction of AI-driven foreign language teachers using macaronic language.

Primary Reader and Advisor: Philipp Koehn

Committee Member and Advisor: Jason Eisner

Committee Member: Kevin Duh

Acknowledgements

Portions of this dissertation have been previously published in the following jointly authored papers: Renduchintala et al. (2016b), Renduchintala et al. (2016a), Knowles et al. (2016), Renduchintala, Koehn, and Eisner (2017),Renduchintala, Koehn, and Eisner (2019a) and Renduchintala, Koehn, and Eisner (2019b). This work would not be possible without the direct efforts of my co-authors Rebecca Knowles, Jason Eisner and Philipp Koehn.

Philipp Koehn has provided me with a calm, collected, and steady mentorship thought my Ph.D. life. Sometimes, I found myself interested in topics far removed from my thesis, but Philipp still supported me and gave me the freedom to pursue them. Jason Eisner raised the bar for what I thought being a researcher meant, opened my eyes to thinking deeply about a problem, and effectively communicated my findings. I like to believe Jason has made me a better researcher by teaching me (apart from all the technical knowledge) simple yet profound things – details matter, be mindful of the bigger picture even if you are solving a specific problem.

I am thankful to my defense committee and oral exam committee for valuable feedback and guidance: Kevin Duh, Tal Linzen, Chadia Abras, and Mark Dredze (alternate). I am incredibly grateful to Kevin Duh, who not only served on my committee but also mentored me on two projects leading to two publications. Kevin not only cared about my work but also about me and my well being in stressful times. Thank you, Kevin.

I want to thank Mark Dredze and Suchi Saria, my advisors, for a brief period when I first started my Ph.D. I am incredibly grateful to Adam Lopez, who first inspired me to research Machine Translation. I will never forget my first CLSP-PIRE workshop experience in Prague, thanks to Adam. Matt Post, along with Adam Lopez, were the instructors in my first year MT course, Matt has been fantastic to discuss project ideas and collaborate.

I have been lucky to work with excellent researchers in Speech recognition as well. Shinji Watanabe took a chance on me and helped me work through my first every speech paper in 2018, which eventually won the best paper. Shinji encouraged me and gave me the freedom to explore an idea even though I was not an experienced speech researcher. I would also like to thank Sanjeev Khudanpur and Najim Dehak for several discussions during my stint in the 2018 JSALT workshop.

I want to thank my fellow CLSP students and CLSP Alumni who shared some time with me. Thank you, Tongfei Chen, Jaejin Cho, Shuoyang Ding, Katie Henry, Jonathan Jones, Huda Khayrallah, Gaurav Kumar, Ke Li, Ruizhi Li, Chu-Cheng Lin, Xutai Ma, Matthew Maciejewski, Matthew Weisner, Kelly Marchisio, Chandler May, Arya McCarthy, Hongyuan Mei, Sabrina Mielke, Phani Nidadavolu, Raghavendra Pappagari, Adam Poliak, Samik Sadhu, Elizabeth Salesky, Peter Schulam, Pamela Shapiro, Suzanna Sia, David Snyder, Brian Thompson, Tim Vieira, Yiming Wang, Zachary Wood-Doughty, Winston Wu, Patrick Xia, Hainan Xu, Jinyi Yang, Xuan Zhang, Ryan Cotterell, Naomi Saphra, Pushpendre Rastogi, Adrian Benton, Rachel Rudinger and Keisuke Sakaguchi, Yuan Cao, Matt Gormley, Ann Irvine, Keith Levin, Harish Mallidi, Vimal Manohar, Chunxi Liu, Courtney Napoles, Vijayaditya Peddinti, Nanyun Peng, Adam Teichert, and Xuchen Yao. You are all the best part of my time at CLSP.

Ruth Scally, Yamese Diggs and Carl Pupa, thank you for your incredible support. The CLSP runs smoothly mainly due to your efforts.

My Mom, Dad, and brother provided encouragement during the most trying times in my Ph.D., thank you, Amma, Nana, and Chait.

Finally, Nichole, I can not state what it means to have had your support through my Ph.D. You've been by my side from the first exciting phone call I received about my acceptance to the very end and everything in between. This thesis is more yours than mine. Thank you, bebbe.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Growing interest in self-directed language learning methods like Duolingo (Ahn, 2013), along with recent advances in machine translation and the widespread ease of access to a variety of texts in a large number of languages, has given rise to a number of web-based tools related to language learning, ranging from dictionary apps to more interactive tools like Alpheios (Nelson, 2007) or Lingua.ly (2013). Most of these require hand-curated lesson plans and learning activities, often with explicit instructions.

Proponents of language acquisition through extensive reading, such as Krashen (1989), argue that much of language acquisition takes place through incidental learning—when a learner is exposed to novel vocabulary or structures and must find a way to understand them in order to comprehend the text. Huckin and Coady (1999) and Elley and Mangubhai (1983) observe that incidental learning is not limited to reading in one's native language (L1) and extends to reading in a second language (L2) as well. Free reading also offers the benefit of being a “low-anxiety” (or even pleasurable) source of comprehensible input for many L2 learners (Krashen, 2003).

There is considerable evidence showing that free voluntary reading can play a role in foreign language learning. Lee, Krashen, and Gribbons (1996) studied international students in the United States and found the amount of free reading of English to be a significant predictor to judge the grammatically of complex sentences. The amount of formal study and length of residence in the United States were not strong predictors. In Stokes, Krashen, and Kartchner (1998) students learning Spanish were tested on their understanding of the subjunctive. Students were not informed of the specific focus of the test (i.e. that it was on the subjunctive). The study found that attributes such as formal study, length of residence in foreign speaking country and the student’s subjective rating of the quality of the formal study they receive failed to predict performance on the subjunctive test. The amount of free reading, however, was a strong predictor. Constantino et al. (1997) showed that free reading was a strong predictor of the performance in Test of English as a Foreign Language (TOEFL). However, they did find that other attributes such as time of residence in the United States and amount of formal study were also significant predictors of performance. Kim and Krashen (1998) go beyond self-reported reading amounts and were able to find a correlation between the performance of students in the English as a foreign language test and performance on the “author recognition” test. In the author recognition test, subjects indicate whether they recognize a name as an author among a list of names provided to them. The author recognition test has also been used in other first language studies as well such as Chinese (Lee and Krashen, 1996) Korean (Kim and Krashen, 1998) and Spanish (Rodrigo, McQuillan, and Krashen, 1996). In the case of second language acquisition, however, learning by reading already requires considerable L2 fluency, which may prove a barrier for beginners. Thus, in order to allow students to engage with the L2 language early on, educators may use texts written in simplified forms, texts specifically designed for L2 learners (e.g. texts with limited/focused vocabularies), or texts intended for young L1 learners of the given L2. "Handcrafted Books" has been proposed by (Dupuy and McQuillan, 1997) as a way to generate L2 reading material that is both accessible and enjoyable to foreign language students. Handcrafted Books are essentially articles, novels or essays written by Foreign language students at an intermediate level and subsequently corrected by a teacher. The student writers are instructed not to look-up words while writing, which helps keep the resulting material within the ability of beginner students. While this approach gives educators control over the learning material, it lacks scalability and suffers from similar issues as hand-curated lesson plans. As a result, a learner interested reading in a second language might have few choices in the type of texts made available to them.

Our proposal is to make use of "macaronic language," which offers a mixture of the learner's L1 and their target L2. The amount of mixing can vary and might depend on the learner's proficiency and desired content. Additionally, we propose automatically creating such "macaronic language," allowing our learning paradigm to scale across a wide variety of content. Our hope is that this paradigm can potentially convert any reading material into one of pedagogical value and could easily become a part of the learner's daily routine.

1.1 Macaronic Language

Why do the French only eat one egg for breakfast?
Because one egg is *un œuf*.

The term “Macaronic” traditionally refers to a mash-up of languages, often intended to be humorous. Similar to text that contains code-switching, typical macaronic texts are intended for a bilingual audience; however, they differ from code-switching as they are not governed by syntactic and pragmatic considerations. Macaronic texts are also “synthetic” in the sense that they are traditionally constructed for the purpose of humor (bilingual puns) and do not arise naturally in conversation. In this thesis, however, we use the term macaronic for bilingual texts that have been synthetically constructed for the purpose of *second language learning*. Thus, our macaronic texts do not require fluency in both languages and their usage only assumes that the student is fluent in their native language. This thesis investigates the applicability of macaronic texts as a medium for life-long second language learning. Flavors of this idea have been done before, which we cover in Chapter 2, but it is especially worth noting that the earliest published macaronic memoir we could find is “On Foreign Soil” (Zolf and Green, 2003) which is a translation of an earlier Yiddish novel (Zolf, 1945). Green’s translation begins in English with a few Yiddish words transliterated, as the novel progresses, entire transliterated Yiddish phrases are presented to the reader.

1.2 Zone of proximal development

Second language (L2) learning requires the acquisition of vocabulary as well as knowledge of the language's constructions. One of the ways in which learners become familiar with novel vocabulary and linguistic constructions is through reading. According to Krashen's Input Hypothesis (Krashen, 1989), learners acquire language through incidental learning, which occurs when learners are exposed to comprehensible input. What constitutes "comprehensible input" for a learner varies as their knowledge of the L2 increases. For example, a student in their first month of German lessons would be hard-pressed to read German novels or even front-page news, but they might understand brief descriptions of daily routines. Comprehensible input need not be completely familiar to the learner; it could include novel vocabulary items or structures, whose meanings they can glean from context. Such input falls in the "zone of proximal development" (Vygotsky, 2012), just outside of the learner's comfort zone. The related concept of "scaffolding" (Wood, Bruner, and Ross, 1976) consists of providing assistance to the learner at a level that is just sufficient enough for them to complete their task, which in our case is understanding a sentence. In this context, macaronic text can offer a flexible modality for L2 learning. The L1 portion of the text can provide scaffolding while the L2 portion, if not previously seen by the student, can provide novel vocabulary and linguistic constructions of pedagogical value. So, if we re-purpose the pun from above into a macaronic text for L2 french learners (whose L1 is English) we might construct the following with the hope that the readers can infer the meanings of **`un`** and

œuf.

> Why do the French have only **un** egg for breakfast?
> Because **un œuf** is enough.

In addition to vocabulary, macaronic scaffolding can extend to syntactic structures as well. For example,consider the following text: "**Der Student** turned in **die Hausaufgaben,** that the teacher *assigned had*." Here, German vocabulary is indicated in bold and German syntactic structures are indicated in italics. Even a reader with no knowledge of German is likely to be able to understand this sentence by using context and cognate clues. One can imagine increasing the amount of German in such sentences as the learner's vocabulary increases, thus carefully removing scaffolding (English) and keeping the learner in their zone of proximal development.

1.3 Our Goal: A Macaronic Machine Foreign Language Teacher

Our vision is to build an AI foreign-language teacher that gradually converts documents (stories, articles, etc.) written in a student's native language into the L2 the student wants to learn, by automatically replacing the L1 vocabulary, morphology, and grammar with the L2 forms (see 1.1). This "gradual conversion" involves replacing more L1 words (or phrases) with their L2 counterparts, and occurs as the leaner slowly gains L2 proficiency (say over a period of days or weeks). We don't expect significant language proficiency increase

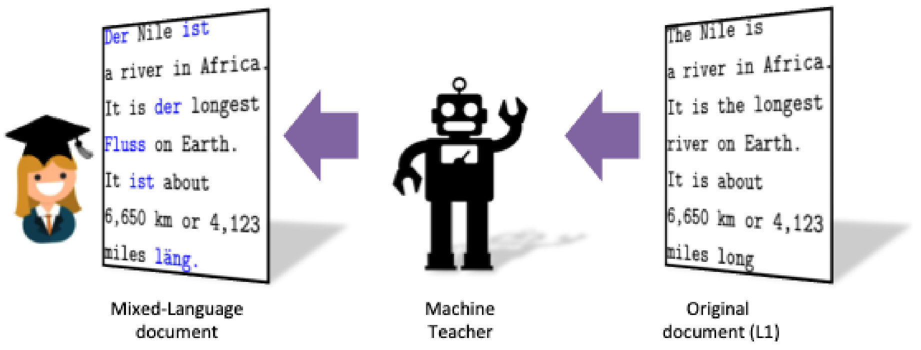


Figure 1.1: A schematic overview of our goal.

during the course of a single novel, however, we expect more conversions mainly due to the repeated appearance of key vocabulary items during the novel. Thus, L2 conversions can accumulate over the course of the novel. This AI teacher will leverage a student's inherent ability to guess the meaning of foreign words and constructions based on the context in which they appear and similarities to previously known words. We envision our technology being used *alongside* traditional classroom L2 instruction—the same instructional mix that leads parents to accept inventive spelling (Gentry, 2000), [1] in which early writers are encouraged to write in their native language without concern for correct spelling, in part so they can more fully and happily engage with the writing challenge of composing longer and more authentic texts without undue distraction (Clarke, 1988). Traditional grammar-based instruction and assessment, which uses "toy" sentences in pure L2, should provide further scaffolding for our users to acquire language by reading more advanced (but macaronic) text.

[1]Learning how to spell, like learning an L2, is a type of linguistic knowledge that is acquired after L1 fluency and largely through incidental learning (Krashen, 1993).

Automatic construction of comprehensible macaronic texts as reading material—perhaps online and personalized—would be a useful educational technology. Broadly speaking, this requires:

1. A data structure for manipulating word (or phrase) aligned bitexts so they can be rendered as macaronic sentences,

2. Modeling student's comprehension when they read these macaronic sentences (i.e. what can an L2 learner understand in a given context?), and

3. Searching over many possible candidate comprehensible inputs to find one that is most suited to the student, balancing the amount of new L2 they encounter as well as ease of reading.

In schematic Figure 1.1 the AI teacher performs the points (2) and (3) from above. While (1) defines the input (along with meta-data) required by the AI teacher to perform (2) and (3). There are several ways of realizing each of the three points above. In this thesis, the each chapter explores a subset of these three points. Chapter 3 and Chapter 4 cover points (1) and (2) using human data to construct student models. Chapter 5 describes another AI teacher that can accomplish (2) and (3) but makes some simplifying assumptions about the input data (1). Chapter 6 also details another kind of student modeling (point (2)) for a verb-inflection learning task and focuses on short-term longitudinal modeling of student knowledge as they progress through this task.

The points made above can also be recast from a Reinforcement Learning perspective.

The data structure we design defines the set of all possible actions an agent (the AI teacher) can take. The agent is acting upon the student's observations and tries to infer their comprehension of a sentence (and, more generally, their level of proficiency of L2). Thus, the student is the environment that the agent is acting upon. Finally, the search algorithm that the AI teacher employs is the policy the agent follows. This perspective also suggests that the policy could involve planning for long-term rewards (which in our framework is L2 proficiency) using policies that look-ahead into the future to make optimal decisions in the present. However, we leave planning in the macaronic space to future work. In this thesis, we limit ourselves to greedy-search techniques that do not consider long-term rewards.

A reasonable concern is whether exposure to the mixture of languages a macaronic text offers might actually harm acquisition of the "correct" version of a text written solely in the L2. To address this, our proposed interface uses color and font to mark the L1 "intrusions" into the L2 sentence, or the L2 intrusions into L1 sentence. We again draw a parallel to inventive spelling and highlight that the focus of a learner should be more on continuous engagement with L2 content even if it appears "incorrectly" within L2 texts.

1.4 Macaronic Data Structures

Several strategies can be followed to create the required data structures so that an AI teacher can manipulate and render macaronic texts. One such structure for an English-French bitext is shown in Figure 1.2. We refer to a single translation pair of a source sentence and target sentence as bitext. We use word alignments, which are edges connecting words from the source sentence to those in the target sentence in the bitext, and convert the bitext into a set of connected units. While the majority of units contain single works, note that some of the units contain phrases with internal reorderings, such as **`une telle`** or `such` **`une`**. Each unit is a bipartite graph with French words on one end and English words on the other. The AI teacher can select different cuts in each unit to render different *macaronic configurations* (See Table 1.2). Figure 1.2 shows the graph data structure and Table 1.2 lists some macaronic renderings or configurations that can be obtained from different cuts in the graph data structure. We construct these data structures automatically, but in our experiments we correct them manually as word alignments are noisy and sometime link source and target tokens that are not translation of each other. Furthermore, intermediate tokens in the macaronic such as `The Arizona` and `submit` in Figure 1.2 can not be obtained by merely using bitext and word alignments. These intermediate forms were added manually as well. We refer to the items (rows of text) in Table 1.2 as macaronic configurations. Note that macaronic configurations *are* macaronic sentences; we use the term "configuration" to denote the set of macaronic sentences from a single piece of content (i.e. from a single bitext with its supporting data structures).

An alternative strategy is to only use lexical replacements for L1 tokens. This data structure strategy assumes the availability of L2 glosses for each token in the L1 sentence. The units formed in this case are simple "one-to-one" mappings (see Figure 1.3). This data structure is less expressive in the sense that it can only render macaronic configurations

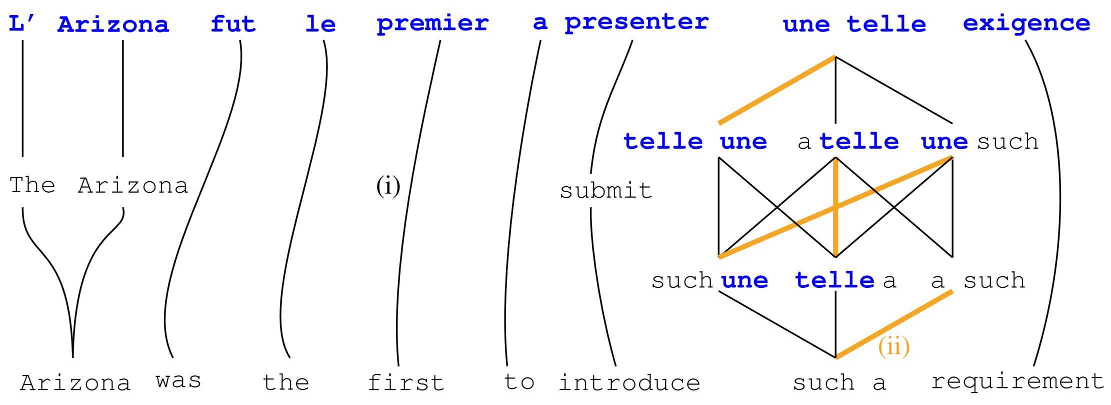

L′ Arizona fut le premier a presenter une telle exigence
telle une
a telle
une such
The Arizona
(i)
submit
such une
telle a
a such
(ii)
Arizona was the first to introduce such a requirement

Figure 1.2: Macaronic data structure extracted from word-alignments. The black lines represent edges that replace a unit, usually from one language (black for English) to another (blue for French). For example edge (i) is replaces `first` with **`premier`** and vice-versa. In some cases the edges connect two English tokens (such as `introduce` and `submit`) as in intermediate step between `introduce` and **`presenter`**. In other cases the black substitution edge define a single substitution even when there are more than one tokens in the unit. For example, `such a` is connected to `such` **`une`** via a substitution edge (ii). The orange edges perform a reordering action, for example `such a` can be transformed into `a such` by traversing an orange edge. Only two edges are marked with roman numerals for clarity.

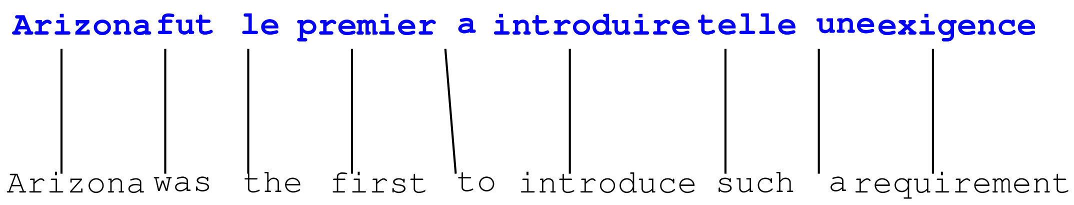

Arizona fut le premier a introduire telle une exigence
Arizona was the first to introduce such a requirement

Figure 1.3: A Simplified Macaronic data structure that only considers word replacements without any reordering of words.

L′ Arizona fut le premier a presenter une telle exigence

L′ Arizona fut le first **a presenter une telle exigence**

⋮

L′ Arizona was the first **a presenter une telle exigence**

L′ Arizona was the first **a presenter telle une** requirement

L′ Arizona was the first **a presenter telle** a requirement

L′ Arizona was the first **a presenter** such a requirement

⋮

L′ Arizona was the first to introduce such a requirement

Arizona was the first to introduce such a requirement

Table 1.1: Examples of possible macaronic configurations from the macaronic data structure depicted in Figure 1.2. This data structure supports actions that reorder phrases within the macaronic sentence, thus generating substrings like **telle une** and a such which are both not in the word-ordering of the language they are in.

Arizona fut le premier a introduire telle une exigence

Arizona **fut le premier a introduire telle une exigence**

Arizona was **le premier a introduire telle une exigence**

⋮

Arizona was **le premier a introduire** such a requirement

⋮

Arizona was the first to **introduire** such a requirement

Arizona was the first to introduce such a requirement

Table 1.2: Examples of possible macaronic configurations from the simplified macaronic data structure Figure 1.3. Note that the words (even French words) are always in the English word-order. Thus, using this data structure we can not obtain configurations that include substrings like `a such` or **`une telle`**. We envision this data structure to be useful for a native speaker of English learning french vocabulary, but it is also possible that a student seeing English words in French word order can learn about French word ordering. Then, gradually, we can replace the English words (in French word order) with French words (in French word order) forming fluent French sentences.

in the L1 word order. That is, even though it can display L2 words, these words are only displayed in their L1 orderings. This limitation simplifies the AI teacher's choice of actions but also limits the resulting content to only have value for learning foreign vocabulary. Note that even this simple structure allows for exponentially many possible configurations that the AI teacher must be able to search over.

The data structures described above are a subset of more complex structures. It is possible to construct data structures that connect subword units from the English side to an equivalent subword unit on the French side. Such a data structure could be used to render macaronic sentences and are macaronic at the word-level. Furthermore, the data structure could also support non-contiguous reorderings. We leave such data structures for future work and focus on the more straightforward local-reorderings and simple word replacement methods mainly to allow for fast search procedures.

The graph structure in Figure 1.2 could be made more complex with hyper-edges connecting a set of tokens from the English side (`such a`) to a set of tokens in the French side (**`une telle`**) of the text.

1.5 User Modeling

1.5.1 Modeling Incidental Comprehension

In order to deliver macaronic content that can be understood by a learner, we must first build a model of a learner's comprehension abilities. Would a native English speaker learning German, be able to comprehend a sentence like `The` **`Nil`** `is a` **`Fluss`** `in Africa`? Would they correctly map the German words **`Nil`** and **`Fluss`** to `Nile` and `river`? Is there a chance to incorrectly map the German words to other plausible English words, for example, `Namib` and `desert`? We approach this comprehension modeling problem by building probabilistic models that can predict if an L2 student might comprehend a macaronic sentence when they read it. One way to estimate comprehension is to ask L2 students to guess the meaning of L2 words or phrases within the macaronic sentence. A correct guess implies that there are sufficient clues, either from the L1 words in the sentence, from the L2 words, or from both, to comprehend the sentence.

We begin with a simplified version of macaronic sentences wherein only a single word (a noun) is replaced with its L2 translation. For example: `The next important` **`Klima`** `conference is in December`. We build predictive models that take the L1 context and the L2 word as input and predict what a novice L2 learner (studying German in this example) might say is the meaning for the novel word **`Klima`**.

Next, we address the case in which multiple words in the macaronic sentence are converted into their L2 form. Consider the earlier example: `The` **`Nil`** `is a` **`Fluss`** `in Africa`. In such cases, we need to jointly predict the meaning a student would assign to all the L2 words. This is necessary because a student's interpretation of one word will influence how they interpret the remaining L2 words in the sentence. For example, if the student assigns the meaning `Nile` to **`Nil`**, this might influence them to guess that **`Fluss`** should be interpreted as `River`. Alternatively, if they interpret **`Fluss`** as `forest`, they might then guess that **`Nil`** is the name of a forest in Africa. In other words, there is a cyclical dependency between the guesses that a student makes. Details of our proposed models for capturing incidental learning are discussed in Chapter 3. Code used for our experiments is available at `https://github.com/arendu/MacaronicUserModeling`.

1.5.2 Proxy Models for Incidental Comprehension

One way to build models of human incidental comprehension is to collect data from humans. In the previous section (with details in Chapter 3) we require humans to read a candidate macaronic text and provide feedback as to whether the text was comprehensible and whether the L2 words in the text were understood. Using this feedback (i.e. labeled data) we build a model of a "generic" student. Collecting this labeled data from student annotators is expensive. Not only from a data collection point of view but also for students, as they would have to give feedback on candidate macaronic texts generated by an untrained machine teacher.

As an alternative to collecting labeled data in this way, we investigate using cloze language models as a proxy for models of incidental comprehension. A cloze language model can be trained with easily available L1 corpora from any domain (that potentially is of interest to a student). We can refine the cloze language model with real student supervision in an online fashion as a student interacts with macaronic text generated by the AI teacher. In other words, this cloze language model can be personalized to individual students by looking at what they are able to understand and making updates to the model accordingly. Essentially, the cloze language model allows us to bootstrap the macaronic learning setup without expensive data collection overhead. Details of our use of proxy user models are described in Chapter 5.

1.5.3 Knowledge Tracing

Apart from modeling incidental comprehension, an AI teacher would benefit from modeling how a student might update their knowledge based on different pedagogical stimuli, which in our case take the form of different macaronic sentences. Furthermore, in the case of "pop quizzes" (see Chapter 4), the student may receive explicit feedback for their guesses. Ideally, such explicit feedback would also cause the student to update their knowledge. Here an "update" could entail learning (adding to their knowledge) or forgetting (removing from their knowledge). The longitudinal tracking of knowledge a learner has as they proceed through a sequence of learning activities is referred to as "knowledge tracing" (Corbett and Anderson, 1994). We study a feature-rich knowledge tracing method that captures a student's acquisition and retention of knowledge during a foreign language phrase-learning task. Note that in we deviate from our macaronic paradigm for this task and focus on short-term longitudinal modeling of a students knowledge in a phrase-learning task that teaches L2 verb inflections. In this study, we use flash-cards instead of macaronic texts, mainly because of the ease in obtaining longitudinal participation in user studies. We model a student's behavior as making predictions under a log-linear model, and adopt a neural gating mechanism to model how the student updates their log-linear parameters in response to feedback. The gating mechanism allows the model to learn complex patterns of retention and acquisition for each feature, while the log-linear parameterization results in an interpretable knowledge state.

We collect human data and evaluate several versions of the model. We hypothesize that human data collection for verb inflection is not as problematic as the full macaronic setting as there are only a handful (a few dozen inflectional forms even for morphological rich languages) inflectional forms to master. Secondly, we do not have to subject student annotators to stimuli has been generated by untrained AI teachers, making the data collection process a beneficial for the student annotators as well. Details of our proposal for knowledge tracing are presented in Chapter 6. The code and data for our experiments is available at `https://github.com/arendu/vocab-trainer-experiments`.

1.6 Searching in Macaronic Configurations

In §1.5.1 and §1.5.2 we introduce the notion of modeling human comprehension in macaronic settings (either using actual human data or by proxy models). However, our AI teacher still has the difficult task of deciding which specific macaronic sentence to generate for a student reach some new piece of text. Even in the case of the simplified lexical macaronic data structure, where each L1 word maps to a single L2 word and there are no phrase re-orderings, there are exponentially many possible macaronic configurations that can be generated. The AI teacher must decide on a particular configuration that will be rendered or displayed to a reader by *searching* over (some subset) of the possible configurations and picking a *good* candidate. We propose greedy and best-first heuristics to tackle this search problem and also design a scoring function that guides the search process to find good candidates to display. While simple, we find the greedy and best-first heuristics approach offers an effective strategy for finding a macaronic configuration with a low computational footprint. In order to continuously update our language model in response to a student's real-time interactions, the speed of our search is a critical factor. The greedy best-first approach offers a solution that prioritizes speed. To measure the goodness of each search state, our scoring function compares the initially trained L1 word embeddings with the incrementally trained L2 word embeddings and assigns a score reflecting the proximity of the L2 word embeddings to their L1 counterparts. We conduct intrinsic experiments using this proposed scoring function to determine viable search heuristics. Further details on our scoring function and heuristic search are presented in Chapter 5. The code for our experiments are available here `https://github.com/arendu/Mixed-Lang-Models`.

1.7 Interaction Design

While macaronic texts can be "consumed" as static documents, the prevalence of e-readers and web-based reading interfaces allows us to take advantage of a more interactive reading and learning experience. We propose a user interface that can be helpful to a human learner, while also enabling the AI teacher to adapt itself towards a specific learner. Specifically, when a macaronic document is presented to the student, we provide functionality that allows a student to click on L2 words or phrases in order to reveal their L1 translation. This helps the reader to progress if they are struggling with how to interpret a given word or phrase. Furthermore, we can log a student's clicking interactions and use them as feedback for our machine teacher. Many clicks within a sentence might indicate that the macaronic configuration of the sentence was beyond the learner's reading abilities, and the teacher can update its models accordingly. Apart from this, there is also the option of the teacher not revealing the L1 translation, but rather prompting the student to first type in a guess for the meaning of the word. This process helps to disambiguate between those students who click just for confirmation of their knowledge and those who genuinely don't know the word at all. By looking at what a student has typed, we can determine how close the student's knowledge is to actual understanding of the word or phrase. Details of our Interaction Design proposal are described in Chapter 4, but we leave the construction of an interactive system which iteratively refines its model for future work.

1.8 Publications

This thesis is mainly the culmination of the following six publications (including one demonstration track publication and one workshop publication):

1. Analyzing Learner Understanding of Novel L2 Vocabulary

   Rebecca Knowles, Adithya Renduchintala, Philipp Koehn and Jason Eisner, Conference on Computational Natural Language Learning (CoNLL), 2016.

2. Creating Interactive Macaronic Interfaces for Language Learning.

   Adithya Renduchintala, Rebecca Knowles, Philipp Koehn and Jason Eisner, System Description, Annual Meeting of the Association for Computational Linguistics (ACL), 2016.

3. User Modeling in Language Learning with Macaronic Texts.

   Adithya Renduchintala, Rebecca Knowles, Philipp Koehn and Jason Eisner, Annual Meeting of the Association for Computational Linguistics (ACL), 2016.

4. Knowledge Tracing in Sequential Learning of Inflected Vocabulary

   Adithya Renduchintala, Philipp Koehn and Jason Eisner, Conference on Computational Natural Language Learning (CoNLL), 2017.

5. Simple Construction of Mixed-Language Texts for Vocabulary Learning

   Adithya Renduchintala, Philipp Koehn and Jason Eisner. Annual Meeting of the Association for Computational Linguistics (ACL) Workshop on Innovative Use of NLP for Building Educational Applications, 2019

6. Spelling-Aware Construction of Macaronic Texts for Teaching Foreign-Language Vocabulary

   Adithya Renduchintala, Philipp Koehn and Jason Eisner. Empirical Methods in Natural Language Processing (EMNLP), 2019

Chapter 2

Related Work

Early adoption of Natural Language Processing (NLP) and speech technology in education was mainly focused on *Summative Assessment*, where a student's writing, speaking, or reading is analyzed by an NLP system. Such systems, essentially assigns a score to the input. Prominent examples include Heilman and Madnani (2012), Burstein, Tetreault, and Madnani (2013) and Madnani et al. (2012). More recently, NLP systems have also been used to provide *Formative Assessment*. Here, the system provides feedback in a form that a student can act upon and improve their abilities. Formative Assessment has also been studied in other areas of education such as Math and Science. In language education, Formative Assessment may take the form of giving a student qualitative feedback on a particular part of the student's essay. For example, suggesting a different phrasing. Such systems fall along the lines of intelligent and adaptive tutoring solutions designed to improve learning outcomes. Recent research such as Zhang et al. (2019) and commercial solutions such as

Grammarly (2009) are expanding the role of NLP in formative feedback and assessment. An overview of NLP-based work in the education sphere can be found in Litman (2016).

There is also lines of work that are not within the definitions of Summative or Formative assessment. For example, practice question generation is the task of creating pedagogically useful questions for a student allowing them to practice without the need of a human teacher. (Du, Shao, and Cardie, 2017) and (Heilman and Smith, 2010) is one of the newer research which focused on question generation for reading comprehension. Prior to that (Mitkov and Ha, 2003) used rule-based methods to generate questions.

There has also been NLP work specific to second language acquisition, such as Özbal, Pighin, and Strapparava (2014), where the focus has been to build a system to help learners retain new vocabulary. As previously mentioned, mobile and web-based apps for second language learning such as like Duolingo (Ahn, 2013) are popular among learners as they allow self-paced study and hand-crafted curricula. While most of these apps have "gamified" the learner's experience, they still demand dedicated time from the learner.

The process of generating training data for Machine Translation systems also have potential to be language learning tools. Hermjakob et al. (2018) is a tool that allows human annotators to generate translations (target references) from source sentences in a language they do not read. The tool presents a source sentence (for which a reference target is required) to a human annotator in romanized form along with phrasal glosses of the romanized-source text using a look up table. Hermjakob et al. (2018) observed that by simply allowing the annotators to translate source sentences (with their supporting interface) the annotators learned vocabulary and syntax over time. A similar observation was made in Hu et al. (2011) and Hu, Bederson, and Resnik (2010) who also built tools to obtain reference translations from human annotators who do not read the source language.

The work in this thesis, however, seeks to build a framework based on incidental learning when reading macaronic passages from documents such as news articles, stories, or books. Our approach does not rely on hand-made curricula, does not present explicit instructions, and (hopefully) can be used by foreign language students in the daily course of their lives.[1] Our goal is that this would encourage continued engagement, leading to "life-long learning." This notion of incidental learning has been explored in previous work as well. Chen et al. (2015) create a web-based plugin that can expose learners to new foreign language vocabulary while reading news articles. They use a dictionary to show Chinese translations of English words when the learner clicks on an English word in the document (their prototype targets native English speakers learning Chinese). When a particular English word is clicked, the learner is shown that word's Chinese translation. Once the application records the click, it then determines whether the user has reached a certain threshold for that word and automatically replaces future occurrences of the English with its Chinese translation. The learner can also click on the Chinese translation, at which point they receive a multiple choice question asking them to identify the correct English translation. While they don't use surrounding context when determining which words to teach, the learner has access to context in the form of the rest of the document when making multiple-choice

[1]Variations of our framework could (if the learner chooses) provide explicit instructions and make the task more learning focused at the expense of casual reading.

guesses. Our work is also related to Labutov and Lipson (2014), which also tries to leverage incidental learning using mixed L1 and L2 languages. Whereas their work uses surprisal to choose contexts in which to insert L2 vocabulary, we consider both context features and other factors such as cognate features. Further, we collect data that gives direct evidence of the user's understanding of words by asking them to provide English guesses, rather than indirectly, via questions about sentence validity. The latter indirect approach runs the risk of overestimating the student's knowledge of a word; for instance, a student may have only learned other linguistic information about a word, such as whether it is animate or inanimate, rather than learning its exact meaning. In addition, we are not only interested in whether a mixed L1 and L2 sentence is comprehensible; we are also interested in determining a distribution over the learner's belief state for each word in the sentence. We do this in an engaging, game-like setting, which provides the user with hints when the task is too difficult for them to complete.

Incidental learning can be viewed as a kind of "fast mapping," a process by which children are able to map novel words to their meaning with relatively few exposures (Carey and Bartlett, 1978). Fast mapping is usually studied as a mapping between a novel word and some concept in the immediate scene. Carey and Bartlett (1978) studied whether 3 year old children can map a novel word, for example"chromium," to an unfamiliar color (olive-green) using a "referent-selection" task, which required a subject to retrieve the correct unfamiliar object from a set of objects. Children were given instructions such as *bring the chromium tray, not the blue one*. It was observed that children were able to perform such mappings quickly. Subsequently, Alishahi, Fazly, and Stevenson (2008) constructed a probabilistic model and were able to tune this model to fit the empirical observations of previous fast mapping experiments. The model experiences a sequence of utterances U_t in scenes S_t. Each utterance contains words $w \in U_t$, and the "scene" contains a set of concepts $m \in S_t$. With each U_t, S_t pair, the model parameters $p(m \mid w)$, were updated using an online EM update. At the end of a sequence of U_t, S_t $t \in 1, \ldots T$ pairs, the final model parameters were used to simulate "referent-selection" and retention tasks. We can view the student's task (in our macaronic setting) as an instance of cross-lingual structured fast-mapping, where an utterance is a macaronic sentence and the student is trying to map novel foreign words to words in their native language.

We are also encouraged by recent commercial applications that use a mixed language framework for language education. Swych (2015) claims to automatically generate mixed language documents while OneThirdStories (2018) creates human generated mixed-language stories that begin in one language and gradually incorporate more and more vocabulary and syntax for a second language. Such new developments indicate that there is both space and demand in the language learning community for further exploration of language learning via mixed languages modalities.

In the following chapters, we detail our experiments with modeling user comprehension in mixed language situations, as well as proposing a simplified process for generating macaronic text without initial human student intervention. Finally, we also model ways to track student knowledge as they progress through a language learning activity.

Chapter 3

Modeling Incidental Learning

This chapter details our work on constructing predictive models of human incidental learning. Concretely, we cast the modeling task as a prediction task in which the model predicts if a human student can *guess* the meaning of a novel L2 word when it appears with the surrounding L1 context. Apart from giving the model access to the context, we also provide the model with features from the novel word itself, such as spelling and pronunciation features, as these would all aid a human student in their guess of the novel word's meaning. Recall that this model is essentially a model of the environment, taking the reinforcement learning perspective of our macaronic learning framework (from §1.5.1), that the agent (the AI teacher) acts upon.

3.1 Foreign Words in Isolation

We first study a constrained setting where we present novice learners with new L2 words inserted in sentences otherwise written in their L1. In this setting only a single L2 word is present in each sentence. While this is not the only possible setting for incidental acquisition (§3.2 discusses the same task for the "full" macaronic setting), this experimental design allows us to assume that all subjects understand the full context, without needing to assess how much L2 they previously understood. We also present novice learners with the same novel words out of context. This allows us to study how "cognateness" and context interact, in a well-controlled setting. We hypothesize that cognates or very common words may be easy to translate without context, while contextual clues may be needed to make other words guessable.

In the initial experiments we present here, we focus on the language pair of English L1 and German L2, selecting Mechanical Turk users who self-identify as fluent English speakers with minimal exposure to German. We confine ourselves to novel nouns, as we expect that the relative lack of morphological inflection in nouns in both languages[1] will produce less noisy results than verbs, for example, which naive users might incorrectly inflect in their (English) responses.

Even more experienced L2 readers will encounter novel words when reading L2 text. Their ability to decipher a novel word is known to depend on both their understanding

[1]Specifically, German nouns are marked for number but only rarely for case.

of the surrounding context words (Huckin and Coady, 1999) and the cognateness of the novel word. We seek to evaluate this quantitatively and qualitatively in three "extreme" cases (no context, no cognate information, full context with cognate information). In doing so, we are able to see how learners might react differently to novel words based on their understanding of the context. This can serve as a well-controlled proxy for other incidental learning settings, including reading a language that the learner knows well and encountering novel words, encountering novel vocabulary items in isolation (for example on a vocabulary list), or learner-driven learning tools such as ones involving the reading of macaronic text.

3.1.1 Data Collection and Preparation

We use data from `NachrichtenLeicht.de`, a source of news articles in simple German (Leichte Sprache, "easy language") (Deutschlandfunk, 2016). Simple German is intended for readers with cognitive impairments and/or whose first language is not German. It follows several guidelines, such as short sentences, simple sentence structure, active voice, hyphenation of compound nouns (which are common in German), and use of prepositions instead of the genitive case (Wikipedia, 2016).

We chose 188 German sentences and manually translated them into English. In each sentence, we selected a single German noun whose translation is a single English noun. This yields a triple of (German noun, English noun, English translation of the context). Each German noun/English noun pair appears only once[2] and each English sentence is unique, for

[2]The English word may appear in other sentences, but never in the sentence in which its German counterpart appears.

Task	**Text Presented to Learner**	**Correct Answer**
word	*Klima*	climate
cloze	The next important _____ conference is in December.	climate
combined	The next important *Klima* conference is in December.	climate

Table 3.1: Three tasks derived from the same German sentence.

a total of 188 triples. Sentences ranged in length from 5 tokens to 28 tokens, with a mean of 11.47 tokens (median 11). Due to the short length of the sentences, in many cases there was only one possible pair of aligned German and English nouns (both of which were single words rather than noun phrases). In the cases where there were multiple, the translator chose one that had not yet been chosen, and attempted to ensure a wide range of clear cognates to non-cognates, as well as a range of how clear the context made the word.

As an outside resource for training language models and other resources, we chose to use Simple English Wikipedia (Wikimedia Foundation, 2016). It contains 767,826 sentences, covers a similar set of topics to the `NachrichtenLeicht.de` data, and uses simple sentence structure. The sentence lengths are also comparable, with a mean of 17.6 tokens and a median of 16 tokens. This makes it well-matched for our task.

Our main goal is to examine students' ability to understand novel L2 words. To better separate the effects of context and cognate status and general familiarity with the nouns, we assess subjects on the three tasks illustrated in Table 3.1:

1. *word*: Subjects are presented with a single German word out of context, and are asked

to provide their best guess for the translation.

2. *cloze*: A single noun is deleted from a sentence and subjects are asked to fill in the blank.

3. *combined*: Subjects are asked to provide their best-guess translation for a single German noun in the context of an English sentence. This is identical to the cloze task, except that the German noun replaces the blank.

We used Amazon Mechanical Turk (henceforth MTurk), a crowdsourcing platform, to recruit subjects and collect their responses to our tasks. Tasks on MTurk are referred to as HITs (Human Intelligence Tasks). In order to qualify for our tasks, subjects completed short surveys on their language skills. They were asked to rate their language proficiency in four languages (English, Spanish, German, and French) on a scale from "None" to "Fluent." The intermediate options were "Up to 1 year of study (or equivalent)" and "More than 1 year of study (or equivalent)".[3] Only subjects who indicated that they were fluent in English but indicated "None" for German experience were permitted to complete the tasks.

Additional stratification of users into groups is described in the subsection below. The HITs were presented to subjects in a somewhat randomized order (as per MTurk standard setup).

Data Collection Protocol: In this setup, each subject may be asked to complete instances of all three tasks. However, the subject is shown at most one task instance that was

[3]Subjects were instructed to list themselves as having experience equivalent to language instruction even if they had not studied in a classroom if they had been exposed to the language by living in a place that it was spoken, playing online language-learning games, or other such experiences.

derived from a given data triple (for example, at most one line from Table 3.1).[4] Subjects were paid between \$0.05 and \$0.08 per HIT, where a HIT consists of 5 instances of the same task. Each HIT was completed by 9 unique subjects. Subjects voluntarily completed from 5 to 90 task instances (1–18 hits), with a median of 25 instances (5 HITs). HITs took subjects a median of 80.5 seconds according to the MTurk output timing. Each triple gives rise to one cloze, one word, and one combined task. For each of those tasks, 9 users make guesses, for a total of 27 guesses per triple. Data was preprocessed to lowercase all guesses and to correct obvious typos.[5] Users made 1863 unique guesses (types across all tasks). Of these, 142 were determined to be errors of some sort; 79 were correctable spelling errors, 54 were multiple-word phrases rather than single words, 8 were German words, and 1 was an ambiguous spelling error. In our experiments, we treat all of the uncorrectable errors as out of vocabulary tokens, for which we cannot compute features (such as edit distance, etc.).

Data Splits: After collecting data on all triples from our subjects, we split the dataset for purposes of predictive modeling. We randomly partitioned the triples into a training set (112 triples), a development set (38 triples), and a test set (38 triples). Note that the same partition by triples was used across all tasks. As a result, a German noun/English noun pair that appears in test data is genuinely unseen—it did not appear in the training data for *any* task.

[4]Each triple gives rise to an instance of each task. Subjects who completed one of these instances were prevented from completing the other two by being assigned an additional MTurk "qualification"—in this case, a disqualification.

[5]All guesses that were flagged by spell-check were manually checked to see if they constituted typos (e.g., "langauges" for "languages") or spelling errors (e.g., "speach" for "speech") with clear corrections.

3.1.2 Modeling Subject Guesses

In an educational technology context, such as a tool for learning vocabulary, we would like a way to compute the difficulty of examples automatically, in order to present learners with examples with appropriate level of difficulty. For such an application, it would be useful to know not only whether the learner is likely to correctly guess the vocabulary item, but also whether their incorrect guesses are "close enough" to allow the user to understand the sentence and proceed with reading. We seek to build models that can predict a subject's likely guesses and their probabilities, given the context with which they have been presented.

We use a small set of features (described below) to characterize subjects' guesses and build predictive models of what a subject is likely to guess. Feature functions can jointly examine the input presented to the subject and candidate guesses.

We evaluate the models both in terms of how well they predict subject guesses, as well as how well they perform on the simpler subtask of modeling guessability. We define *guessability* for a word in context to be how easy that word is for a subject to guess, given the context. In practice, we estimated guessability as the proportion of subjects that exactly guessed the word (i.e., the reference English translation).

3.1.2.1 Features Used

When our formula for computing a feature draws on parameters that are estimated from corpora, we used the Simple English Wikipedia data and GloVe vectors (Pennington, Socher, and Manning, 2014). Our features are functions whose arguments are the candidate guess and the triple (of German noun, English noun, and English sentence). They are divided into three categories based on which portions of the triple they consider:

Generic Features: These features are independent of subject input, and could be useful regardless of whether the subject made their guess in the word, cloze, or combined task.

1. `Candidate==Correct Answer` This feature fires when the candidate is equal to the correct answer.
2. `Candidate==OOV` This is used when the candidate guess is not a valid English word (for example, multiple words or an incomprehensible typo), in which case no other features about the candidate are extracted.
3. `Length` We compute the number of characters in the correct answer.
4. `Embedding` Cosine distance between embedding of candidate and embedding of correct answer. For the embeddings, we use the 300-dimensional GloVe vectors from the 6B-token dataset.
5. `Log Unigram Frequency` of candidate in the Simple English Wikipedia corpus.
6. `Levenshtein Distance` between candidate and correct answer.
7. `Sound Edit Distance` Levenshtein Distance between phonetic representations of candidate and correct answer, as given by Metaphone (Philips, 1990).[6]

[6]When several variations were available for a particular feature, such as which phonetic representation to use or whether or not to normalize, the version we selected for our studies (and described here) is the version that correlated most strongly with guessability on training data.

8. `LCS` Length of longest common substring between candidate and correct answer, normalized.

9. `Normalized Trigram Overlap` count of character trigrams (types) that match between the candidate and correct answer, normalized by the maximum possible matches.

Word Features: These features are dependent on the German word, and should thus only be useful in the word and combined tasks. The second half of the generic features (from Levenshtein Distance through Normalized Trigram Overlap) are also computed between the candidate and the German word and are used as measures of cognateness. The use of Metaphone (which is intended to predict the pronunciation of English words) is appropriate to use for German words in this case, as it corresponds to the assumption that our learners do not yet have accurate representations of German pronunciation and may be applying English pronunciation rules to German. **Cloze Features**: These features are dependent on the surrounding English context, and should thus only be useful in the cloze and combined tasks.

1. `LM Score` of candidate in context, using a 5-gram language model built using KenLM (Heafield et al., 2013) and a neural RNNLM (Mikolov et al., 2011).[7] We compute three different features for each language model, a raw LM score, a sentence-length-normalized LM score, and the difference between the LM score with the correct answer in the sentence and the LM score with the candidate in its place.

[7]We use the Faster-RNNLM toolkit available at `https://github.com/yandex/faster-rnnlm`.

2. `PMI` Maximum pointwise mutual information between any word in the context and the candidate.

3. `Left Bigram Collocations` These are bigram association measures between the candidate's neighbor(s) to the left and the candidate (Church and Hanks, 1990). We include a version that just examines the neighbor directly to the left (which we'd expect to do well in collocations like "San Francisco") as well as a version that returns the maximum score over a window of five, which behaves like an asymmetric version of PMI.

4. `Embeddings` Minimum cosine distance between embeddings of any word in the context and the candidate.

Intuitively, we expect it to be easiest to guess the correct word in the combined task, followed by the cloze task, followed by the L2 word with no context.[8] As shown in Figure 3.1, this is borne out in our data.

In Table 3.2 we show Spearman correlations between several features and the guessability of the word (given a word, cloze, or combined context). The first two features (log unigram probability and length of the correct solution) in Table 3.2 belong to the generic category of features. We expect that learners may have an easier time guessing short or common words (for instance, it may be easier to guess "cat" than "trilobite") and we do observe such correlations.

[8]All plots/values in the remainder of this subsection are computed only over the training data unless otherwise noted.

Feature	**Spearman's Correlation w/ Guessability**			
	Word	*Cloze*	*Combined*	*All*
Log Unigram Frequency	0.310*	0.262*	0.279*	0.255*
English Length	-0.397*	-0.392*	-0.357*	-0.344*
Sound Edit Distance (German + Answer)	-0.633*	n/a	-0.575*	-0.409*
Levenshtein Distance (German + Answer)	-0.606*	n/a	-0.560*	-0.395*
Max PMI (Answer + Context)	n/a	0.480*	0.376*	0.306*
Max Left Bigram Collocations (Answer + Window=5)	n/a	0.474*	0.186	0.238*
Max Right Bigram Collocations (Answer + Window=5)	n/a	0.119	0.064	0.038

Table 3.2: Correlations between selected feature values and answer guessability, computed on training data (starred correlations significant at $p < 0.01$. Unavailable features are represented by "n/a" (for example, since the German word is not observed in the cloze task, its edit distance to the correct solution is unavailable). Due to the format of our triples, it is still possible to test whether these unavailable features influence the subject's guess: in almost all cases they indeed do not appear to, since the correlation with guessability is low (absolute value < 0.15) and not statistically significant even at the $p < 0.05$ level.

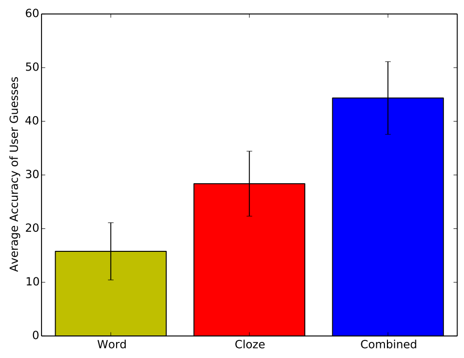


Figure 3.1: Average guessability by context type, computed on 112 triples (from the training data). Error bars show 95%-confidence intervals for the mean, under bootstrap resampling of the 112 triples (we use BCa intervals). Mean accuracy increases significantly from each task to the next (same test on difference of means, $p < 0.01$).

In the middle of the table, we can see how, despite the word task being most difficult on average, there are cases such as *Gitarrist* (guitarist), where cognateness allows all or nearly all learners to guess the meaning of the word with no context. The correlation between guessability and Sound Edit Distance as well Levenshtein Distance demonstrate their usefulness as proxies for cognateness. The other word features described earlier also show strong correlation with guessability in the word and combined tasks.

Similarly, in some cloze tasks, strong collocations or context clues, as in the case of "His plane landed at the _____," make it easy to guess the correct solution (airport). Arguably, even the number of blank words to fill are a clue to complete this sentence, but we do not model this in our study. We would expect, for instance, a high PMI between "plane" and "airport", and we see this reflected in the correlation between high PMI and guessability.

The final two lines of the table examine an interesting quirk of bigram association measures. We see that Left Bigram Collocations with a window of 5 (that is, where the feature returns the maximum collocation score between a word in the window to the left of the word to be guessed) shows reasonable correlation with guessability. Right bigram collocations, on the other hand, do not appear to correlate. This suggests that the subjects focus more on the words preceding the blank when formulating their guess (which makes sense as they read left-to-right). Due to its poor performance, we do not include Right Bigram Collocations in our later experiments.

We expect that learners who see only the word will make guesses that lean heavily on cognateness (for example, incorrectly guessing “Austria” for “Ausland”), while learners who see the cloze task will choose words that make sense semantically (eg. incorrectly guessing “tornado” in the sentence “The ______ destroyed many houses and uprooted many trees”). Guesses for the combined task may fall somewhere between these two, as the learner takes advantage of both sources of information. Here we focus on incorrect guesses (to control for the differences in task difficulty).

For example, in Figure 3.2, we see that guesses (made by our human subjects) in the word task have higher average Normalized Character Trigram Overlap than guesses in the cloze task, with the combined task in between.

This pattern of the combined task falling between the word and cloze task is consistent across most features examined.

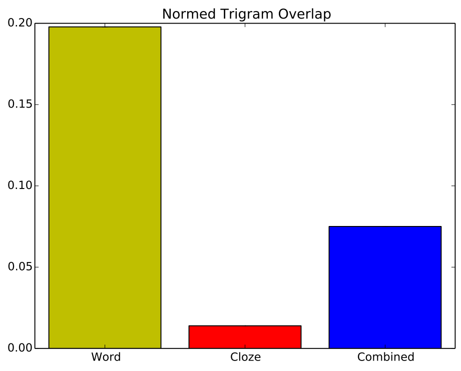


Figure 3.2: Average Normalized Character Trigram Overlap between guesses and the German word.

3.1.3 Model

The correlations in the previous subsection support our intuitions about how to model subject behavior in terms of cognateness and context. Of course, we expect that rather than basing their guesses on a single feature, subjects are performing cue combination, balancing multiple cognate and context clues (whenever they are available). Following that to its natural conclusion, we choose a model that also allows for cue combination in order to model subject guesses.

We use log-linear models to model subject guesses as probability distributions over the vocabulary V, as seen in Equations 3.1 and 3.2.

$$p(y|x) = \frac{\exp(w \cdot f(x,y))}{\sum_{y' \in V} \exp(w \cdot f(x,y'))} \tag{3.1}$$

$$w \cdot f(x,y) = \sum_k w_k f_k(x,y) \tag{3.2}$$

We use a 5000-word vocabulary, containing all complete English vocabulary from the triples and user guesses, padded with frequent words from the Simple English Wikipedia dataset.

Given the context x that the subject was shown (word, cloze, or combined), $p(y|x)$ represents the probability that a subject would guess the vocabulary item $y \in V$. The model learns weights w_k for each feature $f_k(x, y)$. We train the model using MegaM (Daumé III, 2004) via the NLTK interface.

An example feature function is shown in Equation 3.3.

$$f_k(x, y) = \begin{cases} 1, & \text{if Correct Answer == y} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

In order to best leverage the cloze features (shared across the cloze and combined tasks), the word features (shared across the word and combined task) and the generic features (shared across all tasks), we take the domain adaptation approach used in (Daumé III, 2007). In this approach, instead of a single feature for Levenshtein distance between a German word and a candidate guess, we instead have three copies of this feature, one that fires only when the subject is presented with the word task, one that fires when the subject is presented with the combined task, and one which fires in either of those situations (note that since a subject who sees the cloze task does not see the German word, we omit such a version of the feature). This allows us to learn different weights for different versions of the same features. For example, this allows the model to learn high weight for cognateness features in the word and combined settings, without being influenced to learn a low weight on it by the cloze setting.

3.1.3.1 Evaluating the Models

We evaluate the models in several ways: using conditional cross-entropy, by computing mean reciprocal rank, and computing correlation with guessability.

The empirical distribution for a given context x is calculated from all $count(\cdot|x)$ learner guesses for that context, with $p(g|x) = \frac{count(g|x)}{count(\cdot|x)}$ (where $count(g|x)$ is the number of learners who guessed g in the context x).

The *conditional cross-entropy* is defined to be the mean negative log probability over all test task instances (pairs of subject guesses g and contexts x), $\frac{1}{N}\sum_{i=0}^{N} - \log_2 p(g_i \mid x_i)$.

The *mean reciprocal rank* is computed after ranking all vocabulary words (in each context) by the probability assigned to them by the model, calculating the reciprocal rank of the each subject guess g_i, and then averaging this across all contexts x in the set X of all contexts, as shown in Equation 3.4.

$$MRR = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\text{rank}(g_i|x_i)} \tag{3.4}$$

In order to compute *correlation with guessability*, we use Spearman's rho to check the correlation between guessability and the probability assigned by the model to the correct answer.

3.1.4 Results and Analysis

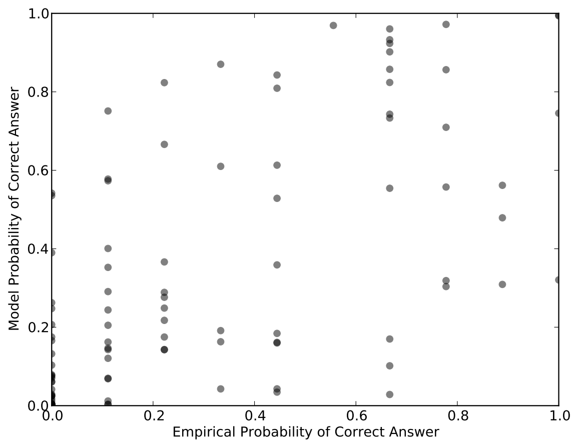


Figure 3.3: Correlation between empirically observed probability of the correct answer (i.e. the proportion of human subject guesses that were correct) and model probability assigned to the correct answer across all tasks in the test set. Spearman's correlation of 0.725.

In Table 3.3 we show model results over several feature sets. None of the feature sets (generic features, word features, or cloze features) can perform well individually on the full test set, which contains word, cloze, and combined tasks. Once combined, they perform better on all metrics.

Additionally, using domain adaptation improves performance. Manually examining the best model's most informative features, we see, for example, that edit distance features are ranked highly in their word-only or word-combined versions, while the combined-only version of those features is less informative. This reflects our earlier observation that edit distance features are highly correlated with guessability in the word task, and slightly less so

Features	**Cross-Entropy**	**MRR**	**Correlation**
LCS (Candidate + Answer)	10.72	0.067	0.346*
All Generic Features	8.643	0.309	0.168
Sound Edit Distance (Candidate + German Word)	10.847	0.081	0.494*
All Word Features	10.018	0.187	0.570*
LM Difference	11.214	0.051	0.398*
All Cloze Features	10.008	0.105	0.351*
Generic + Word	7.651	0.369	0.585*
Generic + Cloze	8.075	0.320	0.264*
Word + Cloze	8.369	0.227	0.706*
All Features (No Domain Adapt.)	7.344	0.338	0.702*
All Features + Domain Adapt.	**7.134**	**0.382**	**0.725***

Table 3.3: Feature ablation. The single highest-correlating feature (on dev set) from each feature group is shown, followed by the entire feature group. All versions with more than one feature include a feature for the OOV guess. In the correlation column, p-values < 0.01 are marked with an asterisk.

Context Observed	**Guess**	**Truth**	**Hypothesized Explanation**
Helfer	cow	helpers	False Friend: Helfer $\rightarrow$ Heifer $\rightarrow$ Cow
Journalisten	reporter	journalists	Synonym and incorrect number.
The *Lage* is too dangerous.	lake	location	Influenced by spelling and context.

Table 3.4: Examples of incorrect guesses and potential sources of confusion.

(though still relevant) in the combined task. We show in Figure 3.3 that the model probability assigned to the correct guess correlates strongly with the probability of learners correctly guessing it.

Annotated Guesses: To take a fine-grained look at guesses, we broke down subject guesses into several categories.

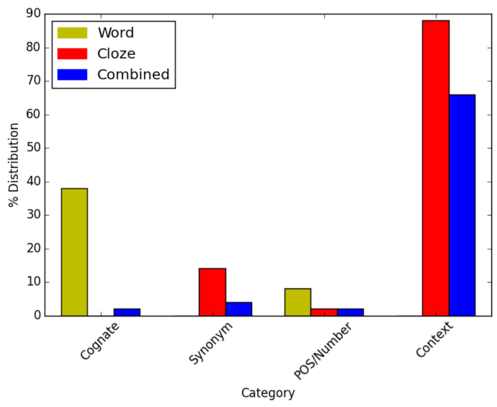


Figure 3.4: Percent of examples labeled with each label by a majority of annotators (may sum to more than 100%, as multiple labels were allowed).

We had 4 annotators (fluent English speakers, but non-experts) label 50 incorrect subject guesses from each task, sampled randomly from the spell-corrected incorrect guesses in the training data, with the following labels indicating why the annotator thought the subject made the (incorrect) guess they did, given the context that the subject saw: false friend/cognate/spelling bias (learner appears to have been influenced by the spelling of the German word), synonym (learner guess is a synonym or near-synonym to the correct answer), incorrect number/POS (correct noun with incorrect number or incorrect POS), and context influence (a word that makes sense in the cloze/combo context but is not correct). Examples of the range of ways in which errors can manifest are shown in Table 3.4. Annotators made a binary judgment for each of these labels. Inter-annotator agreement was substantial, with Fleiss's kappa of 0.654. Guesses were given a label only if the majority of annotators agreed.

In Figure 3.4, we can make several observations about subject behavior. First, the labels for the combined and cloze tasks tend to be more similar to one another, and quite different from the word task labels. In particular, in the majority of cases, subjects completing cloze and combo tasks choose words that fit the context they've observed, while spelling influence in the word task doesn't appear to be quite as strong. Even if the subjects in the cloze and combined tasks make errors, they choose words that still make sense in context more than 50% of the time, while spelling doesn't exert an equally strong influence in the word task.

Our model also makes predictions that look plausibly like those made by the human subjects. For example, given the context "In ________, the AKP now has the most representatives." the model ranks the correct answer ("parliament") first, followed by "undersecretary," "elections," and "congress," all of which are thematically appropriate, and most of which fit contextually into the sentence. For the German word “Spieler”, the top ranking predictions made by the model are “spider,” “smaller,” and “spill,” while one of the actual user guesses (“speaker”) is ranked as 10th most likely (out of a vocabulary of 5000 items).

3.2 Macaronic Setting

In §3.1 we restricted the sentences to only contain a single noun token in the foreign language, while the rest of the tokens were in L1. In this section we model incidental comprehension for the “full macaronic” setting, where multiple tokens can be in the foreign language, and the word ordering may also be in the foreign language word-order. The stimuli of interest are now like “**`Der Polizist`** `arrested the` **`Bankräuber.`**” (“The police arrested the bank robber”). Even in this scenario, with multiple tokens have been replaced with their L2 equivalent, a reader with no knowledge of German is likely to be able to understand this sentence reasonably well by using cognate and context clues. These clues provide enough *scaffolding* for the reader to infer the meaning of novel words and hopefully infer the meaning of the entire sentence. In these stimuli the novel foreign words jointly influence each other along with other words that are in the students native language. Of course, there are several possible configurations for this sentence where a reader might not be able to understand this sentence. Our goal is to model configurations that are understandable while exposing the reader to as much novel L2 vocabulary and sentence structure as possible.

Our experimental subjects are required to guess what “Polizist” and “Bankräuber” mean in this sentence. We train a featurized model to predict these guesses jointly within each sentence and thereby predict incidental comprehension on any macaronic sentence. Indeed, we hope our model design will generalize from predicting incidental comprehension on *macaronic* sentences (for our beginner subjects, who need some context words to be in English) to predicting incidental comprehension on *full German* sentences (for more advanced students, who understand some of the context words *as if* they were in English). In addition, we developed a user interface that uses macaronic sentences directly as a medium of language instruction. Chapter 4 details the user interface.

3.2.1 Data Collection Setup

Our method of scaffolding is to replace certain foreign words and phrases with their English translations, yielding a macaronic sentence.[9] Simply presenting these to a learner would not give us feedback on the learner’s belief state for each foreign word. Even assessing the learner’s reading comprehension would give only weak indirect information about what was understood. Thus, we collect data where a learner explicitly guesses a foreign word’s translation when seen in the macaronic context. These guesses are then treated as supervised labels to train our user model.

We used Amazon Mechanical Turk (MTurk) to collect data. Users qualified for tasks by completing a short quiz and survey about their language knowledge. Only users whose

[9]Although the language distinction is indicated by italics and color, users were left to figure this out on their own.

results indicated no knowledge of German and self-identified as native speakers of English were allowed to complete tasks. With German as the foreign language, we generated content by crawling a simplified-German news website, `nachrichtenleicht.de`. We chose simplified German in order to minimize translation errors and to make the task more suitable for novice learners. We translated each German sentence using the Moses Statistical Machine Translation (SMT) toolkit (Koehn et al., 2007). The SMT system was trained on the German-English Commoncrawl parallel text used in WMT 2015 (Bojar et al., 2015).

We used 200 German sentences, presenting each to 10 different users. In MTurk jargon, this yielded 2000 Human Intelligence Tasks (HITs). Each HIT required its user to participate in several rounds of guessing as the English translation was incrementally revealed. A user was paid US $0.12 per HIT, with a bonus of US $6 to any user who accumulated more than 2000 total points.

3.2.1.1 HITs and Submissions

For each HIT, the user first sees a German sentence[10] (Figure 3.5). A text box is presented below each German word in the sentence, for the user to type in their "best guess" of what each German word means. The user must fill in at least half of the text boxes before submitting this set of guesses. The resulting *submission*—i.e., the macaronic sentence together with the set of guesses—is logged in a database as a single training example, and

[10]Except that we first "translate" any German words that have identical spelling in English (case-insensitive). This includes most proper names, numerals, and punctuation marks. Such translated words are displayed in English style (blue italics), and the user is not asked to guess their meanings.

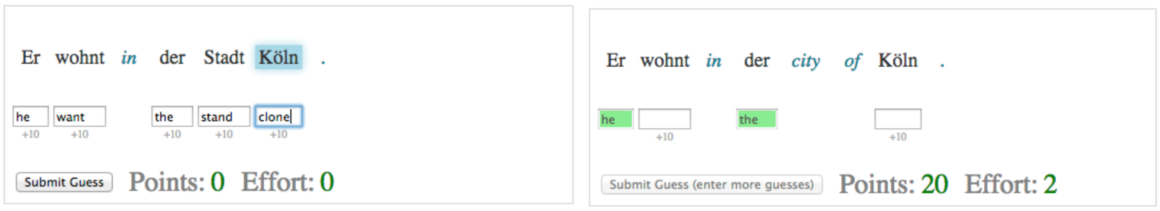

Er wohnt in der Stadt Köln .
he
want
the
stand
clone
+10
Submit Guess
Points: 0
Effort: 0
Er wohnt in der city of Köln .
he
the
+10
Submit Guess (enter more guesses)
Points: 20
Effort: 2

Figure 3.5: After a user submits a set of guesses (top), the interface marks the correct guesses in green and also reveals a set of translation clues (bottom). The user now has the opportunity to guess again for the remaining German words.

the system displays feedback to the user about which guesses were correct.

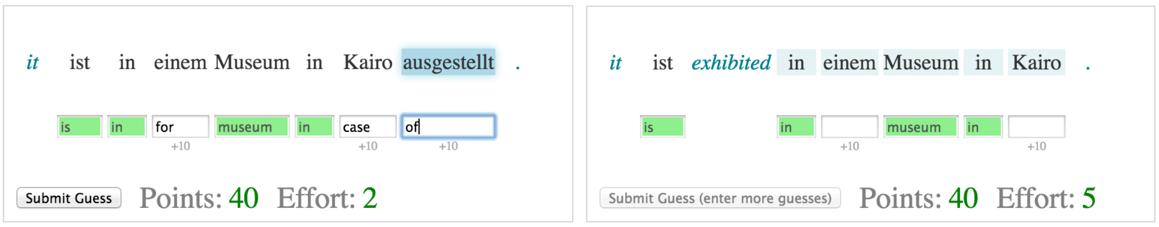
After each submission, new *clues* are revealed (providing increased scaffolding) and the user is asked to guess again. The process continues, yielding multiple submissions, until all German words in the sentence have been translated. At this point, the entire HIT is considered completed and the user moves to a new HIT (i.e., a new sentence).

From our 2000 HITs, we obtained 9392 submissions (4.7 per HIT) from 79 distinct MTurk users.

3.2.1.2 Clues

Each update provides new clues to help the user make further guesses. There are 2 kinds of clues:

Translation Clue (Figure 3.5): A set of words that were originally in German are replaced with their English translations. The text boxes below these words disappear, since it is no longer necessary to guess them.



it ist in einem Museum in Kairo ausgestellt .

is | in | for | museum | in | case | of

Points: 40 Effort: 2

it ist *exhibited* in einem Museum in Kairo .

is | in | | museum | in |

Points: 40 Effort: 5

Figure 3.6: In this case, after the user submits a set of guesses (top), two clues are revealed (bottom): `ausgestellt` is moved into English order and then translated.

Reordering Clue (Figure 3.6): A German substring is moved into a more English-like position. The reordering positions are calculated using the word and phrase alignments obtained from Moses.

Each time the user submits a set of guesses, we reveal a sequence of $n = \max(1, \text{round}(N/3))$ clues, where N is the number of German words remaining in the sentence. For each clue, we sample a token that is currently in German. If the token is part of a movable phrase, we move that phrase; otherwise we translate the minimal phrase containing that token. These moves correspond exactly to clues that a user could request by clicking on the token in the macaronic reading interface, see Chapter 4 for details of how moves are constructed and animated. In our present experiments, the system is in control instead, and grants clues by “randomly clicking” on n tokens.

The system’s probability of sampling a given token is proportional to its unigram type probability in the WMT corpus. Thus, rarer words tend to remain in German for longer, allowing the Turker to attempt more guesses for these “difficult” words. However, close cognates would be an exeption to this rule, as they are not frequent but probably easy to guess by Turkers.

3.2.1.3 Feedback

When a user submits a set of guesses, the system responds with feedback. Each guess is visibly “marked” in left-to-right order, momentarily shaded with green (for correct), yellow (for close) or red (for incorrect). Depending on whether a guess is correct, close, or wrong, users are awarded points as discussed below. Yellow and red shading then fades, to signal to the user that they may try entering a new guess. Correct guesses remain on the screen for the entire task.

3.2.1.4 Points

Adding points to the process (Figures 3.5–3.6) adds a game-like quality and lets us incentivize users by paying them for good performance. We award 10 points for each exactly correct guess (case-insensitive). We give additional “effort points” for a guess that is close to the correct translation, as measured by cosine similarity in vector space. (We used pre-trained GLoVe word vectors (Pennington, Socher, and Manning, 2014); when the guess or correct translation has multiple words, we take the average of the word vectors.) We deduct effort points for guesses that are careless or very poor. Our rubric for effort points is as follows:

$$e_p = \begin{cases} -1, & \text{if } \hat{e} \text{ is repeated or nonsense (red)} \\ -1, & \text{if } \mathrm{sim}(\hat{e}, e^*) < 0 \text{ (red)} \\ 0, & \text{if } 0 \leq \mathrm{sim}(\hat{e}, e^*) < 0.4 \text{ (red)} \\ 0, & \text{if } \hat{e} \text{ is blank} \\ 10 \times \mathrm{sim}(\hat{e}, e^*) & \text{otherwise (yellow)} \end{cases}$$

Here $\mathrm{sim}(\hat{e}, e^*)$ is cosine similarity between the vector embeddings of the user's guess $\hat{e}$ and our reference translation e^*. A "nonsense" guess contains a word that does not appear in the sentence bitext nor in the 20,000 most frequent word types in the GLoVe training corpus. A "repeated" guess is an incorrect guess that appears more than once in the set of guesses being submitted.

In some cases, $\hat{e}$ or e^* may itself consist of multiple words. In this case, our points and feedback are based on the best match between any word of $\hat{e}$ and any word of e^*. In alignments where multiple German words translate as a single phrase,[11] we take the phrasal translation to be the correct answer e^* for *each* of the German words.

3.2.1.5 Normalization

After collecting the data, we normalized the user guesses for further analysis. All guesses were lowercased. Multi-word guesses were crudely replaced by the longest word in the

[11]Our German-English alignments are constructed as in Renduchintala et al. (2016a)

guess (breaking ties in favor of the earliest word).

The guesses included many spelling errors as well as some nonsense strings and direct copies of the input. We defined the *dictionary* to be the 100,000 most frequent word types (lowercased) from the WMT English data. If a user's guess $\hat{e}$ does not match e^* and is not in the dictionary, we replace it with:

- the special symbol `<COPY>`, if $\hat{e}$ appears to be a copy of the German source word f (meaning that its Levenshtein distance from f is $< 0.2 \cdot \max(|\hat{e}|, |f|)$);
- else, the closest word in the dictionary[12] as measured by Levenshtein distance (breaking ties alphabetically), provided the dictionary has a word at distance ≤ 2;
- else `<BLANK>`, as if the user had not guessed.

3.2.2 User Model

In each submission, the user jointly guesses several English words, given spelling and context clues. One way that a *machine* could perform this task is via probabilistic inference in a factor graph—and we take this as our model of how the *human user* solves the problem.

The user observes a German sentence $\mathbf{f} = [f_1, f_2, \ldots, f_i, \ldots f_n]$. The translation of each word token f_i is E_i, which is from the user's point of view a random variable. Let Obs denote the set of indices i for which the user also observes that $E_i = e_i^*$, the aligned reference translation, because e_i^* has already been guessed correctly (green feedback) or shown as a

[12]Considering only words returned by the Pyenchant 'suggest' function (http://pythonhosted.org/pyenchant/).

clue. Thus, the user's posterior distribution over $\mathbf{E}$ is $P_{\boldsymbol{\theta}}(\mathbf{E} = \mathbf{e} \mid \mathbf{E}_{\text{Obs}} = \mathbf{e}^*_{\text{Obs}}, \mathbf{f}, \text{history})$, where "history" denotes the user's history of past interactions.

We assume that a user's submission ê is derived from this posterior distribution simply as a random sample. We try to fit the parameter vector $\boldsymbol{\theta}$ to maximize the log-probability of the submission. Note that our model is trained on the user guesses ê, not the reference translations $\mathbf{e}$*. That is, we seek parameters $\boldsymbol{\theta}$ that would explain why all users made their guesses.

Although we fit a single $\boldsymbol{\theta}$, this does not mean that we treat users as interchangeable (since $\boldsymbol{\theta}$ can include user-specific parameters) or unvarying (since our model conditions users' behavior on their history, which can capture some learning).

3.2.3 Factor Graph

We model the posterior distribution as a conditional random field (Figure 3.7) in which the value of E_i depends on the form of f_i as well as on the meanings e_j (which may be either observed or jointly guessed) of the context words at $j \neq i$:

$$P_{\boldsymbol{\theta}}(\mathbf{E} = \mathbf{e} \mid \mathbf{E}_{\text{Obs}} = \mathbf{e}^*_{\text{Obs}}, \mathbf{f}, \text{history}) \tag{3.5}$$

$$\propto \prod_{i \notin \text{Obs}} (\psi^{\text{ef}}(e_i, f_i) \cdot \prod_{j \neq i} \psi^{\text{ee}}(e_i, e_j, i - j))$$

We will define the factors ψ (the *potential functions*) in such a way that they do not "know German" but only have access to information that is available to an naive English speaker. In brief, the factor $\psi^{\text{ef}}(e_i, f_i)$ considers whether the hypothesized English word e_i

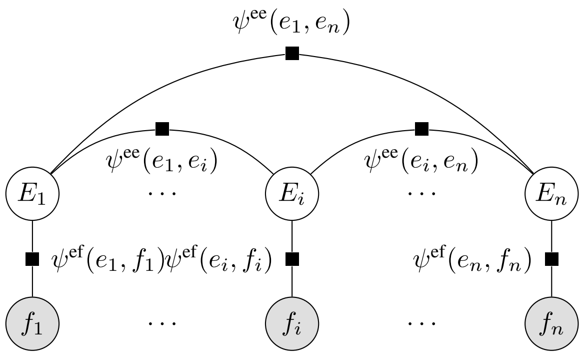

$\psi^{\text{ee}}(e_1, e_n)$
$\psi^{\text{ee}}(e_1, e_i)$
$\psi^{\text{ee}}(e_i, e_n)$
E_1
E_i
E_n
$\psi^{\text{ef}}(e_1, f_1)$
$\psi^{\text{ef}}(e_i, f_i)$
$\psi^{\text{ef}}(e_n, f_n)$
f_1
f_i
f_n

Figure 3.7: Model for user understanding of L2 words in sentential context. This figure shows an inference problem in which all the observed words in the sentence are in German (that is, Obs $= \emptyset$). As the user observes translations via clues or correctly-marked guesses, some of the E_i become shaded.

"looks like" the observed German word f_i, and whether the user has previously observed during data collection that e_i is a correct or incorrect translation of f_i. Meanwhile, the factor $\psi^{\text{ee}}(e_i, e_j)$ considers whether e_i is commonly seen in the context of e_j in English text. For example, the user will elevate the probability that $E_i =$ `cake` if they are fairly certain that E_j is a related word like `eat` or `chocolate`.

The potential functions ψ are parameterized by $\boldsymbol{\theta}$, a vector of feature weights. For convenience, we define the features in such a way that we expect their weights to be positive. We rely on just 6 features at present (see Section 3.2.7 for future work), although each is complex and real-valued. Thus, the weights $\boldsymbol{\theta}$ control the relative influence of these 6 different types of information on a user's guess. Our features broadly fall under the following categories: *Cognate*, *History*, and *Context*. We precomputed cognate and context features, while history features are computed on-the-fly for each training instance. All features are case-insensitive.

3.2.3.1 Cognate Features

For each German token f_i, the ψ^{ef} factor can score each possible guess e_i of its translation:

$$\psi^{\text{ef}}(e_i, f_i) = \exp(\boldsymbol{\theta}^{\text{ef}} \cdot \boldsymbol{\phi}^{\text{ef}}(e_i, f_i)) \tag{3.6}$$

The feature function $\boldsymbol{\phi}^{\text{ef}}$ returns a vector of 4 real numbers:

- *Orthographic Similarity*: The normalized Levenshtein distance between the 2 strings.

  $$\phi^{\text{ef}}_{\text{orth}}(e_i, f_i) = 1 - \frac{\text{lev}(e_i, f_i)}{\max(|e_i|, |f_i|)} \tag{3.7}$$

  The weight on this feature encodes how much users pay attention to spelling.

- *Pronunciation Similarity*: This feature is similar to the previous one, except that it calculates the normalized distance between the *pronunciations* of the two words:

  $$\phi^{\text{ef}}_{\text{pron}}(e_i, f_i) = \phi^{\text{ef}}_{\text{orth}}(\text{prn}(e_i), \text{prn}(f_i)) \tag{3.8}$$

  where the function $\text{prn}(x)$ maps a string x to its pronunciation. We obtained pronunciations for all words in the English and German vocabularies using the CMU pronunciation dictionary tool (Weide, 1998). Note that we use English pronunciation rules even for German words. This is because we are modeling a naive learner who may, in the absence of intuition about German pronunciation rules, apply English pronunciation rules to German.

3.2.3.2 History Features

- *Positive History Feature*: If a user has been rewarded in a previous HIT for guessing e_i as a translation of f_i, then they should be more likely to guess it again. We define $\boldsymbol{\phi}^{\text{ef}}_{\text{hist+}}(e_i, f_i)$ to be 1 in this case and 0 otherwise. The weight on this feature encodes whether users learn from positive feedback.

- *Negative History Feature*: If a user has already incorrectly guessed e_i as a translation of f_i in a previous submission during this HIT, then they should be less likely to guess it again. We define $\boldsymbol{\phi}^{\text{ef}}_{\text{hist-}}(e_i, f_i)$ to be -1 in this case and 0 otherwise. The weight on this feature encodes whether users remember negative feedback.[13]

3.2.3.3 Context Features

In the same way, the ψ^{ef} factor can score the compatibility of a guess e_i with a context word e_j, which may itself be a guess, or may be observed:

$$\psi^{\text{ee}}_{ij}(e_i, e_j) = \exp(\boldsymbol{\theta}^{\text{ee}} \cdot \boldsymbol{\phi}^{\text{ee}}(e_i, e_j, i - j)) \tag{3.9}$$

[13]At least in short-term memory—this feature currently omits to consider any negative feedback from previous HITs.

$\boldsymbol{\phi}^{\text{ee}}$ returns a vector of 2 real numbers:

$$\boldsymbol{\phi}^{\text{ee}}_{\text{pmi}}(e_i, e_j) = \begin{cases} \text{PMI}(e_i, e_j) \text{ if } |i-j| > 1 \\ 0 \text{ otherwise} \end{cases} \tag{3.10}$$

$$\boldsymbol{\phi}^{\text{ee}}_{\text{pmi1}}(e_i, e_j) = \begin{cases} \text{PMI}_1(e_i, e_j) \text{ if } |i-j| = 1 \\ 0 \text{ otherwise} \end{cases} \tag{3.11}$$

where the pointwise mutual information $\text{PMI}(x, y)$ measures the degree to which the English words x, y tend to occur in the same English sentence, and $\text{PMI}_1(x, y)$ measures how often they tend to occur in adjacent positions. These measurements are estimated from the English side of the WMT corpus, with smoothing performed as in Knowles et al. (2016).

For example, if $f_i = \texttt{Suppe}$, the user's guess of E_i should be influenced by $f_j = \texttt{Brot}$ appearing in the same sentence, if the user suspects or observes that its translation is $E_j = \texttt{bread}$. The PMI feature knows that `soup` and `bread` tend to appear in the same English sentences, whereas PMI$_1$ knows that they tend not to appear in the bigram `soup bread` or `bread soup`.

3.2.3.4 User-Specific Features

Apart from the basic 6-feature model, we also trained a version that includes user-specific copies of each feature (similar to the domain adaptation technique of Daumé III (2007)). For example, $\phi^{\text{ef}}_{\text{orth},32}(e_i, f_i)$ is defined to equal $\phi^{\text{ef}}_{\text{orth}}(e_i, f_i)$ for submissions by user 32, and defined to be 0 for submissions by other users.

Thus, with 79 users in our dataset, we learned 6×80 feature weights: a local weight vector for each user and a global vector of "backoff" weights. The global weight $\theta^{\text{ef}}_{\text{orth}}$ is large if users in general reward orthographic similarity, while $\theta^{\text{ef}}_{\text{orth},32}$ (which may be positive or negative) captures the degree to which user 32 rewards it more or less than is typical. The user-specific features are intended to capture individual differences in incidental comprehension.

3.2.4 Inference

According to our model, the probability that the user guesses $E_i = \hat{e}_i$ is given by a marginal probability from the CRF. Computing these marginals is a combinatorial optimization problem that involves reasoning jointly about the possible values of each E_i ($i \notin \text{Obs}$), which range over the English vocabulary V^{e}.

We employ loopy belief propagation (Murphy, Weiss, and Jordan, 1999) to obtain approximate marginals over the variables $\mathbf{E}$. A *tree-based* schedule for message passing was used (Dreyer and Eisner, 2009). We run 3 iterations with a new random root for each iteration.[14]

We define the vocabulary V^{e} to consist of all reference translations e^*_i and normalized user guesses $\hat{e}_i$ from our entire dataset (see Section 3.2.1.5), about 5K types altogether including `<BLANK>` and `<COPY>`. We define the cognate features to treat `<BLANK>` as the empty string and to treat `<COPY>` as f_i. We define the PMI of these special symbols

[14]Remark: In the non-loopy case (which arises for us in cases with ≤ 2 unobserved variables), this schedule boils down to the forward-backward algorithm. In this case, a single iteration is sufficient for exact inference.

with any e to be the mean PMI with e of all dictionary words, so that they are essentially uninformative.

3.2.5 Parameter Estimation

We learn our parameter vector $\boldsymbol{\theta}$ to approximately maximize the regularized log-likelihood of the users' guesses:

$$\Big(\sum \log P_{\boldsymbol{\theta}}(\mathbf{E} = \hat{\mathbf{e}} \mid \mathbf{E}_{\text{Obs}} = \mathbf{e}^*_{\text{Obs}}, \mathbf{f}, \text{history})\Big) - \lambda ||\boldsymbol{\theta}||^2 \tag{3.12}$$

where the summation is over all submissions in our dataset. The gradient of each summand reduces to a difference between observed and expected values of the feature vector $\boldsymbol{\phi} = (\boldsymbol{\phi}^{\text{ef}}, \boldsymbol{\phi}^{\text{ee}})$, summed over all factors in (3.5). The observed features are computed directly by setting $\mathbf{E} = \hat{\mathbf{e}}$. The expected features (which arise from the log of the normalization constant of (3.5)) are computed approximately by loopy belief propagation.

We trained $\boldsymbol{\theta}$ using stochastic gradient descent (SGD),[15] with a learning rate of 0.1 and regularization parameter of 0.2. The regularization parameter was tuned on our development set.

3.2.6 Experimental Results

We divided our data randomly into 5550 training instances, 1903 development instances, and 1939 test instances. Each instance was a single submission from one user, consisting of

[15]To speed up training, SGD was parallelized using Recht et al.'s (2011) Hogwild! algorithm. We trained for 8 epochs.

a batch of "simultaneous" guesses on a macaronic sentence.

We noted qualitatively that when a large number of English words have been revealed, particularly content words, the users tend to make better guesses. Conversely, when most context is German, we unsuprisingly see the user leave many guesses blank and make other guesses based on string similarity triggers. Such submissions are difficult to predict as different users will come up with a wide variety of guesses; our model therefore resorts to predicting similar-sounding words. For detailed examples of this see Appendix 3.2.6.3.

For each foreign word f_i in a submission with $i \notin \text{Obs}$, our inference method (section 3.2.4) predicts a marginal probability distribution over a user's guesses $\hat{e}_i$. Table 3.5 shows our ability to predict user guesses.[16] Recall that this task is essentially a structured prediction task that does joint 4919-way classification of each German word. Roughly 1/3 of the time, our model's top 25 words include the user's exact guess.

However, the recall reported in Table 3.5 is too stringent for our educational application. We could give the model partial credit for predicting a synonym of the learner's guess $\hat{e}$. More precisely, we would like to give the model partial credit for predicting when the learner will make a poor guess of the truth e^*—even if the model does not predict the user's specific incorrect guess $\hat{e}$.

To get at this question, we use English word embeddings (as in Section 3.2.1.4) as a proxy for the semantics and morphology of the words. We measure the *actual quality* of the

[16]Throughout this section, we ignore the 5.2% of tokens on which the user did not guess (i.e., the guess was `<BLANK>` after the normalization of Section 3.2.1.5). Our present model simply treats `<BLANK>` as an ordinary and very bland word (section 3.2.4), rather than truly attempting to predict when the user will not guess. Indeed, the model's posterior probability of `<BLANK>` in these cases is a paltry 0.0000267 on average (versus 0.0000106 when the user does guess). See Section 3.2.7.

Model	**Recall at k (dev)**			**Recall at k (test)**		
	1	**25**	**50**	**1**	**25**	**50**
Basic	15.24	34.26	38.08	16.14	35.56	40.30
User-Adapted	15.33	34.40	38.67	16.45	35.71	40.57

Table 3.5: Percentage of foreign words for which the user's actual guess appears in our top-k list of predictions, for models with and without user-specific features ($k \in \{1, 25, 50\}$).

learner's guess $\hat{e}$ as its cosine similarity to the truth, $\text{sim}(\hat{e}, e^*)$. While quality of 1 is an exact match, and quality scores > 0.75 are consistently good matches, we found quality of ≈ 0.6 also reasonable. Pairs such as (`mosque`, `islamic`) and (`politics`, `government`) are examples from the collected data with quality ≈ 0.6. As quality becomes < 0.4, however, the relationship becomes tenuous, e.g., (`refugee`, `soil`).

Similarly, we measure the *predicted quality* as $\text{sim}(e, e^*)$, where e is the model's 1-best prediction of the user's guess. Figure 3.8 plots predicted vs. actual quality (each point represents one of the learner's guesses on development data), obtaining a correlation of 0.38, which we call the "quality correlation" or QC. A clear diagonal band can be seen, corresponding to the instances where the model exactly predicts the user's guess. The cloud around the diagonal is formed by instances where the model's prediction was not identical to the user's guess but had similar quality.

We also consider the *expected* predicted quality, averaging over the model's predictions

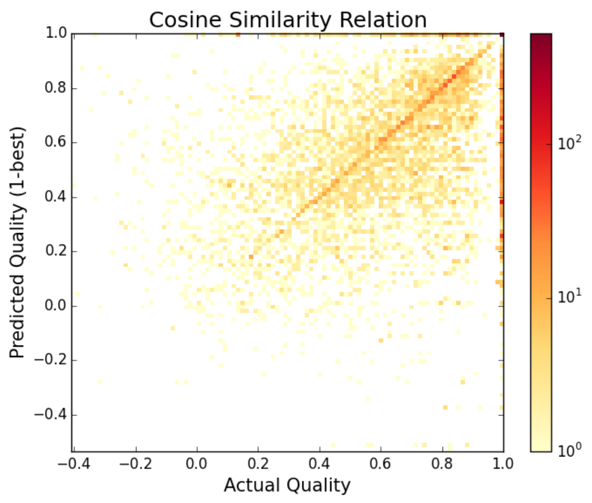


Figure 3.8: Actual quality $\mathrm{sim}(\hat{e}, e^*)$ of the learner's guess $\hat{e}$ on development data, versus predicted quality $\mathrm{sim}(e, e^*)$ where e is the basic model's 1-best prediction.

e of $\hat{e}$ (for all $e \in V^{\mathrm{e}}$) in proportion to the probabilities that it assigns them. This allows the model to more smoothly assess whether the learner is likely to make a high-quality guess. Figure 3.9 shows this version, where the points tend to shift upward and the quality correlation (QC) rises to 0.53.

All QC values are given in Table 3.6. We used expected QC on the development set as the criterion for selecting the regularization coefficient λ and as the early stopping criterion during training.

3.2.6.1 Feature Ablation

To test the usefulness of different features, we trained our model with various feature categories disabled. To speed up experimentation, we sampled 1000 instances from the training set and trained our model on those. The resulting QC values on dev data are shown

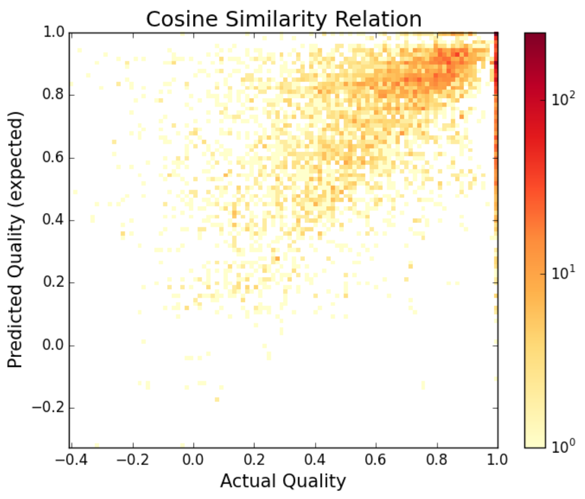


Figure 3.9: Actual quality $\mathrm{sim}(\hat{e}, e^*)$ of the learner's guess $\hat{e}$ on development data, versus the expectation of the predicted quality $\mathrm{sim}(e, e^*)$ where e is distributed according to the basic model's posterior.

in Table 3.7. We see that removing history-based features has the most significant impact on model performance: both QC measures drop relative to the full model. For cognate and context features, we see no significant impact on the expected QC, but a significant drop in the 1-best QC, especially for context features.

3.2.6.2 Analysis of User Adaptation

Table 3.6 shows that the user-specific features significantly improve the *1-best* QC of our model, although the much smaller improvement in *expected* QC is insignificant.

User adaptation allows us to discern different styles of incidental comprehension. A user-adapted model makes fine-grained predictions that could help to construct better macaronic sentences for a given user. Each user who completed at least 10 HITs has their user-specific

Model	**Dev**		**Test**	
	Exp	**1-Best**	**Exp**	**1-Best**
Basic	0.525	0.379	0.543	0.411
User-Adapted	0.527	0.427	0.544	0.439

Table 3.6: Quality correlations: basic and user-adapted models.

weight vector shown as a row in Figure 3.10. Recall that the user-specific weights are not used in isolation, but are *added* to backoff weights shared by all users.

These user-specific weight vectors cluster into four groups. Furthermore, the average points per HIT differ by cluster (significantly between each cluster pair), reflecting the success of different strategies.[17] Users in group (a) employ a generalist strategy for incidental comprehension. They pay typical or greater-than-typical attention to all features of the current HIT, but many of them have diminished memory for vocabulary learned during past HITs (the hist+ feature). Users in group (b) seem to use the opposite strategy, deriving their success from retaining common vocabulary across HITs (hist+) and falling back on orthography for new words. Group (c) users, who earned the most points per HIT, appear to make heavy use of context and pronunciation features *together* with hist+. We also see that pronunciation similarity seems to be a stronger feature for group (c) users, in contrast to the more superficial orthographic similarity. Group (d), which earned the fewest points per HIT,

[17]Recall that in our data collection process, we award points for each HIT (section 3.2.1.4). While the points were designed more as a reward than as an evaluation of learner success, a higher score does reflect more guesses that were correct or close, while a lower score indicates that some words were never guessed before the system revealed them as clues.

Feature Removed	QC	
	Expected	**1-Best**
None	0.522	0.425
Cognate	0.516	0.366*
Context	0.510	0.366*
History	0.499*	0.259*

Table 3.7: Impact on quality correlation (QC) of removing features from the model. Ablated QC values marked with asterisk* differ significantly from the full-model QC values in the first row ($p < 0.05$, using the test of Preacher (2002)).

appears to be an "extreme" version of group (b): these users pay unusually little attention to any model features other than orthographic similarity and hist+. (More precisely, the model finds group (d)'s guesses harder to predict on the basis of the available features, and so gives a more uniform distribution over V^{e}.)

3.2.6.3 Example of Learner Guesses vs. Model Predictions

To give a sense of the problem difficulty, we have hand-picked and presented two training examples (submissions) along with the predictions of our basic model and their log-probabilities. In Figure 3.11a a large portion of the sentence has been revealed to the user in English (blue text) only 2 words are in German. The text in bold font is the user's guess. Our model expected both words to be guessed; the predictions are listed below the German

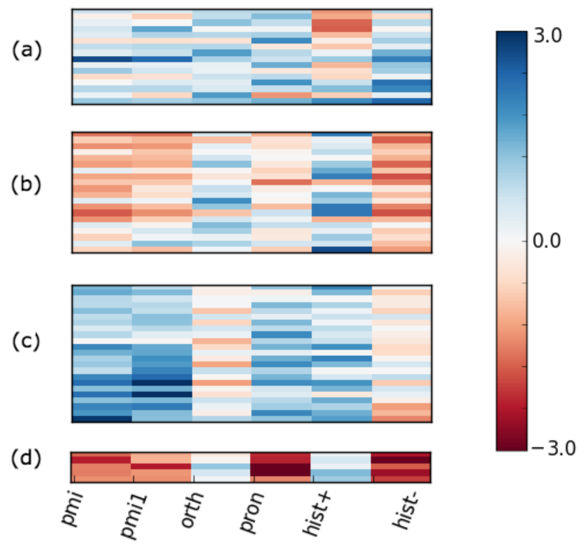


Figure 3.10: The user-specific weight vectors, clustered into groups. Average points per HIT for the HITs completed by each group: (a) 45, (b) 48, (c) 50 and (d) 42.

words `Verschiedene` and `Regierungen`. The reference translation for the 2 words are `Various` and `governments`. In Figure 3.11b we see a much harder context where only one word is shown in English and this word is not particularly helpful as a contextual anchor.

3.2.7 Future Improvements to the Model

Our model's feature set (section 3.2.3) could clearly be refined and extended. Indeed, in the previous section, we use a more tightly controlled experimental design to explore some simple feature variants. A cheap way to vet features would be to test whether they help on the task of modeling reference translations, which are more plentiful and less noisy than the user guesses.

For *Cognate* features, there exist many other good string similarity metrics (including trainable ones). We could also include $\boldsymbol{\phi}^{\text{ef}}$ features that consider whether e_i's part of speech, frequency, and length are plausible given f_i's burstiness, observed frequency, and length. (E.g., only short common words are plausibly translated as determiners.)

For *Context* features, we could design versions that are more sensitive to the position and status of the context word j. We speculate that the actual influence of e_j on a user's guess e_i is stronger when e_j is observed rather than itself guessed;when there are fewer intervening tokens (and particularly fewer observed ones);and when $j < i$. Orthogonally, $\boldsymbol{\phi}^{\text{ef}}(e_i, e_j)$ could go beyond PMI and windowed PMI to also consider cosine similarity, as well as variants of these metrics that are thresholded or nonlinearly transformed. Finally, we do not have to treat the context positions j as independent multiplicative influences as in equation (3.5) (cf. Naive Bayes): we could instead use a topic model or some form of language model to determine a conditional probability distribution over E_i given all other words in the context.

An obvious gap in our current feature set is that we have no $\boldsymbol{\phi}^{\text{e}}$ features to capture that some words $e_i \in V^{\text{e}}$ are more likely guesses *a priori*. By defining several versions of this feature, based on frequencies in corpora of different reading levels, we could learn user-specific weights modeling which users are unlikely to think of an obscure word.We should also include features that fire specifically on the reference translation e_i^* and the special symbols `<BLANK>` and `<COPY>`, as each is much more likely than the other features would suggest.

For *History* features, we could consider *negative* feedback from other HITs (not just the current HIT), as well as *positive* information provided by revealed clues (not just confirmed guesses).We could also devise non-binary versions in which more recent or more frequent feedback on a word has a stronger effect.More ambitiously, we could model generalization: after being shown that `Kind` means `child`, a learner might increase the probability that the similar word `Kinder` means `child` or something related (`children`, `childish`, . . .), whether because of superficial orthographic similarity or a deeper understanding of the morphology. Similarly, a learner might gradually acquire a model of typical spelling changes in English-German cognate pairs.

A more significant extension would be to model a user's learning process. Instead of representing each user by a small vector of user-specific weights, we could recognize that the user's guessing strategy and knowledge can change over time.

A serious deficiency in our current model is that we treat `<BLANK>` like any other word. A more attractive approach would be to learn a stochastic link from the posterior distribution to the user's guess or non-guess, instead of assuming that the user simply samples the guess from the posterior. As a simple example, we might say the user guesses $e \in V^{\mathrm{e}}$ with probability $p(e)^{\beta}$—where $p(e)$ is the posterior probability and $\beta > 1$ is a learned parameter—with the remaining probability assigned to `<BLANK>`. This says that the user tends to avoid guessing except when there are relatively high-posterior-probability words to guess.

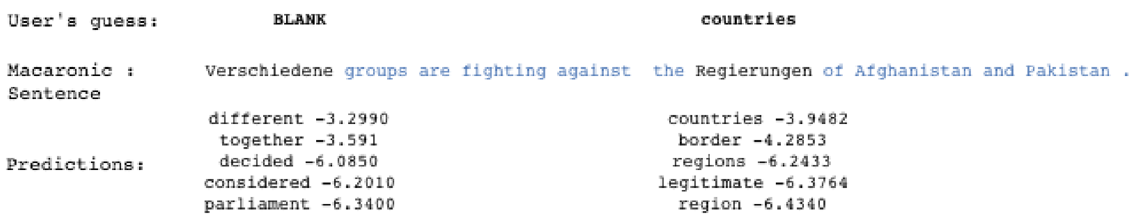
Finally, newer representation learning models such as BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019) could also be used in our model. XLNet in particular, with its ability to account for interdependencies between output tokens, would be a good candidate to provide rich contextual features to either replace or used along with the weaker pair-wise PMI based features of our current model.

3.2.8 Conclusion

We have presented a methodology for collecting data and training a model to estimate a foreign language learner's understanding of L2 vocabulary in partially understood contexts. Both are novel contributions to the study of L2 acquisition.

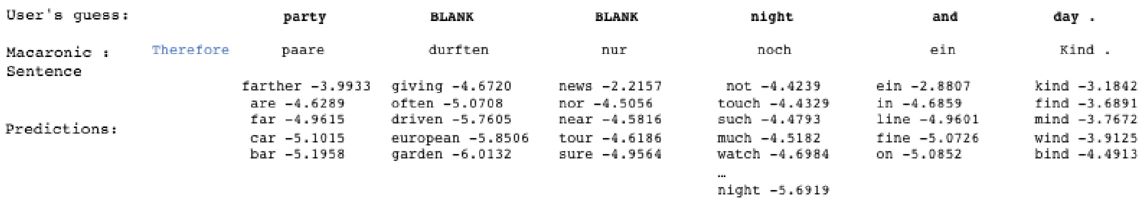
Our current model is arguably crude, with only 6 features, yet it can already often do a reasonable job of predicting what a user might guess and whether the user's guess will be roughly correct. This opens the door to a number of future directions with applications to language acquisition using personalized content and learners' knowledge.

We leave as future work the integration of this model into an adaptive system that tracks learner understanding and creates scaffolded content that falls in their zone of proximal development, keeping them engaged while stretching their understanding.



User's guess:	BLANK		countries
Macaronic : Sentence	Verschiedene groups are fighting against	the	Regierungen of Afghanistan and Pakistan .
Predictions:	different -3.2990 together -3.591 decided -6.0850 considered -6.2010 parliament -6.3400		countries -3.9482 border -4.2853 regions -6.2433 legitimate -6.3764 region -6.4340

(a)



User's guess:		party	BLANK	BLANK	night	and	day .
Macaronic : Sentence	Therefore	paare	durften	nur	noch	ein	Kind .
Predictions:		farther -3.9933 are -4.6289 far -4.9615 car -5.1015 bar -5.1958	giving -4.6720 often -5.0708 driven -5.7605 european -5.8506 garden -6.0132	news -2.2157 nor -4.5056 near -4.5816 tour -4.6186 sure -4.9564	not -4.4239 touch -4.4329 such -4.4793 much -4.5182 watch -4.6984 … night -5.6919	ein -2.8807 in -4.6859 line -4.9601 fine -5.0726 on -5.0852	kind -3.1842 find -3.6891 mind -3.7672 wind -3.9125 bind -4.4913

(b)

Figure 3.11: Two examples of the system's predictions of what the user will guess on a single submission, contrasted with the user's actual guess. (The user's previous submissions on the same task instance are not shown.) In 3.11a, the model correctly expects that the substantial context will inform the user's guess. In 3.11b, the model predicts that the user will fall back on string similarity—although we can see that the user's actual guess of `and day` was likely informed by their guess of `night`, an influence that our CRF did consider. The numbers shown are log-probabilities. Both examples show the sentences in a macaronic state (after some reordering or translation has occurred). For example, the original text of the German sentence in 3.11b reads `Deshalb durften die Paare nur noch ein Kind bekommen .` The macaronic version has undergone some reordering, and has also erroneously dropped the verb due to an incorrect alignment.

Chapter 4

Creating Interactive Macaronic Interfaces for Language Learning

In the previous chapter, we presented models for incidental learning. We hope to generate macaronic text by consulting such models. Recall that the AI teacher's goal is to generate comprehensible macaronic texts for a student to read. Given a macaronic data structure associated with a piece of text, the AI teacher must render a macaronic configuration that it believes the student will understand (and learn from). But what if the AI teacher makes a sentence that is too difficult for the student to read? Or too easy with very little L2 words? For such cases, we would like to give "control" back to the student and let them interactively modify the macaronic sentence. Suppose the sentence is too difficult, we would like the student to not get completely stuck, so we would like to give them the chance to ask for hints. On the flip-side, if a student feels there is not enough L2 content in a macaronic sentence, we want interact with the data structure and explore macaronic spectrum.

To provide these features to the student, we design a user-interface in which such modifications are possible. We present the details of our user-interface along with interaction modalities in this chapter.

We provide details of the current user interface and discuss how content for our system can be automatically generated using existing statistical machine translation (SMT) methods, enabling learners or teachers to choose their own texts to read. Our interface lets the user navigate through the spectrum from L2 to L1, going beyond the single-word or single-phrase translations offered by other online tools such as Swych (2015), or dictionary-like browser plugins.

Finally, we note that the interaction design could include logging all of the actions a student makes while reading a text. We can then use the logged actions to refine our incidental learning model to hopefully produce macaronic text that is more personalized to the student's L2 level. We leave this for future work.

4.1 Macaronic Interface

To illustrate the workings of our interface, we assume a native English speaker (L1=English) who is learning German (L2=German). However, our existing interface can accommodate any pair of languages whose writing systems share directionality.[1] The primary goal of the interface is to empower a learner to translate and reorder parts of a confusing foreign

[1]We also assume that the text is segmented into words.

language sentence. These translations and reorderings serve to make the German sentence more English-like. The interface also permits reverse transformations, letting the curious learner “peek ahead” at how specific English words and constructions would surface in German.

Using these fundamental interactions as building blocks, we create an interactive framework for a language learner to explore this continuum of “English-like” to “foreign-like” sentences. By repeated interaction with new content and exposure to recurring vocabulary items and linguistic patterns, we believe a learner can pick up vocabulary and other linguistic rules of the foreign language.

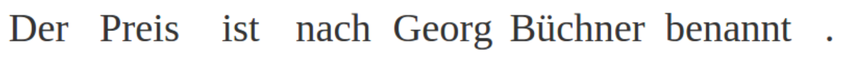
4.1.1 Translation

The basic interface idea is that a line of macaronic text is equipped with hidden interlinear annotations. Notionally, English translations lurk below the macaronic text, and German ones above.

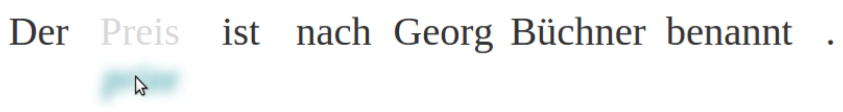
The *Translation* interaction allows the learner to change the text in the macaronic sentence from one language to another. Consider a macaronic sentence that is completely in the foreign state (i.e.,, entirely in German), as shown in Fig. 4.1a. Hovering on or under a German word shows a *preview* of a translation (Fig. 4.1b). Clicking on the preview will cause the translation to “rise up” and replace the German word (Fig. 4.1c).

To translate in the reverse direction, the user can hover and click above an English word (Fig. 4.1d).

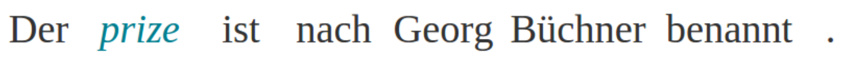
Since the same mechanism applies to all the words in the sentence, a learner can manipulate translations for each word independently. For example, Fig. 4.1e shows two words in English.


Der Preis ist nach Georg Büchner benannt .

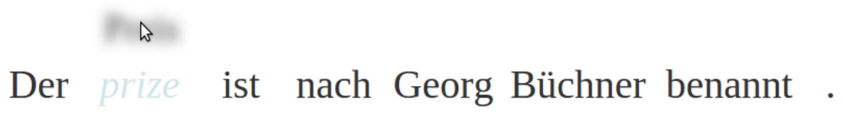
(a) Initial sentence state.


Der Preis ist nach Georg Büchner benannt .

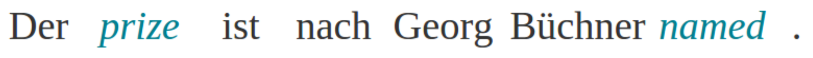
(b) Mouse hovered under `Preis`.


Der *prize* ist nach Georg Büchner benannt .

(c) `Preis` translated to `prize`.


Der *prize* ist nach Georg Büchner benannt .

(d) Mouse hovered above `prize`. Clicking above will revert the sentence back to the initial state 4.1a.


Der *prize* ist nach Georg Büchner *named* .

(e) Sentence with 2 different words translated into English

Figure 4.1: Actions that translate words.

The version of our prototype displayed in Figure 4.1 blurs the preview tokens when a learner is hovering above or below a word. This blurred preview acts as a visual indication of a potential change to the sentence state (if clicked) but it also gives the learner a chance to think about what the translation might be, based on visual clues such as length and shape of the blurred text.

4.1.2 Reordering

When the learner hovers slightly *below* the words `nach Georg Büchner` a *Reordering* arrow is displayed (as shown in Figure 4.2). The arrow is an indicator of reordering. In this example, the German past participle `benannt` appears at the end of the sentence (the conjugated form of the verb is `ist benannt`, or `is named`); this is the grammatically correct location for the participle in German, while the English form should appear earlier in the equivalent English sentence.

Similar to the translation actions, reordering actions also have a directional attribute. Figure 4.2b shows a German-to-English direction arrow. When the learner clicks the arrow, the interface rearranges all the words involved in the reordering. The new word positions are shown in 4.2c. Once again, the user can undo: hovering just *above* `nach Georg Büchner` now shows a gray arrow, which if clicked returns the phrase to its German word order (shown in 4.2d).

German phrases that are not in original German order are highlighted as a warning (Figure 4.2c).

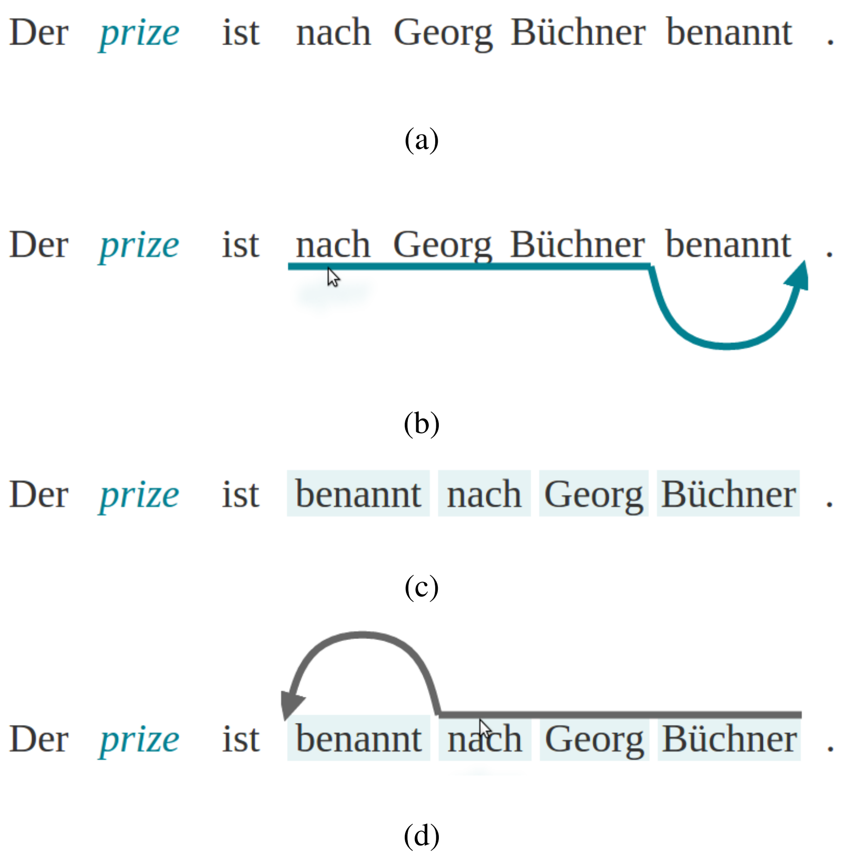

Der prize ist nach Georg Büchner benannt .
(a)
Der prize ist nach Georg Büchner benannt .
(b)
Der prize ist benannt nach Georg Büchner .
(c)
Der prize ist benannt nach Georg Büchner .
(d)

Figure 4.2: Actions that reorder phrases.

4.1.3 “Pop Quiz” Feature

So far, we have described the system’s standard responses to a learner’s actions. We now add occasional “pop quizzes.” When a learner hovers below a German word (s_0 in Figure 4.3) and clicks the blurry English text, the system can either reveal the translation of the German word (state s_2) as described in section 4.1.1 or quiz the learner (state s_3). We implement the quiz by presenting a text input box to the learner: here the learner is expected to type what they believe the German word means. Once a guess is typed, the system indicates if the

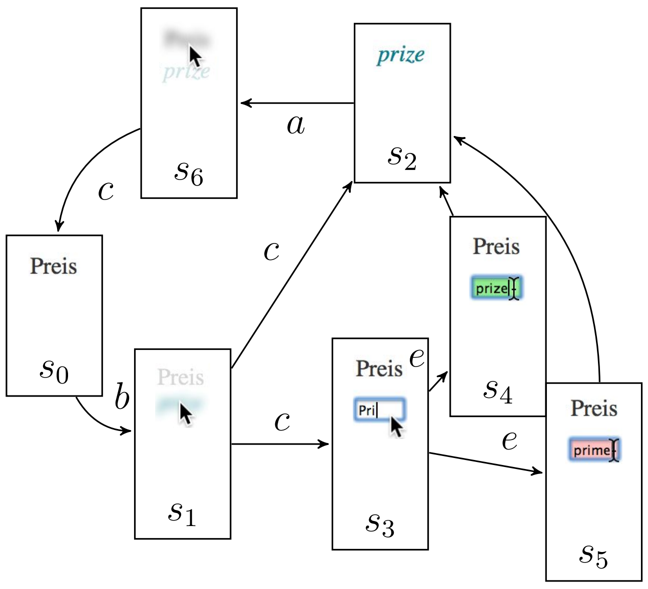


Figure 4.3: State diagram of learner interaction (edges) and system's response(vertices). Edges can be traversed by clicking (c), hovering above (a), hovering below (b) or the enter (e) key. Unmarked edges indicate an automatic transition.

guess is correct (s_4) or incorrect(s_5) by flashing green or red highlights in the text box. The box then disappears (after 700ms) and the system automatically proceeds to the reveal state s_2. As this imposes a high cognitive load and increases the interaction complexity (typing vs. clicking), we intend to use the pop quiz infrequently.

The pop quiz serves two vital functions. First, it further incentivizes the user to retain learned vocabulary. Second, it allows the system to update its model of the user's current L2 lexicon, macaronic comprehension, and learning style; this is work in progress (see section 4.3.2).

4.1.4 Interaction Consistency

Again, we regard the macaronic sentence as a kind of interlinear text, written between two mostly invisible sentences: German above and English below. In general, hovering above the macaronic sentence will reveal German words or word orders, which fall down into the macaronic sentence upon clicking. Hovering below will reveal English translations, which rise up upon clicking.

The words in the macaronic sentence are colored according to their language. We want the user to become accustomed to reading German, so the German words are in plain black text by default, while the English words use a marked color and font (italic blue). Reordering arrows also follow the same color scheme: arrows that will make the macaronic sentence more "German-like" are gray, while arrows that make the sentence more "English-like" are blue. The summary of interactions is shown in Table 4.1.

Action	**Direction**	**Trigger**	**Preview**	**Preview Color**	**Confirm**	**Result**
Translation	E-to-G	Hover above English	Blurry German translation above	Gray Blur	Click on Blurry Text	translation replaces English word(s)
	G-to-E	Hover under German token	Blurry English translation below	Blue Blur	Click on Blurry Text	translation replaces German word(s)
Reordering	E-to-G	Hover above token	Arrow above reordering tokens	Gray Arrow	Click on Arrow	tokens reorder
	G-to-E	Hover under token	Arrow below reordering tokens	Blue Arrow	Click on Arrow	tokens reorder

Table 4.1: Summary of learner triggered interactions in the Macaronic Interface.

4.2 Constructing Macaronic Translations

In this section, we describe the details of the underlying data structures needed to allow all the interactions mentioned in the previous section. A key requirement in the design of the data structure was to support orthogonal actions in each sentence. Making all translation and reordering actions independent of one another creates a large space of macaronic states for a learner to explore.

At present, the input to our macaronic interface is bitext with word-to-word alignments provided by a phrase-based SMT system (or, if desired, by hand). We employ Moses (Koehn et al., 2007) to translate German sentences and generate phrase alignments. News articles written in simple German from `nachrichtenleicht.de` (Deutschlandfunk, 2016) were translated after training the SMT system on the WMT15 German-English corpus (Bojar et al., 2015).

We convert the word alignments into "minimal alignments" that are either one-to-one, one-to-many or many-to-one. For each many-to-many alignment returned by the SMT system, we remove alignment edges (lowest probability first) until the alignment is no longer many-to-many. Then we greedily add edges from unaligned tokens (highest probability first), subject to not creating many-to-many alignments and subject to minimizing the number of crossing edges, until all tokens are aligned. This step ensures consistent reversibility of actions and prevents large phrases from being translated with a single click.[2] The resulting

[2]Preliminary experiments showed that allowing large phrases to translate with one click resulted in abrupt jumps in the visualization, which users found hard to follow.

bipartite graph can be regarded as a collection of connected components, or *units* (Fig. 4.4).[3]

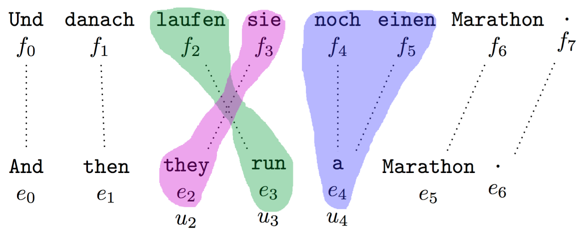


Figure 4.4: The dotted lines show word-to-word alignments between the German sentence $f_0, f_1, \ldots, f_7$ and its English translation $e_0, e_1, \ldots, e_6$. The figure highlights 3 of the 7 units: u_2, u_3, u_4.

4.2.1 Translation Mechanism

In a given state of the macaronic sentence, each unit is displayed in either English or German. A translation action toggles the display language of the unit, leaving it in place. For example, in Figure 4.5, where the macaronic sentence is currently displaying $f_4 f_5 = \texttt{noch einen}$, a translation action will replace this with $e_4 = \texttt{a}$.

[3]In the sections below, we gloss over cases where a unit is discontiguous (in one language). Such units are handled specially (we omit details for reasons of space). If a unit would fall outside the bounds of what our special handling can handle, we fuse it with another unit.

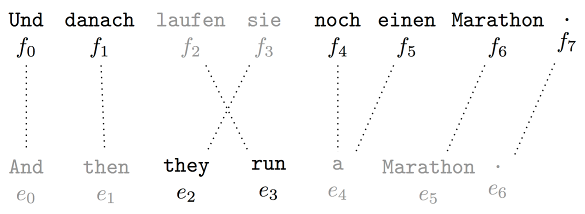


Figure 4.5: A possible state of the sentence, which renders a subset of the tokens (shown in black). The rendering order (section 4.2.2) is not shown but is also part of the state. The string displayed in this case is ”`Und danach` *`they run`* `noch einen Marathon.`” (assuming no reordering).

4.2.2 Reordering Mechanism

A reordering action changes the unit order of the current macaronic sentence. The output string “`Und danach` *`they run`* `noch einen Marathon.`” is obtained from Figure 4.5 only if unit u_2 (as labeled in Figure 4.4) is rendered (in its current language) to the left of unit u_3, which we write as $u_2 < u_3$. In this case, it is possible for the user to change the order of these units, because $u_3 < u_2$ in German. Table 4.2 shows the 8 possible combinations of ordering and translation choices for this pair of units.

String Rendered	**Unit Ordering**
`...they run...`	$\{u_2\} < \{u_3\}$
`...they laufen...`	
`...sie run...`	
`...sie laufen...`	
`...run they...`	$\{u_2\} > \{u_3\}$
`...run sie...`	
`...laufen they...`	
`...laufen sie...`	

Table 4.2: Generating reordered strings using units.

The space of possible orderings for a sentence pair is defined by a bracketing ITG tree (Wu, 1997), which transforms the German ordering of the units into the English ordering by a collection of nested binary swaps of subsequences.[4] The ordering state of the macaronic sentence is given by the subset of these swaps that have been performed. A reordering action toggles one of the swaps in this collection.

Since we have a parser for German (Rafferty and Manning, 2008), we take care to select an ITG tree that is "compatible" with the German sentence's dependency structure, in the following sense: if the ITG tree combines two spans A and B, then there are not dependencies from words in A to words in B *and* vice-versa.

[4]Occasionally no such ITG tree exists, in which case we fuse units as needed until one does.

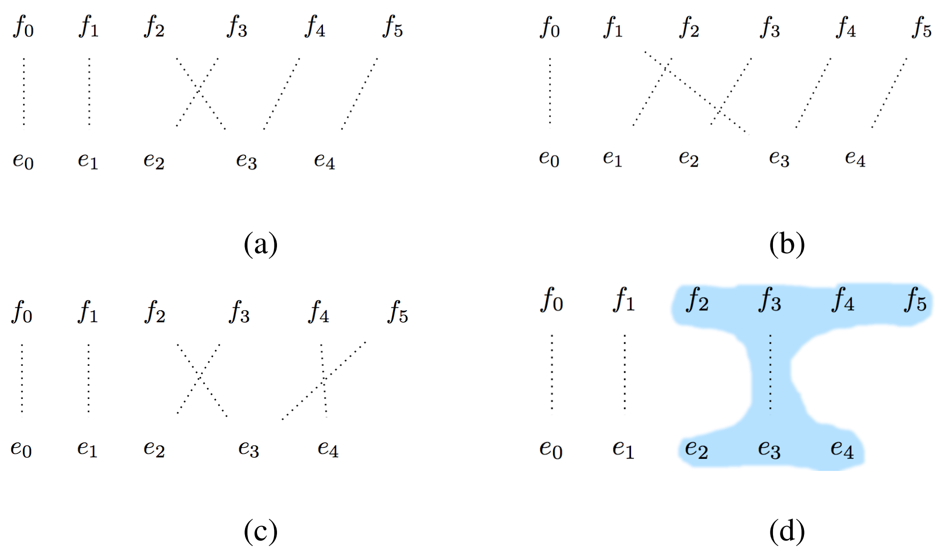


Figure 4.6: Figure 4.6a shows a simple discontiguous unit. Figure 4.6b shows a long distance discontiguity which is supported. In figure 4.6c the interruptions align to both sides of e_3 which is not supported. In situations like 4.6c, all associated units are merged as one phrasal unit (shaded) as shown in figure 4.6d

4.2.3 Special Handling of Discontiguous Units

We provide limited support for alignments which form *discontiguous units*. Figure 4.6a shows a simple discontiguous unit. In this example, a reordering action (G-to-E direction) performed on either f_2 or f_4 will move f_2 to the immediate left of f_4 eliminating the interrupting alignment. After reordering, the translation action becomes available to the learner, just as in a multi-word contiguous unit. The system currently supports one or more interrupting units as long as these units are contiguous and are from only one side of the single token (see Figure 4.6a and 4.6b). If the conditions for special handling are not satisfied (see Figure 4.6c), the system forces all the tokens to a single unit, which results in a phrasal alignment and is treated as a single unit (Figure 4.2d). Such units have no reordering actions and result in a phrasal translation. We also employ this “back off” phrasal alignment in cases where alignments do not satisfy the ITG constraint.

4.3 Discussion

4.3.1 Machine Translation Challenges

When the English version of the sentence is produced by an MT system, it may suffer from MT errors and/or poor alignments.

Even with correct MT, a given syntactic construction may be handled inconsistently on different occasions, depending on the particular words involved (as these affect what phrasal alignment is found and how we convert it to a minimal alignment). Syntax-based MT could be used to design a more consistent interface that is also more closely tied to classroom L2 lessons.

Cross-linguistic divergences in the expression of information (Dorr, 1994) could be confusing. For example, when moving through macaronic space from `Kaffee gefällt Menschen` (coffee pleases humans) to its translation *`humans like coffee`*, it may not be clear to the learner that the reordering is triggered by the fact that *`like`* is not a literal translation of `gefällt`. One way to improve this might be to have the system pass smoothly through a range of intermediate translations from word-by-word glosses to idiomatic phrasal translations, rather than always directly translating idioms. Concretely, we can first transform `Kaffee gefällt Menschen` into `Kaffee gefällt` *`humans`* and then into `Kaffee` *`pleases humans`* and finally into *`coffee pleases humans`*. These transitions could be done via manual rules. Once all tokens of the German phrase are in English the final transition would render the phrase in "correct" English *`humans like coffee`*. We might also see benefit in guiding our gradual translations with cognates (for example, rather than translate directly from the German `Möhre` to the English *`carrot`*, we might offer the cognate `Karotte` as an intermediate step).

Another avenue of research is to transition through words that are macaronic at the sub-word level. For example, hovering over the unfamiliar German word `gesprochen` might decompose it into `ge-sprochen`; then clicking on one of those morphemes might yield `ge-`*`talk`* or `sprech`*`-ed`* before reaching *`talked`*. This could guide learners towards an understanding of German tense marking and stem changes. Generation of these sub-word macaronic forms could be done using multilingual trained morphological reinflectionn systems such as Kann, Cotterell, and Schütze (2017).

4.3.2 User Adaptation and Evaluation

We would prefer to show the learner a macaronic sentence that provides just enough clues for the learner to be able to comprehend it, while still pushing them to figure out new vocabulary or new structures. Thus, we plan to situate this interface in a framework that continuously adapts as the user progresses. As the user learns new vocabulary, the system will automatically present them with more challenging sentences (containing less L1). In **??** we show that we can predict a novice learner's guesses of L2 word meanings in macaronic sentences using a few simple features. We will subsequently track the user's learning by observing their mouse actions and "pop quiz" responses (section 4.1).

While we have had users interact with our system in order to collect data about novice learners' guesses, we are working toward an evaluation where our system is used to supplement classroom instruction for real foreign-language students.

4.4 Conclusion

In this work we present a prototype of an interactive interface for learning to read in a foreign language. We expose the learner to L2 vocabulary and constructions in contexts that are comprehensible because they have been *partially* translated into the learner's native language, using statistical MT. Using MT affords flexibility: learners or instructors can choose which texts to read, and learners or the system can control which parts of a sentence are translated.

In the long term, we would like to extend the approach to allow users also to *produce* macaronic language, drawing on techniques from grammatical error correction or computer-aided translation to help them gradually remove L1 features from their writing (or speech) and make it more L2-like. We leave this for future work.

Chapter 5

Construction of Macaronic Texts for Vocabulary Learning

5.1 Introduction

In the previous chapters, we presented a interactive interface to read macaronic sentences and a model that predicts a student's guessing abilities which used information from the L1 and L2 context as well as cognate information as input features. To train this model we require supervised data, meaning data on student behaviors and capabilities (Renduchintala et al., 2016b; Labutov and Lipson, 2014). The data collection for supervised data involves prompting students (in our experiments we used MTurk users) with macaronic sentences created *randomly* (or with some heuristic) and then asking the MTurk "students" to guess the meanings of L2 words in these sentences. The random macaronic sentences paired with student guesses forms the training data. This step is expensive, not only from a data collection point of view, but also from the point of view of students, as they would have to give feedback (i.e. generate labeled data) on the actions of an initially untrained machine teacher.

In this chapter, we show that it is possible to design a machine teacher without any supervised data from (human) students. We use a neural cloze language model instead of the weaker conditional random field used earlier. We also propose a method to allow our cloze language model to incrementally learn new vocabulary items, and use this language model as a proxy for the word guessing and learning ability of real students. A machine foreign-language teacher decides which subset of words to replace by consulting this cloze language model. The cloze language model is initially trained on a corpus of L1 texts and is therefore not personalized to a (human) student. Despite this, we show that a machine foreign-language can generate pedagogically useful macaronic texts after consulting with the cloze language model. We are essentially using a cloze language model as a "drop-in" replacement for a true user model we refer to the cloze language as a *generic student model*.

We evaluate three variants of our generic student language models through a study on Amazon Mechanical Turk (MTurk). We find that MTurk "students" were able to guess the meanings of L2 words (in context) introduced by the machine teacher with high accuracy for both function words as well as content words in two out of the three models. Furthermore, we select the best performing variant and evaluate if participants can actually *learn* the L2 words by letting participants read a macaronic passage and give an L2 vocabulary quiz at

Sentence	The	Nile	is	a	river	in	Africa
Gloss	**Der**	**Nil**	**ist**	**ein**	**Fluss**	**in**	**Afrika**
Macaronic	**Der**	Nile	**ist**	a	river	in	Africa
Configurations	The	Nile	is	a	**Fluss**	in	Africa
	Der	**Nil**	**ist**	**ein**	river	in	Africa

Table 5.1: An example English (L1) sentence with German (L2) glosses. Using the glosses, many possible macaronic configurations are possible. Note that the gloss sequence is not a fluent L2 sentence.

the end of passage, where the L2 words are presented without their sentential context.

5.1.1 Limitation

While we gain the ability to construct macaronic texts for students without any prior data collection, we limit ourselves to lexical replacements only. This limitation arises because of our proposed method to evaluate the knowledge of the generic student model compares lexical word embeddings and is therefore unable to measure other linguistic knowledge such as word order. This is a key limitation of the work proposed in this chapter.

5.2 Method

Our machine teacher can be viewed as a search algorithm that tries to find the (approximately) best macaronic configuration for the next sentence in a given L1 document. We assume the availability of a "gold" L2 gloss for each L1 word: in our experiments, we obtained these from bilingual speakers using Mechanical Turk. Table 5.1 shows an example English sentence with German glosses and three possible macaronic configurations (there are exponentially many configurations). The machine teacher must assess, for example, how accurately a student would understand the meanings of **`Der`**, **`ist`**, **`ein`**, and **`Fluss`** when presented with the following candidate macaronic configuration: **`Der`** `Nile` **`ist ein Fluss`** `in Africa.`[1] Understanding may arise from inference on this sentence as well as whatever the student has learned about these words from previous sentences. The teacher makes this assessment by presenting this sentence to a generic student model (§§5.2.1–5.5). It uses a L2 embedding scoring scheme (§5.5.1) to guide a greedy search for the best macaronic configuration (§5.5.2).

5.2.1 Generic Student Model

Our model of a "generic student" (`GSM`) is equipped with a cloze language model that uses a bidirectional LSTM to predict L1 words in L1 context (Mousa and Schuller, 2017;

[1]By "meaning" we mean the L1 token that was originally in the sentence before it was replaced by an L2 gloss.

Hochreiter and Schmidhuber, 1997). Given a sentence $\mathbf{x} = [x_1, \ldots, x_t, \ldots, x_T]$, the cloze model defines $p(x_t \mid \mathbf{h^f}_t, \mathbf{h^b}_t)\ \forall t \in \{1, \ldots, T\}$, where:

$$\mathbf{h^f}_t = \text{LSTM}^{\mathbf{f}}([\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}]; \boldsymbol{\theta}^{\mathbf{f}}) \in \mathbb{R}^D \tag{5.1}$$

$$\mathbf{h^b}_t = \text{LSTM}^{\mathbf{b}}([\mathbf{x}_T, \ldots, \mathbf{x}_{t+1}]; \boldsymbol{\theta}^{\mathbf{b}}) \in \mathbb{R}^D \tag{5.2}$$

are hidden states of forward and backward LSTM encoders parameterized by $\boldsymbol{\theta}^{\mathbf{f}}$ and $\boldsymbol{\theta}^{\mathbf{b}}$ respectively. The model assumes a fixed L1 vocabulary of size V, and the vectors $\mathbf{x}_t$ above are embeddings of these word types, which correspond to the rows of an embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times D}$. The cloze distribution at each position t in the sentence is obtained using

$$p(\cdot \mid \mathbf{h^f}, \mathbf{h^b}) = \text{softmax}(\mathbf{E}\ h([\mathbf{h^f}; \mathbf{h^b}];\ \boldsymbol{\theta}^{\mathbf{h}})) \tag{5.3}$$

where $h(\cdot; \boldsymbol{\theta}^{\mathbf{h}})$ is a projection function that reduces the dimension of the concatenated hidden states from $2D$ to D. We "tie" the input embeddings and output embeddings as in Press and Wolf (2017).

We train the parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}^{\mathbf{f}}; \boldsymbol{\theta}^{\mathbf{b}}; \boldsymbol{\theta}^{\mathbf{h}}; \mathbf{E}]$ using Adam (Kingma and Ba, 2014) to maximize $\sum_{\mathbf{x}} \mathcal{L}(\mathbf{x})$, where the summation is over sentences $\mathbf{x}$ in a large L1 training corpus, and

$$\mathcal{L}(\mathbf{x}) = \sum_t \log\ p(x_t \mid \mathbf{h^f}_t, \mathbf{h^b}_t) \tag{5.4}$$

We set the dimensionality of word embeddings and LSTM hidden units to 300. We use the WikiText-103 corpus (Merity et al., 2016) as the L1 training corpus. We apply dropout $(p = 0.2)$ between the word embeddings and LSTM layers, and between the LSTM and projection layers (Srivastava et al., 2014). We assume that the resulting model represents the entirety of the student's L1 knowledge.

5.2.2 Incremental L2 Vocabulary Learning

The model so far can assign probability to an L1 sentence such as `The Nile is a river in Africa`, using equation (5.4), but what about a macaronic sentence such as **`Der`** `Nile` **`ist ein Fluss`** `in Africa`? To accommodate the new L2 words, we use another word-embedding matrix, $\mathbf{F} \in \mathbb{R}^{V' \times D}$ and modify Eq 5.3 to consider both the L1 and L2 embeddings:

$$p(\cdot \mid [\mathbf{h^f} : \mathbf{h^b}]) = \text{softmax}([\mathbf{E}; \mathbf{F}] \cdot h([\mathbf{h^f} : \mathbf{h^b}]; \boldsymbol{\theta}^{\mathbf{h}}))$$

We also restrict the softmax function above to produce a distribution not over the full bilingual vocabulary of size $|V| + |V'|$, but only over the bilingual vocabulary consisting of the V L1 types together with only the $v' \subset V'$ L2 types that actually appear in the macaronic sentence $\mathbf{x}$. In the above example macaronic sentence, $|v'|$ is 4. We initialize $\mathbf{F}$ by drawing its elements IID from Uniform$[-0.01, 0.01]$. Thus, all L2 words initially have random embeddings $[-0.01, 0.01]^{1 \times D}$.

These modifications lets us compute $\mathcal{L}(\mathbf{x})$ for a macaronic sentence $\mathbf{x}$. We assume that when a human student reads a macaronic sentence $\mathbf{x}$, they update their L2 parameters $\mathbf{F}$ (but not their L1 parameters $\boldsymbol{\theta}$) to increase $\mathcal{L}(\mathbf{x})$. Specifically, we assume that $\mathbf{F}$ will be updated to maximize

$$\mathcal{L}(\mathbf{x}; \boldsymbol{\theta}^{\mathrm{f}}, \boldsymbol{\theta}^{\mathrm{b}}, \boldsymbol{\theta}^{\mathrm{h}}, \mathbf{E}, \mathbf{F}) - \lambda \|\mathbf{F} - \mathbf{F}^{\mathbf{prev}}\|^2 \tag{5.5}$$

Maximizing equation (5.5) adjusts the embeddings of each L2 word in the sentence so that it is more easily predicted from the other L1/L2 words, and also so that it is more helpful at predicting the other L1/L2 words. Since the rest of the model's parameters do not change, we expect to find an embedding for **Fluss** that is similar to the embedding for `river`. However, the regularization term with coefficient $\lambda > 0$ prevents $\mathbf{F}$ from straying too far from from $\mathbf{F}^{\mathbf{prev}}$, which represents the value of $\mathbf{F}$ before this sentence was read. This limits the degree to which our simulated student will change their embedding of an L2 word such as **Fluss** based on a *single* example. As a result, the embedding of **Fluss** reflects *all* of the past sentences that contained **Fluss**, although (realistically) with some bias toward the most recent such sentences. We do not currently model spacing effects, i.e., forgetting due to the passage of time.

In principle, λ should be set based on human-subjects experiments, and might differ from human to human. In practice, in this paper, we simply took $\lambda = 1$. We (approximately) maximized the objective above using 5 steps of gradient ascent, which gave good convergence in practice.

5.2.3 Scoring L2 embeddings

The incremental vocabulary learning procedure (§5.2.2) takes a macaronic configuration and generates a new L2 word-embedding matrix by applying gradient updates to a previous version of the L2 word-embedding matrix. The new matrix represents the proxy student's L2 knowledge after observing the macaronic configuration.

Thus, if we can score the new L2 embeddings, we can, in essence, score the macaronic configuration that generated it. The ability to score configurations affords search (§§ 5.2.4 and 5.2.5) for high-scoring configurations. With this motivation, we design a scoring function to measure the "goodness" of L2 word-embeddings, $\mathbf{F}$.

The machine teacher evaluates $\mathbf{F}$ with reference to all correct word-gloss pairs from the *entire document*. For our example sentence, the word pairs are {(`The`, **`Der`**), (`is`,**`ist`**), (`a`,**`ein`**), (`river`,**`Fluss`**)}. But the machine teacher also has access to, for example, {(`water`,**`Wasser`**), (`stream`, **`Fluss`**) ... }, which come from elsewhere in the document. Thus, if $\mathcal{P}$ is the set of word pairs,$\{(x_1, f_1), ...(x_{|\mathcal{P}|}, f_{|\mathcal{P}|})\}$, we compute:

$$\tilde{r}_p = R(x_p, \mathbf{cs}(\mathbf{F}_{f_p}, \mathbf{E})) \tag{5.6}$$

$$r_p = \begin{cases} \tilde{r}_p & \text{if } \tilde{r}_p < r_{\text{max}} \\ \infty & \text{otherwise} \end{cases}$$

$$\text{MRR}(\mathbf{F}, \mathbf{E}, r_{\text{max}}) = \frac{1}{|\mathcal{P}|} \sum_p \frac{1}{r_p} \tag{5.7}$$

where $\mathbf{cs}(\mathbf{F}_f, \mathbf{E})$ denotes the vector of cosine similarities between the embedding of an L2 word f and the entire L1 vocabulary. $R(x, \mathbf{cs}(\mathbf{E}, \mathbf{F}_f))$ queries the rank of the correct L1 word x that pairs with f. r can take values from 1 to $|V|$, but we use a rank threshold r_{max} and force pairs with a rank worse than r_{max} to ∞. Thus, given a word-gloss pairing $\mathcal{P}$, the current state of the L2 embedding matrix $\mathbf{F}$, and the L1 embedding matrix $\mathbf{E}$, we obtain the Mean Reciprocal Rank (MRR) score in (5.21).

We can think of the scoring function as a "vocabulary test" in which the proxy student gives (its best) r_{max} guesses for each L2 word type and receives a numerical grade.

5.2.4 Macaronic Configuration Search

So far we have detailed our simulated student that would learn from a macaronic sentence, and a metric to measure how good the learned L2 embeddings would be. Now the machine teacher only has to search for the best macaronic configuration of a sentence. As there are exponentially many possible configurations to consider, exhaustive search is infeasible. We use a simple left-to-right greedy search to approximately find the highest scoring configuration for a given sentence. Algorithm 1 shows the pseudo-code for the search process. The inputs to the search algorithm are the initial L2 word-embeddings matrix $\mathbf{F}^{\text{prev}}$, the scoring function MRR$()$, and the generic student model SPM$()$. The algorithm proceeds left to right, making a binary decision at each token: Should the token be replaced with its L2 gloss or left as is? For the first token, these two decisions result in the two configurations: (i) **Der** Nile... and (ii) The Nile... These configurations are given to the generic student model which updates the L2 word embeddings. The scoring function (section 5.5.1) computes a score for each L2 word-embedding matrix and caches the best configuration (i.e. the configuration associated with the highest scoring L2 word-embedding matrix). If two configurations result in the same MRR score, the number of L2 word types exposed is used to break ties. In Algorithm 1, $\rho(\mathbf{c})$ is the function that counts the number of L2 word types exposed in a configuration $\mathbf{c}$.

5.2.5 Macaronic-Language document creation

Our idea is that a sequence of macaronic configurations is good if it drives the generic student model's L2 embeddings toward an MRR score close to 1 (maximum possible). Note that we do *not* change the sentence order (we still want a coherent document), just the macaronic *configuration* of each sentence. For each sentence in turn, we greedily search over macaronic configurations using Algorithm 1, then choose the configuration that learns the best $\mathbf{F}$, and proceed to the next sentence with $\mathbf{F}^{\mathbf{prev}}$ now set to this learned $\mathbf{F}$.[2] This process is repeated until the end of the document. The pseudo-code for generating an entire document of macaronic content is shown in Algorithm 2.

In summary, our machine teacher is composed of (i) a generic student model which is a contextual L2 word learning model (§5.2.1 and §5.2.2) and (ii) a configuration sequence search algorithm (§5.2.4 and §5.2.5), which is guided by (iii) an L2 vocabulary scoring function (§5.5.1). In the next section, we describe two variations for the generic student models.

[2]For the first sentence, we initialize $\mathbf{F}^{\mathbf{prev}}$ to have values randomly between $[-0.01, 0.01]$.

Algorithm 1 Mixed-Lang. Config. Search

```
Require: $\mathbf{x} = [x_1, x_2, \ldots, x_T]$                ▷ L1 tokens
Require: $\mathbf{g} = [g_1, g_2, \ldots, g_T]$                ▷ L2 glosses
Require: $\mathbf{E}$                                          ▷ L1 emb. matrix
Require: $\mathbf{F}^{\mathbf{prev}}$                          ▷ initial L2 emb. matrix
Require: SPM                                                   ▷ Student Proxy Model
Require: MRR,$r_{\text{max}}$                                  ▷ Scoring Func., threshold
 1: function SEARCH($\mathbf{x}, \mathbf{g}, \mathbf{E}, \mathbf{F}^{\mathbf{prev}}$)
 2:     $\mathbf{c} \leftarrow \mathbf{x}$                     ▷ initial configuration is the L1 sentence
 3:     $\mathbf{F} \leftarrow \mathbf{F}^{\mathbf{prev}}$
 4:     $s = \text{MRR}(\mathbf{E}, \mathbf{F}, r_{\text{max}})$
 5:     for $i = 1; i \leq T; i++$ do
 6:         $\mathbf{c'} \leftarrow c_1 \cdots c_{i-1}\ g_i\ x_{i+1} \cdots x_T$
 7:         $\mathbf{F'} = \text{SPM}(\mathbf{F}^{\mathbf{prev}}, \mathbf{c'})$
 8:         $s' = \text{MRR}(\mathbf{E}, \mathbf{F'}, r_{\text{max}})$
 9:         if $(s', -\rho(\mathbf{c'})) \geq (s, -\rho(\mathbf{c}))$ then
10:             $\mathbf{c} \leftarrow \mathbf{c'}, \mathbf{F} \leftarrow \mathbf{F'}, s \leftarrow s'$
11:         end if
12:     end for
13:     return $\mathbf{c}, \mathbf{F}$                        ▷ Mixed-Lang. Config.
14: end function
```

Algorithm 2 Mixed-Lang. Document Gen.

```
Require: $\mathcal{D} = [(\mathbf{x_1}, \mathbf{g_1}), \ldots, (\mathbf{x_N}, \mathbf{g_N})]$    ▷ Document
Require: $\mathbf{E}$    ▷ L1 emb. matrix
Require: $\mathbf{F^0}$    ▷ initial L2 emb. matrix
1: function DocGen($\mathcal{D}, \mathbf{F^0}$)
2:     $\mathcal{C} = []$    ▷ Configuration List
3:     for $i = 1; i \leq N; i++$ do
4:         $\mathbf{x}_i, \mathbf{g}_i = \mathcal{D}_i$
5:         $\mathbf{c}^i, \mathbf{F^i} = \textsc{Search}(\mathbf{x}_i, \mathbf{g}_i, \mathbf{E}, \mathbf{F^{i-1}})$
6:         $\mathcal{C} \leftarrow \mathcal{C} + [\mathbf{c}^i]$
7:     end for
8:     return $\mathcal{C}$    ▷ Mixed-Lang. Document
9: end function
```

5.3 Variations in Generic Student Models

We developed two variations for the generic student model to compare and contrast the macaronic documents that can be generated.

5.3.1 Unidirectional Language Model

This variation restricts the bidirectional model (from Section 5.2.1) to be unidirectional (uGSM) and follows a standard recurrent neural network (RNN) language model (Mikolov et al., 2010).

$$\log\ p(\mathbf{x}) = \sum_t \log\ p(x_t \mid \mathbf{h^f}_t) \tag{5.8}$$

$$\mathbf{h^f}_t = \text{LSTM}^{\mathbf{f}}(\mathbf{x}_0, \ldots, \mathbf{x}_{t-1}; \boldsymbol{\theta}^{\mathbf{f}}) \tag{5.9}$$

$$p(\cdot \mid \mathbf{h^f}) = \text{softmax}(\mathbf{E} \cdot \mathbf{h^f}) \tag{5.10}$$

Once again, $\mathbf{h^f} \in \mathbb{R}^{D\times 1}$ is the hidden state of the LSTM recurrent network, which is parameterized by $\boldsymbol{\theta}^{\mathbf{f}}$, but unlike the model in Section 5.2.1, no backward LSTM and no projection function is used.

The same procedure from the bidirectional model is used to update L2 word embeddings (Section 5.2.2). While this model does not explicitly encode context from "future" tokens (i.e. words to the right of x_t) , there is still pressure from right-side tokens $x_{t+t:T}$ because the new embeddings will be adjusted to explain the tokens to the right as well. Fixing all the L1 parameters further strengthens this pressure on L2 embeddings from words to their right.

5.3.2 Direct Prediction Model

The previous two models variants adjust L2 embeddings using gradient steps to improve the pseudo-likelihood of the presented macaronic sentences. One drawback of such an approach is computation speed caused by the bottleneck introduced by the softmax operation.

We designed an alternate student prediction model that can "directly" predict the embeddings for words in a sentence using contextual information. We refer to this variation as the *Direct Prediction* (`DP`) model. Like our previous generic student models, the `DP` model also uses bidirectional LSTMs to encode context and an L1 word embedding matrix $\mathbf{E}$. However, the `DP` model does not attempt to produce a distribution over the output vocabulary; instead it tries to predict a real-valued vector using a feed-forward highway network (Srivastava, Greff, and Schmidhuber, 2015). The `DP` model's objective is to minimize the mean square error (MSE) between a predicted word embedding and the *true embedding*. For a time-step t, the predicted word embedding $\hat{\mathbf{x}}_t$, is generated by:

$$\mathbf{h^f}_t = \text{LSTM}^{\mathbf{f}}([\mathbf{x}_1, \dots, \mathbf{x}_{t-1}]; \boldsymbol{\theta}^{\mathbf{f}}) \tag{5.11}$$

$$\mathbf{h^b}_t = \text{LSTM}^{\mathbf{b}}([\mathbf{x}_{t+1}, \dots, \mathbf{x}_T]; \boldsymbol{\theta}^{\mathbf{b}}) \tag{5.12}$$

$$\hat{\mathbf{x}}_t = \text{FF}([\mathbf{x}_t : \mathbf{h^f}_t : \mathbf{h^b}_t]; \boldsymbol{\theta}^{\mathbf{w}}) \tag{5.13}$$

$$\mathcal{L}(\boldsymbol{\theta}^{\mathbf{f}}, \boldsymbol{\theta}^{\mathbf{b}}, \boldsymbol{\theta}^{\mathbf{w}}) = \sum_t (\hat{\mathbf{x}}_t - \mathbf{x}_t)^2 \tag{5.14}$$

where $FF(.; \boldsymbol{\theta}^{\mathbf{w}})$ denotes a feed forward highway network with parameters $\boldsymbol{\theta}^{\mathbf{w}}$. Thus, the `DP` model training requires that we already have the "true embeddings" for all the

L1 words in our corpus. We use pretrained L1 word embeddings from FastText as "true embeddings" (Bojanowski et al., 2017). This leaves the LSTM parameters $\boldsymbol{\theta}^{\text{f}}, \boldsymbol{\theta}^{\text{b}}$ and the highway feed-forward network parameters $\boldsymbol{\theta}^{\text{w}}$ to be learned. Equation 5.14 can be minimized by simply copying the input $\mathbf{x}_t$ as the prediction (ignoring all context). We use *masked training* to prevent the model itself from trivially copying (Devlin et al., 2018). We randomly "mask" 30% of the input embeddings during training. This masking operation replaces the original embedding with either (i) $\mathbf{0}$ vectors, or (ii) vectors of a random word in vocabulary, or (iii) vectors of a "neighboring" word from the vocabulary. [3] The loss, however, is always computed with respect to the correct token embedding.

With the L1 parameters of the `DP` model trained, we turn to L2 learning. Once again the L2 vocabulary is encoded in $\mathbf{F}$, which is initialized to $\mathbf{0}$ (i.e. before any sentence is observed). Consider the configuration: `The Nile is a` **`Fluss`** `in Africa`. The tokens are converted into a sequence of embeddings: $[\mathbf{x}_0 = \mathbf{E}_{x_0}, \ldots, \mathbf{x}_t = \mathbf{F}_{f_t}, ..., \mathbf{x}_T = \mathbf{E}_{x_T}]$. Note that at time-step t the L2 word-embedding matrix is used ($t = 4, f_t =$ **`Fluss`** for the example above). A prediction $\hat{\mathbf{x}}_t$ is generated by the model using Equations 5.11-5.13. Our hope is that the prediction is a "refined" version of the embedding for the L2 word. The refinement arises from considering the context of the L2 word. If **`Fluss`** was not seen before, $\mathbf{x}_t = \mathbf{F}_{f_t} = \mathbf{0}$, forcing the `DP` model to only use contextual information. We apply a

[3]We precompute 20 neighboring words (based on cosine-similarity) for each word in the vocabulary using FastText embeddings before training.

simple update rule that modifies the L2 embeddings based on the direct predictions:

$$\mathbf{F}_{f_t} \leftarrow (1-\eta)\mathbf{F}_{f_t} + \eta\hat{\mathbf{x}}_t \tag{5.15}$$

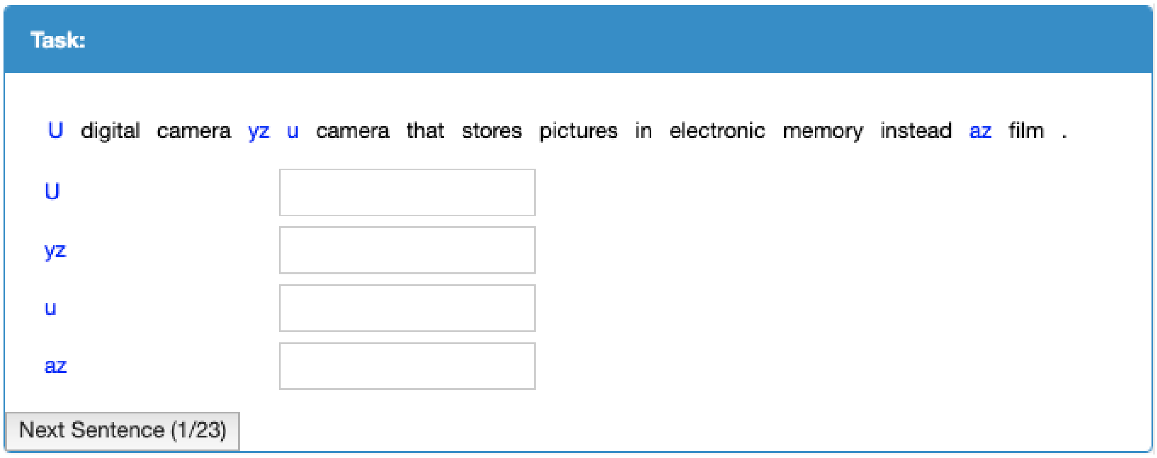
where η controls the interpolation between the old values of a word embedding and the new values which have been predicted based on the current mixed sentence. If there are multiple L2 words in a configuration, say at positions i and j (where $i < j$), we can still follow Eq 5.11–5.13. However, to allow the predictions $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ to jointly influence each other, we need to execute multiple prediction iterations.

Concretely, let $\mathbf{X} = [\mathbf{x}_0, \ldots, \mathbf{F}_{f_i}, \ldots, \mathbf{F}_{f_j}, \ldots, \mathbf{x}_T]$ be the sequence of word embeddings for a macaronic sentence. The `DP` model generates predictions $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_0, \ldots, \hat{\mathbf{x}}_i, \ldots, \hat{\mathbf{x}}_j, \ldots, \hat{\mathbf{x}}_T]$. We only use its predictions at time-steps corresponding to L2 tokens since the L2 words are those we want to update (Eq 5.16).

$$\begin{aligned}\mathbf{X}^1 &= \text{DP}(\mathbf{X}^0)\\ &\quad \text{Where, } \mathbf{X}^0 = [\mathbf{x}_1, \ldots, \mathbf{F}_{f_i}, \ldots, \mathbf{F}_{f_j}, \ldots, \mathbf{x}_T]\\ &\qquad\qquad \mathbf{X}^1 = [\mathbf{x}_1, \ldots, \hat{\mathbf{x}}_i^1, \ldots, \hat{\mathbf{x}}_j^1, \ldots, \mathbf{x}_T] \qquad (5.16)\\ \mathbf{X}^k &= \text{DP}(\mathbf{X}^{k-1}) \quad \forall\, 0 \le k < K-1 \qquad (5.17)\end{aligned}$$

where $\mathbf{X}^1$ contains predictions at i and j and the original L1 word-embeddings in other positions. We then pass $\mathbf{X}^1$ as input again to the `DP` model. This is executed for K iterations (Eq 5.17). With each iteration, our hope is that the `DP` model's predictions $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ get refined by influencing each other and result in embeddings that are well-suited to the sentence context. A similar style of imputation has been studied for one dimensional time-series data by Zhou and Huang (2018). Finally, after $K-1$ iterations, we use the predictions of $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ from $\mathbf{X}^K$ to update the L2 word-embeddings in $\mathbf{F}$ corresponding to the L2 tokens f_i and f_j. η was set to 0.3 and the number of iterations $K = 5$.

$$\mathbf{F}_{f_i} \leftarrow (1-\eta)\mathbf{F}_{f_i} + \eta\hat{\mathbf{x}}_i^K$$
$$\mathbf{F}_{f_j} \leftarrow (1-\eta)\mathbf{F}_{f_j} + \eta\hat{\mathbf{x}}_j^K \tag{5.18}$$


Task:

U digital camera yz u camera that stores pictures in electronic memory instead az film .

U	
yz	
u	
az	

Figure 5.1: A screenshot of a macaronic sentence presented on Mechanical Turk.

5.4 Experiments with Synthetic L2

We first investigate the patterns of word replacement produced by the machine teacher under the influence of the different generic student models and how these replacements affect the guessability of L2 words. To this end, we used the machine teacher to generate macaronic documents and asked MTurk participants to guess the foreign words. Figure 5.1 shows an example screenshot of our guessing interface. The words in blue are L2 words whose meaning (in English) is guessed by MTurk participants. For our study, we created a synthetic L2 language by randomly replacing characters from English word types. This step lets us safely assume that all MTurk participants are "absolute beginners." We tried to ensure that the resulting synthetic words are pronounceable by replacing vowels with vowels, stop-consonants with other stop-consonants, etc. We also inserted or deleted one character from some of the words to prevent the reader from using the length of the synthetic word as a clue.

Metric	**Model**	$\mathbf{r}_{\text{max}} = \mathbf{1}$	$\mathbf{r}_{\text{max}} = \mathbf{4}$	$\mathbf{r}_{\text{max}} = \mathbf{8}$
	`GSM`	0.25	0.31	0.35
Replaced	`uGSM`	0.20	0.25	0.25
	`DP`	0.19	0.22	0.21
	`GSM`	86.00(±0.87)	74.00(±1.10)	55.13(±2.54)
Guess Accuracy	`uGSM`	84.57(±0.56)	73.89(±1.72)	72.83(±1.58)
	`DP`	88.44(±0.73)	81.07(±1.03)	70.85(±1.49)

Table 5.2: Results from MTurk data. The first section shows the percentage of tokens that were replaced with L2 glosses under each condition. The Accuracy section shows the percentage token accuracy of MTurk participants' guesses along with 95% confidence interval calculated via bootstrap resampling.

We studied the three generic student models (`GSM`, `uGSM`, and `DP`) while keeping the rest of the machine teacher's components fixed (i.e. same scoring function and search

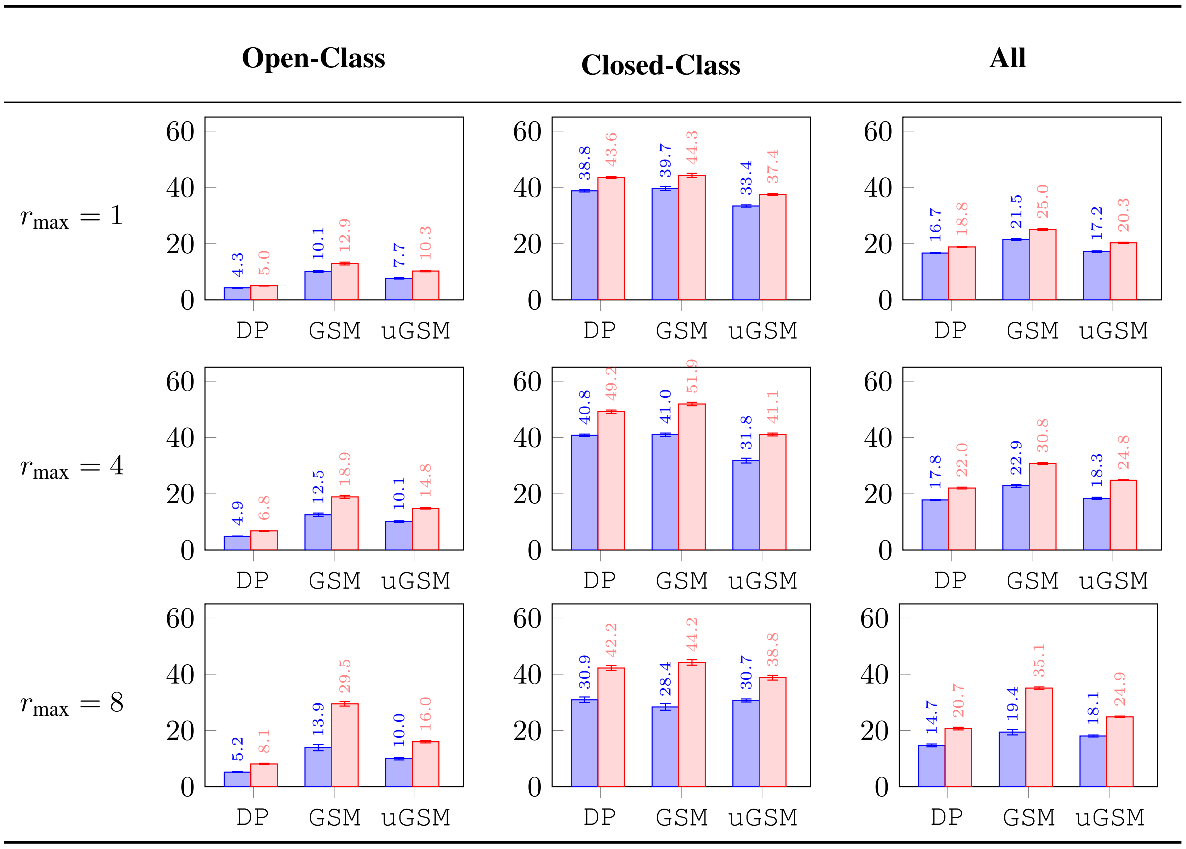


Table 5.3: Results of MTurk results split up by word-class. The y-axis is percentage of tokens belonging to a word-class. The pink bar (right) shows the percentage of tokens (of a particular word-class) that were *replaced* with an L2 gloss. The blue bar (left) and indicates the percentage of tokens (of a particular word-class) that were *guessed correctly* by MTurk participants. Error bars represent 95% confidence intervals computed with bootstrap resampling. For example, we see that only 5.0% (pink) of open-class tokens were replaced into L2 by the DP model at $r_{\max} = 1$ and 4.3% of all open-class tokens were guessed correctly. Thus, even though the guess accuracy for DP at $r_{\max} = 1$ for open-class is high (86%) we can see that participants were not exposed to many open-class word tokens.

algorithms). All three models were constructed to have roughly the same number of L1 parameters ($\approx 20M$). The `uGSM` model used 2 unidirectional LSTM layers instead of a single bidirectional layer. The L1 and L2 word embedding size and the number of recurrent units D were set to 300 for all three models (to match the size of FastText's pretrained embeddings). We trained the three models on the Wikipedia-103 corpus (Merity et al., 2016).[4] All models were trained for 8 epochs using the Adam optimizer (Kingma and Ba, 2014). We limit the L1 vocabulary to the $60k$ most frequent English types.

5.4.1 MTurk Setup

We selected 6 documents from Simple Wikipedia to serve as the input for macaronic content.[5] To keep our study short enough for MTurk, we selected documents that contained $20 - 25$ sentences. A participant could complete up to 6 HITs (Human Intelligence Tasks) corresponding to the 6 documents. Participants were given 25 minutes to complete each HIT (on average, the participants took 12 minutes to complete the HITs). To prevent typos, we used a $20k$ word English dictionary, which includes all the word types from the 6 Simple Wikipedia documents. We provided no feedback regarding the correctness of guesses. We recruited 128 English speaking MTurk participants and obtained 162 responses, with each response encompassing a participant's guesses over a full document.[6] Participants were compensated \$4 per HIT.

[4]FastText pretrained embeddings were trained on more data.
[5]https://dumps.wikimedia.org/simplewiki/20190120/
[6]Participants self-reported their English proficiency, only native or fluent speakers were allowed to participate. Our HITs were only available to participants from the US.

5.4.2 Experiment Conditions

We generated 9 macaronic versions (3 models {GSM,uGSM,DP } in combination with 3 rank thresholds $r_{\max} \in \{1, 4, 8\}$) for each of the 6 Simple Wikipedia documents. For each HIT, an MTurk participant was randomly assigned one of the 9 macaronic versions.

Model	$\mathbf{r}_{\max} = 1$	$\mathbf{r}_{\max} = 8$
GSM	**Hu** Nile (``an-nīl'') **ev** a river **um** Africa. **Up** is **hu** longest river **iñ** Earth (about 6,650 km or 4,132 miles), though other rivers carry more water... Many **ozvolomb** types **iv** **emoner** live in or near **hu** waters **iv** **hu** Nile, including crocodiles, birds, fish **ñb** many others. Not only do animals depend **iñ** **hu** Nile for survival, but also people who live there need **up** **zi** everyday use like washing, as **u** **jopi** supply, keeping crops watered **ñb** other jobs...	**Hu** Nile (``an-nīl'') **ev** **u** river **um** Africa. **Up** **ev** the longest river on Earth (about 6,650 km or 4,132 miles), though other rivers carry more water... **Emu** **ozvolomb** types of **emoner** live **um** or **iul** the waters of **hu** **Uro**, including crocodiles, **ultf**, **yvh** and **emu** others. **Ip** only do animals depend **iñ** the Nile **zi** survival, but also **daudr** who live there need **up** **zi** everyday use like washing, **ez** a **jopi** supply, keeping crops watered **ñb** other jobs...

Table 5.4: Portions of one of our Simple Wikipedia articles. The document has been converted into a macaronic document by the machine teacher using the GSM model.

Tables 5.4 to 5.6 shows the output for the GSM, uGSM and DP generic student models at two settings of r_{max} for one of the documents. In these experiments we use a synthetic L2 language.

Model	$\mathbf{r}_{\text{max}} = 1$	$\mathbf{r}_{\text{max}} = 8$
uGSM	The Nile (``an-nīl'') **ev** a river **um** Africa. It **ev** **hu** longest river on Earth (about 6,650 km or 4,132 miles), though other rivers carry more **jopi**... Many different **pita** of **emoner** live in or near **hu** waters **iv** **hu** Nile, including crocodiles, **ultf**, fish and many others. Not **mru** do **emoner** depend **iñ** **hu** Nile for survival, but also people who live there need it for everyday use like washing, as a **jopi** supply, keeping crops watered **ñb** other jobs...	**Hu** Nile (``an-nīl'') **ev** **u** river **um** Africa. **Up** **ev** the longest river **iñ** Earth (about 6,650 km or 4,132 miles), though other rivers carry more **jopi**... Many different **pita** of **emoner** live **um** or near **hu** waters **iv** **hu** Nile, including crocodiles, **ultf**, fish and many others. Not **mru** do **emoner** depend on the Nile for survival, **id** also people who live there need it **zi** everyday use like washing, as **u** water supply, keeping crops watered **ñb** other jobs...

Table 5.5: Portions of one of our Simple Wikipedia articles. The document has been converted into a macaronic document by the machine teacher using the uGSM model.

Model	$\mathbf{r}_{\text{max}} = 1$	$\mathbf{r}_{\text{max}} = 8$
DP	**Hu** Nile (``an-nīl'') **ev** a river **um** Africa. **Up** **ev** **hu** longest river on Earth (about 6,650 km or 4,132 miles), though other rivers carry more water... Many different types **iv** animals live in or near **hu** waters **iv** **hu** Nile, including crocodiles, birds, fish and many others. Not only do animals depend **iñ** **hu** Nile for survival, but also people who live there need it for everyday use like washing, as **u** water supply, keeping crops watered and other jobs...	**Hu** Nile (``an-nīl'') **ev** a river **um** Africa. **Up** **ev** **hu** longest river on Earth (about 6,650 km or 4,132 miles), though **udho** rivers carry more water... Many different **pita** of animals live in or near **hu** waters of **hu** Nile, including crocodiles, birds, fish and many others. Not **mru** do animals depend **iñ** **hu** Nile **zi** survival, **id** also people who live there need it **zi** everyday use like washing, **ez** a water supply, keeping crops watered and **udho** jobs...

Table 5.6: Portions of one of our Simple Wikipedia articles. The document has been converted into a macaronic document by the machine teacher using the DP generic student model. Only common function words seem to be replaced with their L2 translations.

The two columns show the effect of the rank threshold r_{max}. Note that this macaronic document is 25 sentences long; here, we only show the first 2 sentences and another middle 2 sentences to save space. We see that r_{max} controls the number of L2 words the machine teacher deems guessable, which affects text readability. The increase in L2 words is most noticeable with the GSM model. We also see that the DP model differs from the others by favoring high frequency words almost exclusively. While the GSM and uGSM models similarly replace a number of high frequency words, they also occasionally replace lower frequency word classes like nouns and adjectives (**emoner**, **Emu**, etc.). Table 5.2 summarizes our findings. The first section of 5.2 shows the percentage of tokens that were deemed guessable by our machine teacher. The GSM model replaces more words as r_{max} is increased to 8, but we see that MTurkers had a hard time guessing the meaning of the replaced tokens: their guessing accuracy drops to 55% at $r_{\text{max}} = 8$ with the GSM model. The uGSM model, however, displays a reluctance to replace too many tokens, even as r_{max} was increased to 8.

We further analyzed the replacements and MTurk guesses based on word-class. We tagged the L1 tokens with their part-of-speech and categorized tokens into open or closed class following Universal Dependency guidelines ("Universal Dependencies v1: A Multilingual Treebank Collection.").[7] Table 5.3 summarizes our analysis of model and human behavior when the data is separated by word-class. The pink bars indicate the percentage of tokens replaced per word-class. The blue bars represent the percentage of tokens from a particular word-class that MTurk users *guessed correctly*. Thus, an ideal machine teacher should strive for the highest possible pink bar while ensuring that the blue bar is as close as possible to the pink. Our findings suggest that the uGSM model at $r_{\text{max}} = 8$ and the GSM model at $r_{\text{max}} = 4$ show the desirable properties – high guessing accuracy and more representation of L2 words (particularly open-class words).

[7]https://universaldependencies.org/u/pos/

Metric	**Model**	**Closed**	**Open**
Types Replaced	`random`	59	524
	`GSM`	33	149
Guess Accuracy	`random`	$62.06(\pm 1.54)$	$39.36(\pm 1.75)$
	`GSM`	$\mathbf{74.91(\pm 0.94)}$	$\mathbf{61.96(\pm 1.24)}$

Table 5.7: Results comparing our generic student based approach to a random baseline. The first part shows the number of L2 word types exposed by each model for each word class. The second part shows the average guess accuracy percentage for each model and word class. 95% confidence intervals (in brackets) were computed using bootstrap resampling.

5.4.3 Random Baseline

So far we've compared different generic student models against each other, but is our generic student based approach required at all? How much better (or worse) is this approach compared to a *random* baseline? To answer these questions, we compare the `GSM` with $r_{\text{max}} = 4$ model against a randomly generated macaronic document. As the name suggests, word replacements are decided randomly for the random condition, but we ensure that the number of tokens replaced in each sentence equals that from the `GSM` condition.

We used the 6 Simple Wikipedia documents from §5.4.1 and recruited 64 new MTurk partipants who completed a total of 66 HITs (compensation was $\$4$ per HIT). For each

Model	**Closed**	**Open**
`random`	$9.86(\pm 0.94)$	$4.28(\pm 0.69)$
`GSM`	$\mathbf{35.53(\pm 1.03)}$	$\mathbf{27.77(\pm 1.03)}$

Table 5.8: Results of our L2 learning experiments where MTurk subjects simply read a macaronic document and answered a vocabulary quiz at the end of the passage. The table shows the average guess accuracy percentage along with 95% confidence intervals computed from bootstrap resampling.

HIT, the participant was given either the randomly generated or the `GSM` based macaronic document. Once again, participants were made to enter their guess for each L2 word that appears in a sentence. The results are summarized in Table 5.7.

We find that randomly replacing words with glosses exposes more L2 word types (59 and 524 closed-class and open-class words respectively) while the `GSM` model is more conservative with replacements (33 and 149). However, the random macaronic document is much harder to comprehend, indicated by significantly lower average guess accuracies than those with the `GSM` model. This is especially true for open-class words. Note that Table 5.7 shows the number of word types replaced across all 6 documents.

5.4.4 Learning Evaluation

Our macaronic based approach relies on incidental learning, which states that if a novel word is repeatedly presented to a student with sufficient context, the student will eventually be able to learn the novel word. So far our experiments test MTurk participants on the "guessability" of novel words in context, but not learning. To study if students can actually learn the L2 words, we conduct an MTurk experiment where participants are simply required to read a macaronic document (one sentence at a time). At the end of the document an L2 vocabulary quiz is given. Participants must enter the meaning of every L2 word type they have seen during the reading phase.

Once again, we compare our `GSM` ($r_{\text{max}} = 4$) model against a random baseline using the 6 Simple Wikipedia documents. 47 HITs were obtained from 45 MTurk participants for this experiment. Participants were made aware that there would be a vocabulary quiz at the end of the document. Our findings are summarized in Table 5.8. We find the accuracy of guesses for the vocabulary quiz at the end of the document is considerably lower than guesses with context. However, subjects still managed to retain 35.53% and 27.77% of closed-class and open-class L2 word types respectively. On the other hand, when a random macaronic document was presented to participants, their guess accuracy dropped to 9.86% and 4.28% for closed and open class words respectively. Thus, even though more word types were exposed by the random baseline, fewer words were retained.

Additionally, we would like to investigate how our approach could be extended to enable phrasal learning (which should consider word-ordering differences between the L1 and L2). As the `GSM` and `uGSM` models showed the most promising results in our experiments, we believe these models could serve as the baseline for future work.

5.5 Spelling-Aware Extension

So far, our generic student model ignores the fact that a novel word like **`Afrika`** is guessable simply by its spelling similarity to `Africa`. Thus, we augment the generic student model to use character n-grams. We choose the bidirectional generic student model for our spelling-aware extension based on the pilot experiments detailed in §5.4.2. In addition to an embedding per word type, we learn embeddings for character n-gram types that appear in our L1 corpus. The row in $\mathbf{E}$ for a word w is now parameterized as:

$$\tilde{\mathbf{E}} \cdot \tilde{\mathbf{w}} + \sum_{n} \tilde{\mathbf{E}}^{\mathbf{n}} \cdot \tilde{\mathbf{w}}^{n} \frac{1}{\mathbf{1} \cdot \tilde{\mathbf{w}}^{n}} \tag{5.19}$$

where $\tilde{\mathbf{E}}$ is the full-word embedding matrix and $\tilde{\mathbf{w}}$ is a one-hot vector associated with the word type w, $\tilde{\mathbf{E}}^{\mathbf{n}}$ is a character n-gram embedding matrix and $\tilde{\mathbf{w}}^{n}$ is a *multi*-hot vector associated with all the character n-grams for the word type w. For each n, the summand gives the average embedding of all n-grams in w (where $\mathbf{1} \cdot \tilde{\mathbf{w}}^{n}$ counts these n-grams). We set n to range from 3 to 4 (see §5.7). This formulation is similar to previous sub-word based embedding models (Wieting et al., 2016; Bojanowski et al., 2017).

Similarly, the embedding of an L2 word w is parameterized as

$$\tilde{\mathbf{F}} \cdot \tilde{\mathbf{w}} + \sum_n \tilde{\mathbf{F}}^{\mathbf{n}} \cdot \tilde{\mathbf{w}}^n \frac{1}{\mathbf{1} \cdot \tilde{\mathbf{w}}^n} \tag{5.20}$$

Crucially, we initialize $\tilde{\mathbf{F}}^{\mathbf{n}}$ to $\mu\tilde{\mathbf{E}}^{\mathbf{n}}$ (where $\mu > 0$) so that L2 words can inherit part of their initial embedding from similarly spelled L1 words: $\tilde{\mathbf{F}}^{\mathbf{4}}{}_{\texttt{Afri}} := \mu\tilde{\mathbf{E}}^{\mathbf{4}}{}_{\texttt{Afri}}$.[8] But we allow $\tilde{\mathbf{F}}^{\mathbf{n}}$ to diverge over time in case an n-gram functions differently in the two languages. In the same way, we initialize each row of $\tilde{\mathbf{F}}$ to the corresponding row of $\mu \cdot \tilde{\mathbf{E}}$, if any, and otherwise to $\mathbf{0}$. Our experiments set $\mu = 0.2$ (see §5.7). We refer to this spelling-aware extension to `GSM` as `sGSM`.

5.5.1 Scoring L2 embeddings

Did the simulated student learn correctly and usefully? Let $\mathcal{P}$ be the "reference set" of all (L1 word, L2 gloss) pairs from *all tokens in the entire document*. We assess the machine teacher's success by how many of these pairs the simulated student has learned. (The student may even succeed on some pairs that it has never been shown, thanks to n-gram clues.) Specifically, we measure the "goodness" of the updated L2 word embedding matrix $\mathbf{F}$. For each pair $p = (e, f) \in \mathcal{P}$, sort all the words in the entire L1 vocabulary according to their cosine similarity to the L2 word f, and let r_p denote the rank of e. For example, if the student had managed to learn a matrix $\mathbf{F}$ whose embedding of f exactly equalled $\mathbf{E}$'s embedding of e, then r_p would be 1. We then compute a mean reciprocal rank (MRR) score

[8]We set $\mu = 0.2$ based on findings from our hyperparameter search (see §5.7).

of $\mathbf{F}$:

$$\text{MRR}(\mathbf{F}) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \left(\frac{1}{r_p} \textbf{ if } r_p \leq r_{\text{max}} \textbf{ else } 0 \right) \tag{5.21}$$

We set $r_{\text{max}} = 4$ based on our pilot study. This threshold has the effect of only giving credit to an embedding of f such that the correct e is in the simulated student's top 4 guesses. As a result, §5.5.2's machine teacher focuses on introducing L2 tokens whose meaning can be deduced *rather accurately* from their single context (together with any prior exposure to that L2 type). This makes the macaronic text comprehensible for a human student, rather than frustrating to read. In our pilot study we found that r_{max} substantially improved human learning.

5.5.2 Macaronic Configuration Search

Our current machine teacher produces the macaronic document greedily, one sentence at a time. Actual documents produced are shown in **??**.

Let $\mathbf{F}^{\text{prev}}$ be the student model's embedding matrix after the reading the first $n-1$ macaronic sentences. We evaluate a candidate next sentence **x** by the score $\text{MRR}(\mathbf{F})$ where $\mathbf{F}$ maximizes (5.5) and is thus the embedding matrix that the student would arrive at after reading **x** as the n^{th} macaronic sentence.

We use best-first search to seek a high-scoring **x**. A search state is a pair $(i, \mathbf{x})$ where **x** is a macaronic configuration (Table 5.1) whose first i tokens may be either L1 or L2, but whose remaining tokens are still L1. The state's score is obtained by evaluating **x** as described above. In the initial state, $i = 0$ and **x** is the n^{th} sentence of the original L1 document. The state $(i, \mathbf{x})$ is a final state if $i = |\mathbf{x}|$. Otherwise its two successors are $(i+1, \mathbf{x})$ and $(i+1, \mathbf{x}')$, where $\mathbf{x}'$ is identical to **x** except that the $(i+1)^{\text{th}}$ token has been replaced by its L2 gloss. The search algorithm maintains a priority queue of states sorted by score. Initially, this contains only the initial state. A step of the algorithm consists of popping the highest-scoring state and, if it is not final, replacing it by its two successors. The queue is then pruned back to the top 8 states. When the queue becomes empty, the algorithm returns the configuration **x** from the highest-scoring final state that was ever popped.

5.6 Experiments with real L2

Does our machine teacher generate useful macaronic text? To answer this, we measure whether *human* students (i) comprehend the L2 words in context, and (ii) retain knowledge of those L2 words when they are later seen without context.

We assess (i) by displaying each successive sentence of a macaronic document to a human student and asking them to guess the L1 meaning for each L2 token f in the sentence. For a given machine teacher, all human subjects saw the same macaronic document, and each subject's comprehension score is the average quality of their guesses on all the L2 tokens presented by that teacher. A guess's quality $q \in [0, 1]$ is a thresholded cosine similarity between the embeddings[9] of the guessed word $\hat{e}$ and the original L1 word e:

[9]Here we used pretrained word embeddings from Mikolov et al. (2018), in order to measure actual semantic similarity.

$q = \text{cs}(e, \hat{e})$ **if** $\text{cs}(e, \hat{e}) \geq \tau$ **else** 0. Thus, $\hat{e} = e$ obtains $q = 1$ (full credit), while $q = 0$ if the guess is "too far" from the truth (as determined by τ).

To assess (ii), we administer an L2 vocabulary quiz after having human subjects *simply* read a macaronic passage (without any guessing as they are reading). They are then asked to guess the L1 translation of each L2 word type that appeared at least once in the passage. We used the same guess quality metric as in (i).[10] This tests if human subjects naturally learn the meanings of L2 words, in informative contexts, well enough to later translate them out of context. The test requires only short-term retention, since we give the vocabulary quiz immediately after a passage is read.

We compared results on macaronic documents constructed with the generic student model (`GSM`), its spelling-aware variant (`sGSM`), and a random baseline. In the baseline, tokens to replace are randomly chosen while ensuring that each sentence replaces the same number of tokens as in the `GSM` document. This ignores context, spelling, and prior exposures as reasons to replace a token.

Our evaluation was aimed at native English (L1) speakers learning Spanish or German (L2). We recruited L2 "students" on Amazon Mechanical Turk (MTurk). They were absolute beginners, selected using a placement test and self-reported L2 ability.

[10]If multiple L1 types e were glossed in the document with this L2 type, we generously use the e that maximizes $\text{cs}(e, \hat{e})$.

L2	Model	Closed-class	Open-class
	random	$0.74 \pm 0.0126(54)$	$0.61 \pm 0.0134(17)$
Es	`GSM`	$0.72 \pm 0.0061(54)$	$0.70 \pm 0.0084(17)$
	`sGSM`	$\mathbf{0.82 \pm 0.0038}(41)$	$\mathbf{0.80 \pm 0.0044}(21)$
	random	$0.59 \pm 0.0054(34)$	$0.38 \pm 0.0065(13)$
De	`GSM`	$0.80 \pm 0.0033(34)$	$0.78 \pm 0.0056(13)$
	`sGSM`	$\mathbf{0.82 \pm 0.0063}(33)$	$\mathbf{0.79 \pm 0.0062}(14)$

Table 5.9: Average token guess quality ($\tau = 0.6$) in the comprehension experiments. The $\pm$ denotes a 95% confidence interval computed via bootstrap resampling of the set of human subjects. The % of L1 tokens replaced with L2 glosses is in parentheses. §5.8 evaluates with other choices of τ.

5.6.1 Comprehension Experiments

We used the first chapter of Jane Austen's "Sense and Sensibility" for Spanish, and the first 60 sentences of Franz Kafka's "Metamorphosis" for German. Bilingual speakers provided the L2 glosses (see §5.9 for examples).

For English-Spanish, 11, 8, and 7 subjects were assigned macaronic documents generated with `sGSM`, `GSM`, and the random baseline, respectively. The corresponding numbers for English-German were 12, 7 and 7. A total of 39 subjects were used in these experiments (some subjects did both languages). They were given 3 hours to complete the entire document (average completion time was ≈ 1.5 hours) and were compensated $\$10$.

Table 5.9 reports the mean comprehension score over all subjects, broken down into comprehension of function words (closed-class POS) and content words (open-class POS).[11] For Spanish, the `sGSM`-based teacher replaces *more* content words (but fewer function words), and furthermore the replaced words in both cases are *better understood* on average, which we hope leads to more engagement and more learning. For German, by contrast, the number of words replaced does not increase under `sGSM`, and comprehension only improves marginally. Both `GSM` and `sGSM` do strongly outperform the random baseline. But the `sGSM`-based teacher only replaces a few additional cognates (**`hundert`** but not **`Mutter`**), apparently because English-German cognates do not exhibit large *exact* character n-gram overlap. We hypothesize that character skip n-grams might be more appropriate for

[11] https://universaldependencies.org/u/pos/

L2	**Model**	**Closed-class**	**Open-class**
	random	$0.47 \pm 0.0058(60)$	$0.40 \pm 0.0041(46)$
Es	`GSM`	$0.48 \pm 0.0084(60)$	$0.42 \pm 0.0105(15)$
	`sGSM`	$\mathbf{0.52 \pm 0.0054}(47)$	$\mathbf{0.50 \pm 0.0037}(24)$

Table 5.10: Average type guess quality ($\tau = 0.6$) in the retention experiment. The % of L2 gloss types that were shown in the macaronic document is in parentheses. §5.8 evaluates with other choices of τ.

English-German.

5.6.2 Retention Experiments

For retention experiments we used the first 25 sentences of our English-Spanish dataset. New participants were recruited and compensated \$5. Each participant was assigned a macaronic document generated with the `sGSM`, `GSM` or random model (20, 18, and 22 participants respectively). As Table 5.10 shows, `sGSM`'s advantage over `GSM` on comprehension holds up on retention. On the vocabulary quiz, students correctly translated > 30 of the 71 word types they had seen (Table 5.15), and more than half when near-synonyms earned partial credit (Table 5.10).

5.7 Hyperparameter Search

We tuned the model hyperparameters by hand on separate English-Spanish data, namely the second chapter of “Sense and Sensibility,” equipped with glosses. Hyperparameter tuning results are reported in this appendix. All other English-Spanish results in the paper are on the first chapter of “Sense and Sensibility,” which was held out for testing. We might have improved the results on English-German by tuning separate hyperparameters for that setting.

The tables below show the effect of different hyperparameter choices on the quality MRR$(\mathbf{F})$ of the embeddings learned by the simulated student. Recall from §5.5.1 that the MRR score evaluates $\mathbf{F}$ using all glosses, not just those used in a particular macaronic document. Thus, it is comparable across the different macaronic documents produced by different machine teachers.

QueueSize (§5.5.2) affects only how hard the machine teacher searches for macaronic sentences that will help the simulated student. We find that larger QueueSize is in fact valuable.

The other choices (Model, n-grams, μ) affect how the simulated student actually learns. The machine teacher then searches for a document that will help that particular simulated student learn as many of the words in the reference set as possible. Thus, the MRR score is high to the extent that the simulated student “can be successfully taught.” By choosing hyperparameters that achieve a high MRR score, we are assuming that human students are adapted (or can adapt online) to be teachable.

The scale factor μ (used only for sGSM) noticeably affects the macaronic document generated by the machine teacher. Setting it high ($\mu = 1.0$) has a adverse effect on the MRR score. Table 5.11 shows how the MRR score of the simulated student (§5.5.1) varies according to the student model's μ value. Tables 5.12 and 5.13 show the result of the same hyperparameter sweep on the number of L1 word tokens and types replaced with L2 glosses.

Note that μ only affects initialization of the $\mathcal{F}$ parameters. Thus, with $\mu = 0$, the L2 word and subword embeddings are initialized to $\mathbf{0}$, but the simulated sGSM student still has the ability to learn subword embeddings for both L1 and L2. This allows it to beat the simulated GSM student.

We see that for sGSM, $\mu = 0.2$ results in replacing the most words (both types and tokens), and also has very nearly the highest MRR score. Thus, for sGSM, we decided to use $\mu = 0.2$ and allow both 3-gram and 4-gram embeddings.

5.8 Results Varying τ

L2	τ	Model	Closed-class	Open-class
		rand	$0.81 \pm 0.0084(54)$	$0.72 \pm 0.0088(17)$
	0.0	GSM	$0.80 \pm 0.0045(54)$	$0.79 \pm 0.0057(17)$
		sGSM	$0.86 \pm 0.0027(41)$	$0.84 \pm 0.0032(21)$
		rand	$0.81 \pm 0.0085(54)$	$0.72 \pm 0.0089(17)$
	0.2			

		GSM	$0.80 \pm 0.0045(54)$	$0.79 \pm 0.0057(17)$
		sGSM	$0.86 \pm 0.0027(41)$	$0.84 \pm 0.0033(21)$
		rand	$0.79 \pm 0.0101(54)$	$0.66 \pm 0.0117(17)$
	0.4	GSM	$0.76 \pm 0.0057(54)$	$0.75 \pm 0.0071(17)$
Es		sGSM	$0.84 \pm 0.0033(41)$	$0.82 \pm 0.0039(21)$
		random	$0.74 \pm 0.0126(54)$	$0.61 \pm 0.0134(17)$
	0.6	GSM	$0.72 \pm 0.0061(54)$	$0.70 \pm 0.0084(17)$
		sGSM	$0.82 \pm 0.0038(41)$	$0.80 \pm 0.0044(21)$
		rand	$0.62 \pm 0.0143(54)$	$0.46 \pm 0.0124(17)$
	0.8	GSM	$0.59 \pm 0.0081(54)$	$0.58 \pm 0.0106(17)$
		sGSM	$0.71 \pm 0.0052(41)$	$0.67 \pm 0.0062(21)$
		rand	$0.62 \pm 0.0143(54)$	$0.45 \pm 0.0124(17)$
	1.0	GSM	$0.59 \pm 0.0081(54)$	$0.55 \pm 0.0097(17)$
		sGSM	$0.70 \pm 0.0052(41)$	$0.64 \pm 0.0063(21)$
		random	$0.70 \pm 0.0039(34)$	$0.56 \pm 0.0046(13)$
	0.0	GSM	$0.85 \pm 0.0023(34)$	$0.84 \pm 0.0039(13)$
		sGSM	$0.87 \pm 0.0045(33)$	$0.84 \pm 0.0044(14)$
		random	$0.69 \pm 0.0042(34)$	$0.56 \pm 0.0047(13)$
	0.2	GSM	$0.85 \pm 0.0024(34)$	$0.84 \pm 0.0039(13)$
		sGSM	$0.87 \pm 0.0046(33)$	$0.84 \pm 0.0044(14)$

		random	$0.64 \pm 0.0052(34)$	$0.45 \pm 0.0064(13)$
	0.4	GSM	$0.83 \pm 0.0029(34)$	$0.81 \pm 0.0045(13)$
De		sGSM	$0.84 \pm 0.0055(33)$	$0.81 \pm 0.0054(14)$
		random	$0.59 \pm 0.0054(34)$	$0.38 \pm 0.0065(13)$
	0.6	GSM	$0.80 \pm 0.0033(34)$	$0.78 \pm 0.0056(13)$
		sGSM	$0.82 \pm 0.0063(33)$	$0.79 \pm 0.0062(14)$
		random	$0.45 \pm 0.0058(34)$	$0.25 \pm 0.0061(13)$
	0.8	GSM	$0.72 \pm 0.0037(34)$	$0.66 \pm 0.0081(13)$
		sGSM	$0.75 \pm 0.0079(33)$	$0.65 \pm 0.0077(14)$
		random	$0.45 \pm 0.0058(34)$	$0.24 \pm 0.0061(13)$
	1.0	GSM	$0.71 \pm 0.0040(34)$	$0.63 \pm 0.0082(13)$
		sGSM	$0.75 \pm 0.0079(33)$	$0.63 \pm 0.0081(14)$

Table 5.14: An expanded version of Table 5.9 (human comprehension experiments), reporting results with various values of τ.

A more comprehensive variant of Table 5.9 is given in Table 5.14. This table reports the same human-subjects experiments as before; it only varies the measure used to assess the quality of the humans' guesses, by varying the threshold τ. Note that $\tau = 1$ assesses exact-match accuracy, $\tau = 0.6$ as in Table 5.9 corresponds roughly to synonymy (at least for content words), and $\tau = 0$ assesses average *unthresholded* cosine similarity. We find that `sGSM` consistently outperforms both `GSM` and the random baseline over the entire range of τ.

Model	n**-grams**	**QueueSize**	**Scale Factor** μ					
			1.0	0.4	0.2	0.1	0.05	0.0
`sGSM`	2,3,4	1	0.108	0.207	0.264	0.263	0.238	0.175
`sGSM`	3,4	1	0.113	0.199	0.258	0.274	0.277	0.189
`sGSM`	3,4	4	-	-	0.267	0.286	-	-
`sGSM`	3,4	8	-	-	0.288	0.292	-	-
`GSM`	$\emptyset$	1						0.159
`GSM`	$\emptyset$	4						0.171
`GSM`	$\emptyset$	8						0.172

Table 5.11: MRR scores obtained with different hyperparameter settings.

As we get closer to exact match, the random baseline suffers the largest drop in performance.

Similarly, Table 5.15 shows a expanded version of the retention results in Table 5.10. The gap between the models is smaller on retention than it was on comprehension. However, again `sGSM` $>$ `GSM` $>$ random across the range of τ. We find that for function words, the random baseline performs as well as `GSM` as τ is increased. For content words, however, the random baseline falls faster than `GSM`.

We warn that the numbers are not genuinely comparable across the 3 models, because each model resulted in a different document and thus a different vocabulary quiz. Our human subjects were asked to translate just the L2 words in the document they read. In

Model	**n-grams**	**QueueSize**	**Scale Factor μ**					
			1.0	0.4	0.2	0.1	0.05	0.0
sGSM	2,3,4	1	149	301	327	275	201	247
sGSM	3,4	1	190	340	439	399	341	341
sGSM	3,4	4	-	-	462	440	-	-
sGSM	3,4	8	-	-	478	450	-	-
GSM	∅	1						549
GSM	∅	4						557
GSM	∅	8						530

Table 5.12: Number of L1 tokens replaced by L2 glosses under different hyperparameter settings.

particular, sGSM taught *fewer* total types (71) than GSM (75) or the random baseline (106). All that Table 5.15 shows is that it taught its chosen types better (on average) than the other methods taught their chosen types.

5.9 Macaronic Examples

Below, we display the actual macaronic documents generated by our methods. First few paragraphs of "Sense and Sensibility" with the sGSM model using $\mu = 0.2$, 3- and 4-grams, priority queue size of 8, and $r_{\text{max}=4}$ are shown below:

Model	n-grams	QueueSize	Scale Factor μ					
			1.0	0.4	0.2	0.1	0.05	0.0
sGSM	2,3,4	1	39	97	121	106	75	88
sGSM	3,4	1	44	97	125	124	112	99
sGSM	3,4	4	-	-	124	127	-	-
sGSM	3,4	8	-	-	145	129	-	-
GSM	∅	1						106
GSM	∅	4						111
GSM	∅	8						114

Table 5.13: Number of distinct L2 word types present in the macaronic document under different hyperparameter settings.

Sense **y** Sensibility

La family **de** Dashwood **llevaba** long been settled **en** **Sussex**. Their estate **era** large, and their **residencia** was **en** Norland **Park**, in **el** centre **de** their property, where, for **muchas** generations, they **habían** lived **en** so respectable a manner as to engage **el** **general** good opinion of **los** surrounding acquaintance. **El** late owner **de** this **propiedad** was **un** single man, **que** lived to **una** very advanced age, **y** **que** **durante** many years of his life, had a **constante** companion and housekeeper in **su** sister. But **ella** death,

que happened ten **años antes** his own, produced a great alteration in **su** home; for to supply her loss, he invited and received into his house **la** family of **su sobrino señor** Henry Dashwood, the legal inheritor of the Norland estate, and the person to whom he intended to bequeath it. **En la** society **de su** nephew **y** niece, **y** their children, **el** old Gentleman's days **fueron** comfortably spent. **Su** attachment **a** them all increased. The constant attention **de** Mr. **y** Mrs. Henry Dashwood to **sus** wishes, **que** proceeded **no** merely from interest, but from goodness of heart, **dio** him every degree **de** solid comfort which **su** age **podía** receive; and **la** cheerfulness of the children added a relish to his **existencia**.

By **un** former marriage, Mr. Henry **Dashwood tenía** one son: by **su** present lady, three **hijas**. **El** son, **un** steady respectable young man, was amply provided for **por** the **fortuna de** his mother, which **había** been large, **y** half of which devolved on him on his coming of **edad**. **Por su** own **matrimonio**, likewise, which happened soon **después**, he added **a** his wealth. To him therefore **la** succession **a la** Norland estate **era no** so really **importante** as to his sisters; **para** their **fortuna**, independent **de** what **pudiera** arise **a ellas** from **su** father's inheriting that **propiedad**, could **ser** but small. **Su** mother had nothing, **y** their father only seven **mil** pounds **en** his own **disposición**; for **la** remaining moiety of his first **esposa**'s fortune was also secured to her child, and **él tenía sólo** a life-**interés** in it.

el anciano gentleman died: his will was read, and like almost **todo** other will, **dio** as **tanto** disappointment as pleasure. He **fue** neither so unjust, **ni** so ungrateful, as **para** leave his estate **de** his nephew; --but he left it **a** him **en** such terms as destroyed half the **valor de el** bequest. Mr. Dashwood **había** wished for it more **por el** sake of his **esposa** and **hijas** than for himself or **su** son; --but **a** his son, **y su** son's son, **un** child **de** four **años** old, it **estaba** secured, in **tal** a way, as **a** leave **a** himself no power **de** providing **por** those **que** were most dear **para** him, and who most **necesitaban** a **provisión** by any charge on **la** estate, or **por** any sale **de** its valuable woods. **El** whole **fue** tied **arriba para** the **beneficio de** this child, **quien**, in occasional visits with his **padre** and mother at Norland, had **tan** far gained on **el** affections **de** his uncle, by such attractions as are by no means unusual in children of two **o** three years old; **una** imperfect **articulación**, an earnest desire of having his own way, many cunning tricks, and a great deal of noise, as to outweigh all the value **de** all the attention which, for years, **él había** received from his niece and **sus** daughters. He meant **no a ser** unkind, however, **y**, **como** a mark **de** his affection for **las** three

```
girls, he left ellas un mil libras a-piece.
```

Next, the first few paragraphs of "Sense and Sensibility" with the GSM model using priority queue size of 8 and $r_{\text{max}=4}$.

Sense **y** Sensibility

La family **de** Dashwood **llevaba** long been settled **en** Sussex. **Su** estate **era** large, and **su** residence **estaba en** Norland Park, in **el** centre **de** their property, where, **por** many generations, they had lived in so respectable **una** manner as **a** engage **el** general good opinion **de los** surrounding acquaintance. **El** late owner **de esta** estate was **un** single man, **que** lived to **una** very advanced age, **y** who **durante** many years **de su existencia**, had **una** constant companion **y** housekeeper in his sister. But **ella** death, **que** happened ten years **antes su** own, produced a great alteration in **su** home; for **para** supply her loss, **él** invited and received into his house **la** family **de su** nephew Mr. Henry Dashwood, the legal inheritor **de** the Norland estate, and the person to whom **se** intended to bequeath it. In the society **de su** nephew and niece, and **sus** children, **el** old Gentleman's days **fueron** comfortably spent. **Su** attachment **a** them all increased. **La** constant attention **de** Mr. **y** Mrs. Henry Dashwood to **sus** wishes, which proceeded not merely from interest, but **de** goodness **de** heart, **dio** him every degree **de** solid comfort **que** his age could receive; **y la** cheerfulness of the children added **un** relish **a su** existence.

By **un** former marriage, Mr. Henry Dashwood **tenía** one son:

by **su** present lady, three **hijas**. **El** son, **un** steady respectable **joven** man, was amply provided for **por la** fortune **de su madre**, **que había** been large, **y** half **de cuya** devolved on him on **su** coming **de edad**. By **su** own marriage, likewise, **que** happened soon **después**, he added **a su** wealth. **Para** him therefore **la** succession **a la** Norland estate was **no** so really **importante** as to his sisters; **para** their fortune, independent **de** what **pudiera** arise **a** them from **su** father's inheriting that property, could **ser** but small. **Su madre** had nothing, **y su padre** only **siete** thousand pounds in **su** own disposal; for **la** remaining moiety of his first wife's fortune **era** also secured **a su** child, **y él** had only **una** life-interest in **ello**.

el old gentleman died: **su** will was read, **y** like almost every **otro** will, gave as **tanto** disappointment as pleasure. He **fue** neither so unjust, nor so ungrateful, as to leave **su** estate from his nephew; --but he left it to him **en** such terms **como** destroyed half the **valor** of the bequest. Mr. Dashwood **había** wished for it **más** for **el** sake **de su** wife and daughters than **para** himself or **su hijo**; --but **a su hijo**, **y** his son's **hijo**, **un** child **de** four **años** old, it **estaba** secured, **en tal un** way, as **a** leave **a** himself no power of providing for **aquellos** who were most dear **para** him, **y** who most needed **un** provision by any charge **sobre la** estate, or **por** any sale **de** its valuable woods. **El** whole was tied **arriba** for

el benefit **de** this child, **quien**, **en ocasionales** visits with his father and mother at Norland, had **tan** far gained on the affections of his uncle, by such attractions as are **por** no means unusual in children of two or three years old; an imperfect articulation, an earnest desire of having his own way, many cunning tricks, and a **gran** deal of noise, as to outweigh **todo** the value of all the attention which, for years, he had received from his niece and her daughters. He meant **no a ser** unkind, however, **y**, **como una** mark **de su** affection **por las** three girls, he left them **un mil** pounds a-**pieza**.

First few paragraphs of “The Metamorphosis” with the sGSM model using $\mu = 0.2$, 3- and 4-grams, priority queue size of 8, and $r_{\text{max}=4}$.

Metamorphosis

One morning, **als** Gregor Samsa woke from troubled dreams, he **fand** himself transformed in **seinem** bed into **einem** horrible vermin. He lay on **seinem** armour-like back, **und** if **er** lifted **seinen** head a little he **konnte** see his brown belly, slightly domed **und** divided by arches into stiff sections. The bedding **war** hardly able **zu** cover it **und** seemed ready **zu** slide off any moment. His many legs, pitifully thin compared **mit** the size **von dem** rest **von** him, waved about helplessly as **er** looked.

''What's happened to **mir**?'' he thought. His room, **ein** proper human room although **ein** little too small, lay peacefully between its four familiar walls. **Eine** collection of textile samples lay spread out on the table - Samsa was **ein** travelling salesman - and above it there hung a picture **das** he had recently cut out **von einer** illustrated magazine **und** housed **in einem** nice, gilded frame. It showed **eine** lady fitted out **mit** a fur hat **und** fur boa who sat upright, raising **einen** heavy fur muff **der** covered the whole **von** her lower arm towards **dem** viewer.

Gregor then turned **zu** look out the window at the dull weather. Drops **von** rain could **sein** heard hitting the pane, **welche** made him **fühlen** quite sad. ''How about if **ich** sleep **ein** little bit longer and forget all **diesen** nonsense,'' he thought, **aber** that was something **er** was unable **zu** do because he **war** used **zu** sleeping **auf** his right, **und** in **seinem** present state couldn't **bringen** into that position. However hard he threw **sich** onto **seine** right, he always rolled **zurück** to where he was. He must **haben** tried it a **hundert** times, shut **seine** eyes so **dass er** wouldn't **haben zu** look at **die** floundering legs, and only stopped when **er** began **zu fühlen einen** mild, dull pain there **das** he **hatte** never felt before.

``**Ach**, God,'' he thought, ``what a strenuous career it is **das** I've chosen! Travelling day in **und** day out. Doing business like **diese** takes **viel** more effort than doing your own business at home, **und auf** top of that there's the curse **des** travelling, worries **um** making train connections, bad **und** irregular food, contact **mit** different people all the time so that **du** can **nie** get to know anyone or become friendly **mit ihnen**. It can **alles** go **zum** Hell!'' He felt a slight itch up **auf** his belly; pushed himself slowly up **auf** his back towards **dem** headboard so **dass** he **konnte** lift his head better; **fand** where **das** itch was, **und** saw that **es** was covered **mit vielen** of little **weißen** spots which he didn't know what to make of; **und als** he **versuchte** to **fühlen** the place with one **von seinen** legs he drew it quickly back because as soon as he touched it he was overcome **von** a cold shudder.

First few paragraphs of “The Metamorphosis” with the GSM model using priority queue size of 8 and $r_{\max=4}$.

Metamorphosis

One morning, **als** Gregor Samsa woke from troubled dreams, he **fand** himself transformed **in** his bed into **einem** horrible vermin. **Er** lay on **seinem** armour-like back, **und** if **er** lifted his head a little **er** could see **seinen** brown belly, slightly domed **und** divided by arches into stiff **teile**. **das** bedding was hardly **fähig** to cover **es** **und** seemed ready **zu** slide off any moment. His many legs, pitifully thin compared **mit** the size **von** **dem** rest **von** him, waved about helplessly **als** **er** looked.

``What's happened to **mir**?'' **er** thought. His room, **ein** proper human room although **ein** little too **klein**, lay peacefully between **seinen** four familiar walls. **Eine** collection of textile samples lay spread out on the table - Samsa was **ein** travelling salesman - **und** above it there hung a picture that **er** had recently cut **aus** of **einer** illustrated magazine **und** housed in **einem** nice, gilded frame. **Es** showed a lady fitted out with a fur hat and fur boa who **saß** upright, raising a heavy fur muff **der** covered the whole of her lower arm towards **dem** viewer.

Gregor then turned **zu** look out the window at the dull weather. Drops **von** rain could **sein** heard hitting the pane, which **machte** him feel **ganz** sad. ``How about if **ich** sleep **ein** little bit longer and forget all **diesen** nonsense,’’ he thought, but that **war** something he was unable to **tun** because **er** was used to sleeping **auf** his right, and in his present state couldn’t get into that position. However hard he **warf** himself onto **seine** right, he always rolled **zurück** to **wo** he was. **Er** must **haben** tried it **ein** hundred times, shut **seine** eyes so **dass** he wouldn’t **haben** to **sehen** at **die** floundering legs, **und** only stopped when he **begann** to feel **einen** mild, dull pain there that he **hatte nie** felt before.

‘‘**Ach**, God,’’ he thought, ‘‘what a strenuous career it **ist** that I’ve chosen! Travelling day in **und** day **aus**. Doing business like **diese** takes much **mehr** effort than doing your own business at home, **und** on **oben** of that there’s **der** curse of travelling, worries **um** making train connections, bad and irregular food, contact with different people all the time so that you **kannst nie** get to know anyone or become friendly with **ihnen**. It **kann** all go to **Teufel**!’’ He felt **ein** slight itch up **auf seinem** belly; pushed himself slowly up **auf** his back towards **dem** headboard **so dass** he could lift his head better; **fand** where **das** itch was, and saw that it was **besetzt** with lots of little **weißen** spots which he didn’t know what to make of; and **als** he tried to feel the place with one of his legs he drew it quickly back because as soon as he touched it he was overcome by a cold shudder.

5.10 Conclusion

We presented a method to generate macaronic (mixed-language) documents to aid foreign language learners with vocabulary acquisition. Our key idea is to derive a model of student learning from only a cloze language model, which uses both context and spelling features. We find that our model-based teacher generates comprehensible macaronic text that promotes vocabulary learning. We find noticeable differences between the word replacement choices by the GSM (only uses context) and sGSM (uses spelling and context) models, especially in the English-Spanish case shown in §5.9. We find more L2 replaces for words that have a high overlap with their spelling in English. For example, **existencia**, **fortuna**, **matrimonio**, **propiedad**, **necesitaban**, **beneficio**, **articulacion**, **interés**, **importante**, **constante** and **residencia** were all replaced using the sGSM model. As futher confirmation, we find exact replacements were also selected by the sGSM model, such as **Dashwood**, **Park** and **general**. The GSM model replaced fewer tokens with high-overlap, **ocasionales**, **importante** and **existencia** can be seen in L2. We leave the task of extending it to phrasal translation and incorporating word reordering as future work. We also leave the exploration of alternate character-based compositions such as Kim et al. (2016) for future work. Beyond that, we envision machine teaching interfaces in which the student reader *interacts* with the macaronic text—advancing through the document, clicking on words for hints, and facing occasional quizzes (Renduchintala et al., 2016b)—and with other educational stimuli.

As we began to explore in Renduchintala et al. (2016a) and Renduchintala, Koehn, and Eisner (2017), interactions provide feedback that the machine teacher could use to adjust its model of the student's lexicons (here $\mathbf{E}, \mathbf{F}$), inference (here $\boldsymbol{\theta}^{\mathbf{f}}, \boldsymbol{\theta}^{\mathbf{b}}, \boldsymbol{\theta}^{\mathbf{h}}, \mu$), and learning (here λ). In this context, we are interested in using models that are *student-specific* (to reflect individual learning styles), *stochastic* (since the student's observed behavior may be inconsistent owing to distraction or fatigue), and able to model *forgetting* as well as learning (Settles and Meeder, 2016).

L2	τ	**Model**	**Closed-class**	**Open-class**
Es	0.0	random	$0.67 \pm 0.0037(60)$	$0.60 \pm 0.0027(46)$
		GSM	$0.67 \pm 0.0060(60)$	$0.62 \pm 0.0076(15)$
		sGSM	$0.71 \pm 0.0035(47)$	$0.68 \pm 0.0028(24)$
	0.2	random	$0.67 \pm 0.0037(60)$	$0.60 \pm 0.0027(46)$
		GSM	$0.67 \pm 0.0061(60)$	$0.61 \pm 0.0080(15)$
		sGSM	$0.71 \pm 0.0036(47)$	$0.67 \pm 0.0029(24)$
	0.4	random	$0.60 \pm 0.0051(60)$	$0.50 \pm 0.0037(46)$
		GSM	$0.60 \pm 0.0086(60)$	$0.51 \pm 0.0106(15)$
		sGSM	$0.66 \pm 0.0044(47)$	$0.61 \pm 0.0037(24)$
	0.6	random	$0.47 \pm 0.0058(60)$	$0.40 \pm 0.0041(46)$
		GSM	$0.48 \pm 0.0084(60)$	$0.42 \pm 0.0105(15)$
		sGSM	$0.52 \pm 0.0054(47)$	$0.50 \pm 0.0037(24)$
	0.8	random	$0.40 \pm 0.0053(60)$	$0.30 \pm 0.0032(46)$
		GSM	$0.41 \pm 0.0078(60)$	$0.37 \pm 0.0097(15)$
		sGSM	$0.46 \pm 0.0055(47)$	$0.41 \pm 0.0041(24)$
	1.0	random	$0.40 \pm 0.0053(60)$	$0.29 \pm 0.0031(46)$
		GSM	$0.40 \pm 0.0077(60)$	$0.36 \pm 0.0092(15)$
		sGSM	$0.45 \pm 0.0053(47)$	$0.39 \pm 0.0042(24)$

Table 5.15: An expanded version of Table 5.10 (human retention experiments), reporting results with various values of τ.

Chapter 6

Knowledge Tracing in Sequential Learning of Inflected Vocabulary

Our macaronic framework facilitates learning novel vocabulary and linguistic structures while a student is progressing through a document sequentially. In doing so, the student should (hopefully) acquire new knowledge but may also forget what they have previously learned. Furthermore, new evidence, in the form of a new macaronic sentence for example, might force the student to adjust their understanding of previously seen L2 words and structures.

In other words, the previous chapters were concerned with what a student can learn when presented with a macaronic sentence. In chapters Chapter 3 and Chapter 5 we make simplistic assumptions about what the student already knows and model what they gain from a new macaronic stimulus. In this chapter, we study *knowledge tracing* in the context of inflection learning task. We view the learning process as a sequence of smaller learning events and model the *interaction* between new knowledge (arriving via some new evidence, perhaps a macaronic sentence or, in this chapter, a flash card), existing knowledge which could be corrupted by forgetting or confusing similar vocabulary items etc.

Knowledge tracing attempts to reconstruct when a student acquired (or forgot) each of several facts. Yet we often hear that "learning is not just memorizing facts." Facts are not atomic objects to be discretely and independently manipulated. Rather, we suppose, a student who recalls a fact in a given setting is demonstrating a *skill*—by solving a structured prediction problem that is akin to reconstructive memory (Schacter, 1989; Posner, 1989) or pattern completion (Hopfield, 1982; Smolensky, 1986). The attempt at structured prediction may draw on many cooperating feature weights, some of which may be shared with other facts or skills.

In this chapter, we study models for knowledge tracing for the task of foreign-language vocabulary inflection learning, we will adopt a specific structured prediction model and learning algorithm. Different knowledge states correspond to model parameter settings (feature weights). Different learning styles correspond to different hyperparameters that govern the learning algorithm.[1] As we interact with each student through a simple online tutoring system, we would like to track their evolving knowledge state and identify their learning style. That is, we would like to discover parameters and hyperparameters that can explain the evidence so far and predict how the student will react in future. This could

[1]currently, we assume that all students share the same hyperparameters (same learning style), although each student will have their own parameters, which change as they learn.

help us make good future choices about how to instruct this student, although we leave this reinforcement learning problem to future work. We show that we can predict the student's next answer.

In short, we expand the notion of a knowledge tracing model to include representations for a student's (i) current knowledge, (ii) retention of knowledge, and (iii) acquisition of new knowledge. Our reconstruction of the student's knowledge state remains interpretable, since it corresponds to the weights of hand-designed features (sub-skills). Interpretability may help a future teaching system provide useful feedback to students and to human teachers, and help it construct educational stimuli that are targeted at improving particular sub-skills, such as features that select correct verb suffixes.

As mentioned, we consider a verb conjugation task instead of a macaronic learning task, where a foreign language learner learns the verb conjugation paradigm by reviewing and interacting with a series of flash cards. This task is a good testbed, as it needs the learner to deploy sub-word features and to generalize to new examples. For example, a student learning Spanish verb conjugation might encounter pairs such as (`tú entras`, `you enter`), (`yo miro`, `I watch`). Using these examples, the student needs to recognize suffix patterns and apply them to new pairs seen such as (`yo entro`, `I enter`). While we considered sub-word features even in out macaronic experiments, the verb inflection task is more focused on sub-word based generalizations that the student must understand in order to perform the task.

Vocabulary learning presents a challenging learning environment due to the large number of skills (words) that need to be traced. Learning vocabulary in conjunction with inflection further complicates the challenge due to the number of new sub-skills that are introduced. Huang, Guerra, and Brusilovsky (2016) suggest that modeling sub-skill interaction is crucial to several knowledge tracing domains. For our domain, a log-linear formulation elegantly allows for arbitrary sub-skills via feature functions.

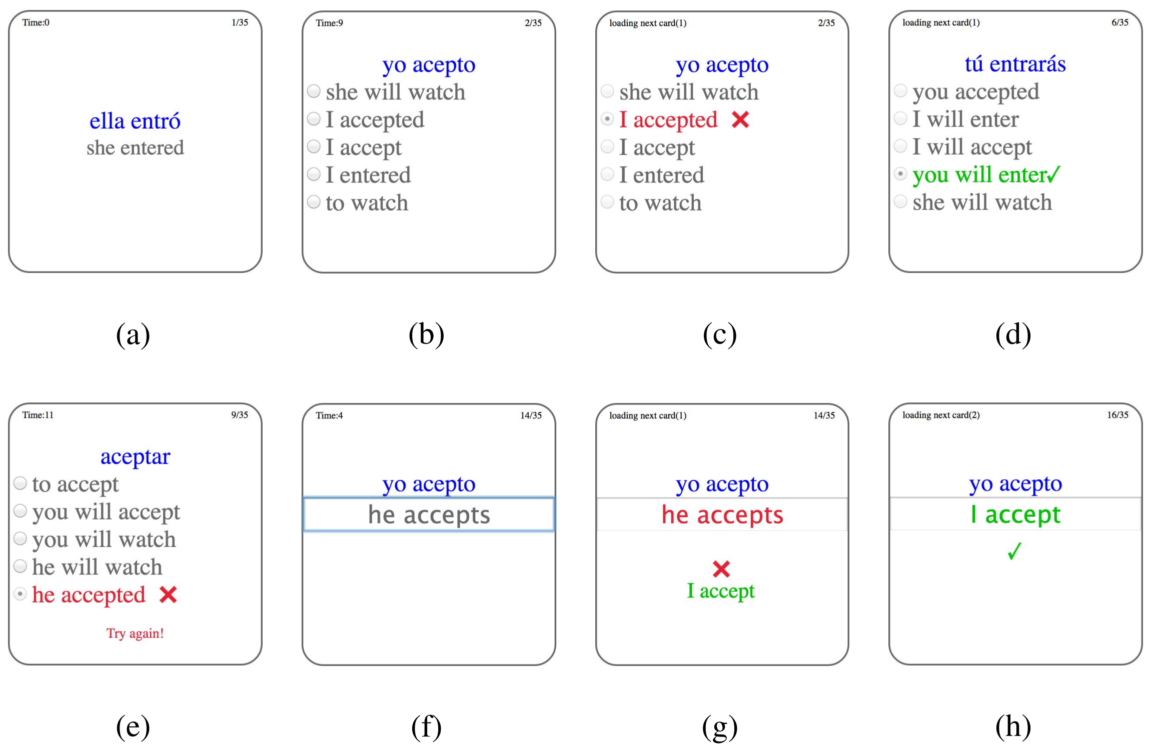
6.1 Related Work

Bayesian knowledge tracing (Corbett and Anderson, 1994) (BKT) has long been the standard method to infer a student's knowledge from his or her performance on a sequence of task items. In BKT, each skill is modeled by an HMM with two hidden states ("known" or "not-known"), and the probability of success on an item depends on the state of the skill it exercises. Transition and emission probabilities are learned from the performance data using Expectation Maximization (EM). Many extensions of BKT have been investigated, including personalization (e.g., Lee and Brunskill, 2012; Khajah et al., 2014a) and modeling item difficulty (Khajah et al., 2014b).

Our approach could be called *Parametric Knowledge Tracing (PKT)* because we take a student's knowledge to be a vector of prediction parameters (feature weights) rather than a vector of skill bits. Although several BKT variants (Koedinger et al., 2011; Xu and Mostow, 2012; González-Brenes, Huang, and Brusilovsky, 2014) have modeled the fact that related skills share sub-skills or features, that work does not associate a real-valued weight with each feature at each time. Either skills are still represented with separate HMMs, whose transition and/or emission probabilities are parameterized in terms of shared features with time-invariant weights; or else HMMs are associated with the individual sub-skills, and the performance of a skill depends on which of its subskills are in the "known" state.

Our current version is not Bayesian since it assumes deterministic updates (but see footnote 4). A closely related line of work with deterministic updates is deep knowledge tracing (DKT) (Piech et al., 2015), which applied a classical LSTM model (Hochreiter and Schmidhuber, 1997) to knowledge tracing and showed strong improvements over BKT. Our PKT model differs from DKT in that the student's state at each time step is a more interpretable feature vector, and the state update rule is also interpretable—it is a type of error-correcting learning rule. In addition, the student's state is able to predict the student's actual response and not merely whether the response was correct. We expect that having an interpretable feature vector has better inductive bias (see experiment in section 6.6.1), and that it may be useful to plan future actions by smart flash card systems. Moreover, in this work we test different plausible state update rules and see how they fit actual student responses, in orer to gain insight about learning.

Most recently, Settles and Meeder (2016)'s half-life regression assumes that a student's retention of a particular skill exponentially decays with time and learns a parameter that models the rate of decay ("half-life regression"). Like González-Brenes, Huang, and Brusilovsky (2014) and Settles and Meeder (2016), our model leverages a feature-rich formulation to predict the probability of a learner correctly remembering a skill, but can also capture complex spacing/retention patterns using a neural gating mechanism. Another distinction between our work and half-life regression is that we focus on knowledge tracing within a single session, while half-life regression collapses a session into a single data point and operates on many such data points over longer time spans.



(a)
Time:0 1/35
ella entró
she entered

(b)
Time:9 2/35
yo acepto
- she will watch
- I accepted
- I accept
- I entered
- to watch

(c)
loading next card(1) 2/35
yo acepto
- she will watch
- I accepted ✗
- I accept
- I entered
- to watch

(d)
loading next card(1) 6/35
tú entrarás
- you accepted
- I will enter
- I will accept
- you will enter✓
- she will watch

(e)
Time:11 9/35
aceptar
- to accept
- you will accept
- you will watch
- he will watch
- he accepted ✗

Try again!

(f)
Time:4 14/35
yo acepto
he accepts

(g)
loading next card(1) 14/35
yo acepto
he accepts
✗
I accept

(h)
loading next card(2) 16/35
yo acepto
I accept
✓

Figure 6.1: Screen grabs of card modalities during training. These examples show cards for a native English speaker learning Spanish verb conjugation. Fig 6.1a is an EX card, Fig 6.1b shows a MC card before the student has made a selection, and Fig 6.1c and 6.1d show MC cards after the student has made an incorrect or correct selection respectively, Fig 6.1e shows a MC card that is giving the student another attempt (the system randomly decides to give the student up to three additional attempts), Fig 6.1f shows a TP card where a student is completing an answer, Fig 6.1g shows a TP card that has marked a student answer wrong and then revealed the right answer (the reveal is decided randomly), and finally Fig 6.1h shows a card that is giving a student feedback for their answer.

6.2 Verb Conjugation Task

We devised a flash card training system to teach verb conjugations in a foreign language. In this study, we only asked the student to translate from the foreign language to English, not vice-versa.[2]

6.2.1 Task Setup

We consider a setting where students go through a series of interactive flash cards during a training session. Figure 6.1 shows the three types of cards: (i) *Example* (EX) cards simply display a foreign phrase and its English translation (for 7 seconds). (ii) *Multiple-Choice* (MC) cards show a single foreign phrase and require the student to select one of five possible English phrases shown as options. (iii) *Typing* (TP) cards show a foreign phrase and a text input box, requiring the student to type out what they think is the English translation. Our system can provide feedback for each student response. (i) *Indicative Feedback*: This refers to marking a student's answer as correct or incorrect (Fig. 6.1c, 6.1d and 6.1h). Indicative feedback is always shown for both MC and TP cards. (ii) *Explicit Feedback*: If the student makes an error on a TP card, the system has a 50% chance of showing them the true answer (Fig. 6.1g). (iii) *Retry*: If the student makes an error on a MC card, the system has a 50% chance of allowing them to try again, up to a maximum of 3 attempts.

[2]We would regard these as two separate skills that share parameters to some degree, an interesting subject for future study.

Categories	Inf	SPre,1,N	SPre,2,N	SPre,3,M	SPre,3,F	SF,1,N	SF,2,N	SF,3,M	SF,3,F	SP,1,N	SP,2,N	SP,3,M	SP,3,F
Lemma	acceptar	yo acepto	tú aceptas	él acepta	ella acepta	yo aceptaré	tú aceptarás	él aceptará	ella aceptará	yo acepté	tú aceptaste	él aceptó	ella aceptó
	to accept	I accept	you accept	he accepts	she accepts	I will accept	you will accept*	he will accept	she will accept	I accepted*	you accepted	he accepted	she accepted
	entrar	yo entro	tú entras	él entra	ella entra	yo entraré	tú entrarás	él entrará	ella entrará	yo entré	tú entraste	él entró	ella entró
	to enter	I enter	you enter	he enters	she enters	I will enter	you will enter	he will enter	she will enter	I entered	you entered	he entered	she entered
	mirar	yo miro	tú miras	él mira	ella mira	yo miraré	tú mirarás	él mirará	ella mirará	yo miré	tú miraste	él miró	ella miró
	to watch	I watch*	you watch*	he watches*	she watches	I will watch	you will watch*	he will watch	she will watch	I watched	you watched	he watched*	she watched

Table 6.1: Content used in training sequences. Phrase pairs with * were used for the quiz at the end of the training sequence. This Spanish content was then transformed using the method in section 6.5.1.

6.2.2 Task Content

In this particular task we used three verb lemmas, each inflected in 13 different ways (Table 6.1). The inflections included three tenses (simple past, present, and future) in each of four persons (first, second, third masculine, third feminine), as well as the infinitive form. We ensured that each surface realization was unique and regular, resulting in 39 possible phrases.[3] Seven phrases from this set were randomly selected for a quiz, which is shown at the end of the training session, leaving 32 phrases that a student may see in the training session. The student's responses on the quiz do not receive any feedback from the system.We also limited the training session to 35 cards (some of which may require multiple rounds of interaction, owing to retries). All of the methods presented in this paper could be applied to larger content sets as well.

[3]The inflected surface forms included explicit pronouns.

6.3 Notation

We will use the following conventions in this paper. System actions a_t, student responses y_t, and feedback items a'_t are subscripted by a time $1 \leq t \leq T$. Other subscripts pick out elements of vectors or matrices. Ordinary lowercase letters indicate scalars (α, β, etc.), boldfaced lowercase letters indicate vectors ($\boldsymbol{\theta}$, $\mathbf{y}$, $\mathbf{w}^{\mathrm{zx}}$), and boldfaced uppercase letters indicate matrices ($\boldsymbol{\Phi}$, $\mathbf{W}^{\mathrm{hh}}$, etc.). The roman-font superscripts are part of the vector or matrix name.

6.4 Student Models

6.4.1 Observable Student Behavior

A flash card is a structured object $a = (x, \mathcal{O})$, where $x \in \mathcal{X}$ is the foreign phrase and $\mathcal{O}$ is a set of allowed responses. For an MC card, $\mathcal{O}$ is the set of 5 multiple-choice options on that card (or fewer on a retry attempt). For a EX or TP card, $\mathcal{O}$ is the set of all 39 English phrases (the TP user interface prevents the student from submitting a guess outside this set).

For non-EX cards, we assume the student samples their response $y \in \mathcal{O}$ from a log-linear distribution parameterized by their knowledge state $\boldsymbol{\theta} \in \mathbb{R}^d$:

$$\begin{aligned} p(y \mid a;\ \boldsymbol{\theta}) &= p(y \mid x, \mathcal{O};\ \boldsymbol{\theta}) \\ &= \frac{\exp(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(x, y))}{\sum_{y' \in \mathcal{O}} \exp(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(x, y'))} \end{aligned} \tag{6.1}$$

where $\boldsymbol{\phi}(x, y) \in \{0, 1\}^d$ is a feature vector extracted from the (x, y) pair.

6.4.2 Feature Design

The student's knowledge state is described by the weights $\boldsymbol{\theta}$ placed on the features $\boldsymbol{\phi}(x, y)$ in equation (6.1). We assume the following binary features will suffice to describe the student's behavior.

- *Phrasal* features: We include a unique indicator feature for each possible (x, y) pair, yielding 39^2 features. For example, there exists a feature that fires iff $x = \texttt{yo\_miro} \wedge y = \texttt{I\_enter}$.
- *Word* features: We include indicator features for all (source word, target word) pairs: e.g., $\texttt{yo} \in x \wedge \texttt{enter} \in y$. (These words need not be aligned.)
- *Morpheme* features: We include indicator features for all (w, mc) pairs, where w is a word of the source phrase x, and m is a possible tense, person, or number for the target phrase y (drawn from Table 6.1). For example, m might be `1st` (first person) or `SPre` (simple present).
- *Prefix* and *suffix* features: For each word or morpheme feature that fires, 8 backoff features also fire, where the source word and (if present) the target word are replaced by their first or last i characters, for $i \in \{1, 2, 3, 4\}$.

These templates yield about 4600 features in all, so the knowledge state has $d \approx 4600$ dimensions.

6.4.3 Learning Models

We now turn to the question of modeling how the student's knowledge state changes during their session. $\boldsymbol{\theta}_t$ denotes the state at the start of round t. We take $\boldsymbol{\theta}_1 = \mathbf{0}$ and assume that the student uses a deterministic update rule of the following form:[4]

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\beta}_t \odot \boldsymbol{\theta}_t + \boldsymbol{\alpha}_t \odot \mathbf{u}_t \tag{6.2}$$

where $\mathbf{u}_t$ is an *update vector* that depends on the student's experience (a_t, y_t, a'_t) at round t. In general, we can regard $\boldsymbol{\alpha}_t \in (0,1)^d$ as modeling the rates at which the learner updates the various parameters according to $\mathbf{u}_t$, and $\boldsymbol{\beta}_t \in (0,1)^d$ as modeling the rates at which those parameters are forgotten. These vectors correspond respectively to the *input gates* and *forget gates* in recurrent neural network architectures such as the LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014). As in those architectures, we will use neural networks to choose $\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t$ at each time step t, so that they may be sensitive in nonlinear ways to the context at round t.

Why this form? First imagine that the student is learning by stochastic gradient descent on some L$_2$-regularized loss function $C\cdot \parallel \boldsymbol{\theta} \parallel^2 + \sum_t \mathcal{L}_t(\boldsymbol{\theta})$. This algorithm's update rule has the simplified form

$$\boldsymbol{\theta}_{t+1} = \beta_t \cdot \boldsymbol{\theta}_t + \alpha_t \cdot \mathbf{u}_t \tag{6.3}$$

[4]Since learning is not perfectly predictable, it would be more realistic to compute $\boldsymbol{\theta}_t$ by a stochastic update—or equivalently, by a deterministic update that also depends on a random noise vector $\boldsymbol{\epsilon}_t$ (which is drawn from, say, a Gaussian). These noise vectors are "nuisance parameters," but rather than integrating over their possible values, a straightforward approximation is to optimize them by gradient descent—along with the other update parameters—so as to locally maximize likelihood.

where $\mathbf{u}_t = -\nabla \mathcal{L}_t(\boldsymbol{\theta})$ is the steepest-descent direction on example t, $\alpha_t > 0$ is the learning rate at time t, and $\beta_t = 1 - \alpha_t C$ handles the weight decay due to following the gradient of the regularizer.

Adaptive versions of stochastic gradient descent—such as AdaGrad (Duchi, Hazan, and Singer, 2011) and AdaDelta (Zeiler, 2012)—are more like our full rule (6.2) in that they allow different learning rates for different parameters.

6.4.3.1 Schemes for the Update Vector $\mathbf{u}_t$

We assume that $\mathbf{u}_t$ is the gradient of some log-probability, so that the student learns by trying to increase the log-probability of the correct answer. However, the student does not always *observe* the correct answer y. For example, there is no output label provided when the student only receives feedback that their answer is incorrect. Even in such cases, the student can change their knowledge state.

In this section, we define schemes for defining $\mathbf{u}_t$ from the experience (a_t, y_t, a'_t) at round t. Recall that $a_t = (x_t, \mathcal{O}_t)$. We omit the t subscripts below.

Suppose the student is told that a particular phrase $y \in \mathcal{O}$ is the correct translation of x (via an EX card or via feedback on an answer to an MC or TP card). Then an apt strategy for the student would be to use the following gradient:[5]

$$\begin{aligned}\boldsymbol{\Delta}^{\checkmark} &= \nabla_{\boldsymbol{\theta}} \log p(y \mid x, \mathcal{O};\ \boldsymbol{\theta}) \qquad (6.4)\\ &= \boldsymbol{\phi}(x,y) - \sum_{y' \in \mathcal{O}} p(y' \mid x) \boldsymbol{\phi}(x,y')\end{aligned}$$

If the student is told that y is *incorrect*, an apt strategy is to move probability mass collectively to the other available options, increasing their total probability, since one of those options *must* be correct. We call this the *redistribution gradient (RG)*:

$$\begin{aligned}\boldsymbol{\Delta}^{\times} &= \nabla_{\boldsymbol{\theta}} \log p(\mathcal{O} - \{y\} \mid x, \mathcal{O};\ \boldsymbol{\theta}) \qquad (6.5)\\ &= \sum_{y' \in \mathcal{O} - \{y\}} p(y' \mid x, y' \neq y) \boldsymbol{\phi}(x,y') \qquad (6.6)\\ &\quad - \sum_{y' \in \mathcal{O}} p(y' \mid x) \boldsymbol{\phi}(x,y')\end{aligned}$$

where $p(y' \mid x, y' \neq y)$ is a renormalized distribution over just the options $y' \in \mathcal{O} - \{y\}$. Note that if the student selects two wrong answers y_1, y_2 in a row on an MC card, the first update will subtract the average features of $\mathcal{O}$ and add those of $\mathcal{O} - \{y_1\}$; the second update will subtract the average features of $\mathcal{O} - \{y_1\}$ and add those of $\mathcal{O} - \{y_1, y_2\}$. The intermediate addition and subtraction cancel out if the same $\boldsymbol{\alpha}$ vector is used at both rounds, so the net effect is to shift probability mass from the 5 initial options to the 3 remaining ones.[6]

An alternate scheme for incorrect y is to use $-\boldsymbol{\Delta}^{\checkmark}$. We call this *negative gradient (NG)*.

[5] An objection is that for an EX or TP card, the student may not actually know the exact set of options $\mathcal{O}$ in the denominator. We attempted setting $\mathcal{O}$ to be the set of English phrases the student has seen prior to the current question. Though intuitive, this setting performed worse on all the update and gating schemes.

[6] Arguably, a zeroth update should be allowed as well: upon first viewing the MC card, the student should have the chance to subtract the average features of the full set of possibilities and add those of the 5 options in $\mathcal{O}$, since again, the system is implying that one of those 5 options *must* be correct.

Update Scheme	Correct	Incorrect
redistribution (RG)	$\mathbf{u}_t = \boldsymbol{\Delta}^{✓}$	$\mathbf{u}_t = \boldsymbol{\Delta}^{✗}$
negative grad. (NG)	$\mathbf{u}_t = \boldsymbol{\Delta}^{✓}$	$\mathbf{u}_t = -\boldsymbol{\Delta}^{✓}$
feature vector (FG)	$\mathbf{u}_t = \boldsymbol{\phi}(x, y)$	$\mathbf{u}_t = -\boldsymbol{\phi}(x, y)$

Table 6.2: Summary of update schemes (other than RNG).

Since the RG and NG update vectors both worked well for handling incorrect y, we also tried linearly interpolating them *(RNG)*, with $\mathbf{u}_t = \boldsymbol{\gamma}_t \odot \boldsymbol{\Delta}^{✗} + (\mathbf{1} - \boldsymbol{\gamma}_t) \odot -\boldsymbol{\Delta}^{✓}$. The interpolation vector $\boldsymbol{\gamma}_t$ has elements in $(0, 1)$, and may depend on the context (possibly different for MC and EX cards, for example).

Finally, the *feature vector (FG)* scheme simply adds the features $\phi(x, y)$ when y is correct or subtracts them when y is incorrect. This is appropriate for a student who pays attention only to y, without bothering to note that the alternative options in $\mathcal{O}$ are (respectively) incorrect or correct.

Recall from section 6.2.1 that the system sometimes gives both indicative and explicit feedback, telling the student that one phrase is incorrect and a different phrase is correct. We treat these as two successive updates with update vectors $\mathbf{u}_t$ and $\mathbf{u}_{t+1}$. Notice that in the FG scheme, adding this pair of update vectors resembles a perceptron update. Table 6.2 summarizes our update schemes.

6.4.3.2 Schemes for the Gates $\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \boldsymbol{\gamma}_t$

We characterize each update t by a 7-dimensional context vector $\mathbf{c}_t$, which summarizes what the student has experienced. The first three elements in $\mathbf{c}_t$ are binary indicators of the type of flash card (EX, MC or TP). The next three elements are binary indicators of the type of information that caused the update: correct student answer, incorrect student answer, or revealed answer (via an EX card or explicit feedback). As a reminder, the system can respond with an indication that the answer is correct or incorrect, or it can reveal the answer. Finally, the last element of $\mathbf{c}_t$ is $1/|\mathcal{O}|$, the chance probability of success on this card. From $\mathbf{c}_t$, we define

$$\boldsymbol{\alpha}_t = \sigma(\mathbf{W}^{\alpha}\mathbf{c}_t \quad + b^{\alpha}\mathbf{1}) \qquad \in (0,1)^d \tag{6.7}$$

$$\boldsymbol{\beta}_t = \sigma(\mathbf{W}^{\beta}\mathbf{c}_{t-1} + b^{\beta}\mathbf{1}) \qquad \in (0,1)^d \tag{6.8}$$

$$\boldsymbol{\gamma}_t = \sigma(\mathbf{W}^{\gamma}\mathbf{c}_t \quad + b^{\gamma}\mathbf{1}) \qquad \in (0,1)^d \tag{6.9}$$

where $\mathbf{c}_0 = \mathbf{0}$. Each gate vector is now parameterized by a weight matrix $\mathbf{W} \in \mathbb{R}^{d\times 7}$, where d is the dimensionality of the gradient and knowledge state.

We also tried simpler versions of this model. In the *vector model (VM)*, we define $\boldsymbol{\alpha}_t = \sigma(\mathbf{b}^{\alpha})$, and $\boldsymbol{\beta}_t, \boldsymbol{\gamma}_t$ similarly. These vectors do not vary with time and simply reflect that some parameters are more labile than others. Finally, the *scalar model (SM)* defines $\boldsymbol{\alpha}_t = \sigma(b^{\alpha}\mathbf{1})$, so that all parameters are equally labile. One could also imagine tying the gates for features derived from the same template, meaning that some *kinds* of features (in some contexts) are more labile than others, or reducing the number of parameters by learning low-rank $\mathbf{W}$ matrices.

While we also tried augmenting the context vector $\mathbf{c}_t$ with the knowledge state $\boldsymbol{\theta}_t$, this resulted in far too many parameters to train well, and did not help performance in pilot tests.

6.4.4 Parameter Estimation

We tune the $\mathbf{W}$ and b parameters of the model by maximum likelihood, so as to better predict the students' responses y_t. The likelihood function is

$$\begin{aligned} p(y_1, \dots y_T \mid a_t, \dots a_T) &= \prod_{t=1}^{T} p(y_t \mid a_{1:t}, y_{1:t-1}, a'_{1:t-1}) \\ &= \prod_{t=1}^{T} p(y_t \mid a_t; \boldsymbol{\theta}_t) \end{aligned} \tag{6.10}$$

where we take $p(y_t \mid \cdots) = 1$ at steps where the student makes no response (EX cards and explicit feedback). Note that the model assumes that $\boldsymbol{\theta}_t$ is a sufficient statistic of the student's past experiences.

For each (update scheme, gating scheme) combination, we trained the parameters using SGD with RMSProp updates (Tieleman and Hinton, 2012) to maximize the regularized log-likelihood

$$\sum_{t, \tau_t = 0} \log\ p(y_t \mid x_t; \boldsymbol{\theta}_t) - C \cdot \parallel \mathbf{W} \parallel^2 \tag{6.11}$$

summed over all students. Note that $\boldsymbol{\theta}_t$ depends on the parameters through the gated update rule (6.2). The development set was used for early stopping and to tune the regularization parameter C.[7]

[7]We searched $C \in \{0.00025, 0.0005, 0.001, \dots, 0.01, 0.025, 0.05, 0.1\}$ for each gating model and update

6.5 Data Collection

We recruited 153 unique "students" via Amazon Mechanical Turk (MTurk). MTurk participants were compensated \$1 for completing the training and test sessions and a bonus of \$10 was given to the three top scoring students. In our dataset, we retained only the 121 students who answered all questions.

6.5.1 Language Obfuscation

Fig. 6.1 shows a few example flash cards for a native English speaker learning Spanish. Fig. 6.1 shows all our Spanish-English phrase pairs. In our actual task, however, we invented an artificial language for the MTurk students to learn, which allowed us to ignore the problem of students with different initial knowledge levels. We generated our artificial language by enciphering the Spanish orthographic representations. We created a mapping from the true source string alphabet to an alternative, manually defined alphabet, while attempting to preserve pronounceability (by mapping vowels to vowels, etc.). For example, `mirar` was transformed into `melil` and `tú aceptas` became `pi icedpiz`.

scheme combination. $C = 0.0025$ gave best results for the CM models, 0.01 for VM and 0.0005 for SM.

6.5.2 Card Ordering Policy

In the future, we expect to use planning or reinforcement learning to choose the sequence of stimuli for the student. For the present study of student behavior, however, we hand-designed a simple stochastic policy for choosing the stimuli.

The policy must decide what foreign phrase and card modality to use at each training step. Our policy likes to repeat phrases with which participants had trouble—in hopes that these already-taught phrases are on the verge of being learned. It also likes to pick out new phrases. This was inspired by the popular Leitner (1972) approach, which devised a system of buckets that control how frequently an item is reviewed by a student. Leitner proposed buckets with review frequency rates of every day, every 2 days, every 4 days and so on.

For each foreign phrase $x \in \mathcal{X}$, we maintain a *novelty score* v_x, which is a function of the number of times the phrase is exposed to a student and an *error score* e_x, which is a function of the number of times the student incorrectly responded to the phrase. These scores are initialized to 1 and updated as follows:[8]

$$\begin{aligned} v_x &\leftarrow v_x - 1 \text{ when } x \text{ is viewed} \\ e_x &\leftarrow \begin{cases} 2e_x \text{ when student gets } x \text{ wrong} \\ 0.5e_x \text{ when student gets } x \text{ right} \end{cases} \\ x &\sim \frac{g(\mathbf{v}) + g(\mathbf{e})}{2} \end{aligned} \tag{6.12}$$

[8]Arguably we should have updated e_x instead by adding/subtracting 1, since it will be exponentiated later.

On each round, we sample a phrase x from either P_v or P_e (equal probability); these distributions are computed by applying a softmax $g(.)$ over the vectors $\mathbf{v}$ and $\mathbf{e}$ respectively (see Eq. 6.12). Once the phrase x is decided, the modality (EX, MC, TP) is chosen stochastically using probabilities $(0.2, 0.4, 0.4)$, except that probabilities $(1, 0, 0)$ are used for the first example of the session, and $(0.4, 0.6, 0)$ if x is not "TP-qualified." A phrase is TP-qualified if the student has seen both x's pronoun and x's verb lemma on previous cards (even if their correct translation was not revealed). For an MC card, the distractor phrases are sampled uniformly without replacement from the 38 other phrases.

6.6 Results & Experiments

We partitioned the students into three groups: 80 students for training, 20 for development, and 21 for testing. Most students found the task difficult; the average score on the 7-question quiz—was 2.81 correct, with maximum score of 6. (Recall from section 6.2.2 that the quiz questions were typing questions, not multiple choice questions.) The histogram of user performance is shown in Fig. 6.2.

After constructing each model, we evaluated it on the held-out data: the 728 responses from the 21 testing students. We measure the log-probability under the model of each actual response ("cross-entropy"), and also the fraction of responses that were correctly predicted if our prediction was the model's max-probability response ("accuracy").

Table 6.3 shows the results of our experiment. All of our models were predictive, doing

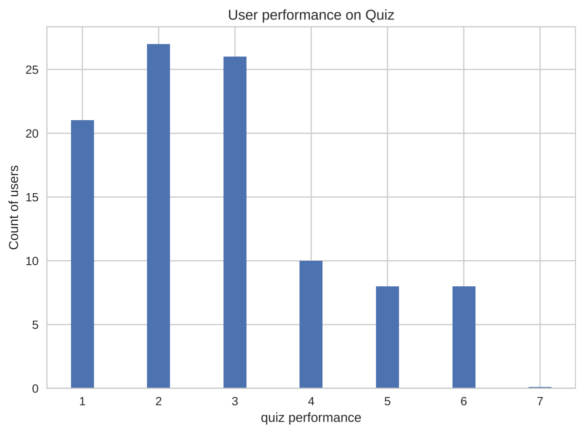


Figure 6.2: Quiz performance distribution (after removing users who scored 0).

far better than a uniform baseline that assigned equal probability $1/|\mathcal{O}|$ to all options. Our best models are shown in the final two lines, RNG+VM and RNG+CM.

Which update scheme was best? Interestingly, although the RG update vector is principled from a *machine* learning viewpoint, the NG update vector sometimes achieved better accuracy—though worse perplexity—when predicting the responses of *human* learners.[9] We got our best results on both metrics by interpolating between RG and NG (the RNG scheme). Recall that the NG scheme was motivated by the notion that students who guessed wrong may not study the alternative answers (even though one is correct), either because it is too much trouble to study them or because (for a TP card) those alternatives are not actually shown.

Which gating mechanism was best? In almost all cases, we found that more parameters helped, with CM $>$ VM $>$ SM on accuracy, and a similar pattern on cross-entropy (with VM sometimes winning but only slightly). In short, it helps to use different learning rates

[9]Even the FG vector sometimes won (on both metrics!), but this happened only with the worst gating mechanism, SM.

for different features, and it probably helps to make them sensitive to the learning context.

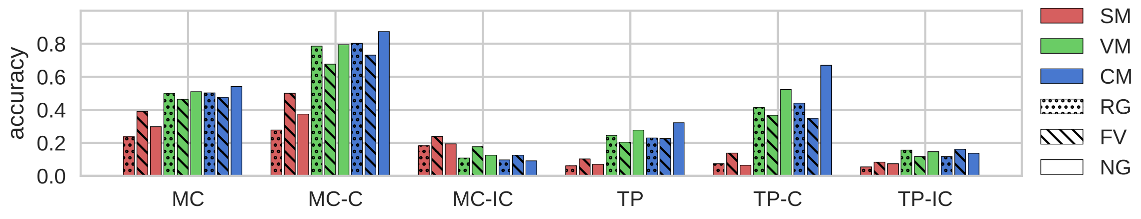


Figure 6.3: Plot comparing the models on test data under different conditions. Conditions MC and TP indicate Multiple-choice and Typing questions respectively. These are broken down to the cases where the student answers them correctly C and incorrectly IC. SM, VM, and CM represent scalar, vector, and context retention and acquisition gates (shown with different colors), respectively, while RG, NG and FG are redistribution, negative and feature vector update schemes(shown with different hatching patterns).

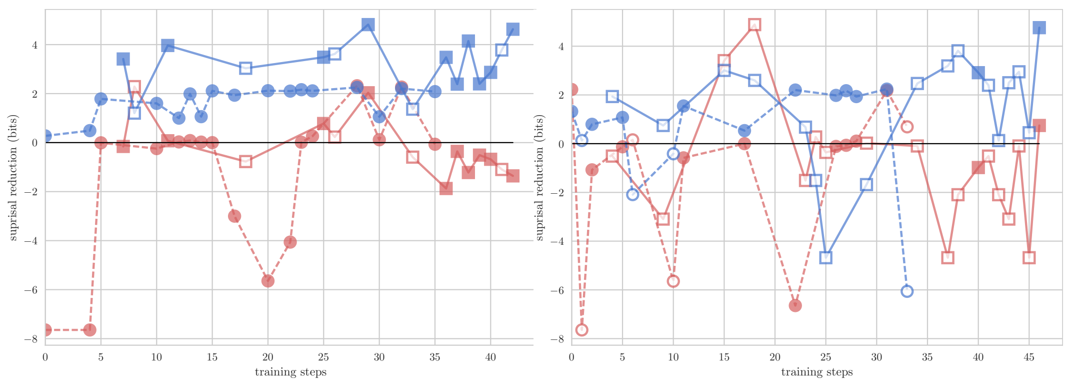
Surprisingly, the simple FG scheme outperformed both RG and NG when used in conjunction with a scalar retention and acquisition gate. This, however, did not extend to more complex gates.

Update Scheme	Gating Mechanism	accuracy	cross-ent.
(Uniform baseline)		0.133	2.459
FG	SM	0.239^{*}	2.362
FG	VM	$0.357^{\dagger}$	2.130
FG	CM	0.401	2.025
RG	SM	0.135	3.194
RG	VM	$0.397^{\dagger}$	1.909
RG	CM	0.405	1.938
NG	SM	0.185^{*}	4.674
NG	VM	$0.394^{\dagger}$	2.320
NG	CM	$0.449^{\dagger *}$	2.244
RNG (mixed)	SM	0.183	3.502
RNG (mixed)	VM	0.427	1.855
RNG (mixed)	CM	0.449	1.888

Table 6.3: Table summarizing prediction accuracy and cross-entropy (in nats per prediction) for different models. Larger accuracies and smaller cross-entropies are better. Within an update scheme, the † indicates significant improvement (McNemar's test, $p < 0.05$) over the next-best gating mechanism. Within g a gating mechanism, the * indicates significant improvement over the next-best update scheme. For example, NG+CM is significantly better than NG+VM, so it receives a †; it is also significantly better than RG+CM, and receives a * as well. These comparisons are conducted only among the pure update schemes (above the double line). All other models are significantly better than RG+SM ($p < 0.01$).

Fig. 6.3 shows a breakdown of the prediction accuracy measures according to whether the card was MC or TP, and according to whether the student's answer was correct (C) or incorrect (IC). Unsurprisingly, all the models have an easier time predicting the student's guess when the student is correct, since the predicted parameters $\boldsymbol{\theta}_t$ will often pick the correct answer. However, this is where the vector and context gates far outperform the scalar gates. All the models find predicting the incorrect answers of the students difficult. Moreover, when predicting these incorrect answers, the RG models do slightly better than the NG models.

The models obviously have higher accuracy when predicting student answers for MC cards than for TP cards, as MC cards have fewer options. Again, within both of these modalities, the vector and context gates outperform the scalar gate.



(a) a student with quiz score 6/7 (b) a student with quiz score 2/7

Figure 6.4: Predicting a specific student's responses. For each response, the plot shows our model's improvement in log-probability over the uniform baseline model. TP cards are the square markers connected by solid lines (the final 7 squares are the quiz), while MC cards—which have a much higher baseline—are the circle markers connected by dashed lines. Hollow and solid markers indicate correct and incorrect answers respectively. The RNG+CM model is shown in blue and the FG+SM model in red.

Finally, Fig. 6.4 examines how these models behave when making specific predictions over a training sequence for a single student. At each step we plot the difference in log-probability between our model and a uniform baseline model. Thus, a marker above 0 means that our model assigned the student's answer a probability higher than chance.[10] To contrast the performance difference, we show both the highest-accuracy model (RNG+CM) and the lowest-accuracy model (RG+SM). For a high-scoring student (Fig. 6.4a), we see RNG+CM

[10]For MC cards, the chance probability is in $\{\frac{1}{5}, \frac{1}{4}, \frac{1}{3}\}$—depending on how many options remain—while for TP cards it is $\frac{1}{39}$.

has a large margin over RG+SM and a slight upward trend. A higher probability than chance is noticeable even when the student makes mistakes (indicated by hollow markers). In contrast, for an average student (Fig. 6.4b), the margin between the two models is less perceptible. While the CM+NG model is still above the SM+RG line, there are some answers where CM+NG does very poorly. This is especially true for some of the wrong answers, for example at training steps 25, 29 and 33. Upon closer inspection into the model's error in step 33, we found the prompt received at this training step was `ekki melü` as a MC card, which had been shown to the student on three prior occasions, and the student even answered correctly on one of these occasions. This explains why the model was surprised to see the student make this error.

6.6.1 Comparison with Less Restrictive Model

Our parametric knowledge tracing architecture models the student as a typical structured prediction system, which maintains weights for hand-designed features and updates them roughly as an online learning algorithm would. A natural question is whether this restricted architecture sacrifices performance for interpretability, or improves performance via useful inductive bias.

To consider the other end of the spectrum, we implemented a flexible LSTM model in the style of recent deep learning research. This alternative model predicts each response by a student (i.e., on an MC or TP card) given the entire history of previous interactions with that student as summarized by an LSTM. The LSTM architecture is formally capable of capturing update rules exactly like those of PKT, but it is far from limited to such rules.

Much like equation (6.1), at each time t we predict

$$p(y_t = y \mid a_t) = \frac{\exp(\mathbf{h}_t \cdot \boldsymbol{\psi}(y))}{\sum_{y' \in \mathcal{O}_t} \exp(\mathbf{h}_t \cdot \psi(y))} \tag{6.13}$$

for each possible response y in the set of options $\mathcal{O}_t$, where $\boldsymbol{\psi}(y) \in \mathbb{R}^d$ is a learned embedding of response y. Here $\mathbf{h}_t \in \mathbb{R}^d$ denotes the hidden state of the LSTM, which evolves as the student interacts with the system and learns. $\mathbf{h}_t$ depends on the LSTM inputs for all times $< t$, just like the knowledge state $\boldsymbol{\theta}_t$ in equations (6.1)–(6.2). It also depends on the LSTM input for time t, since that specifies the flash card a_t to which we are predicting the response y_t.

Each flash card $a = (x, \mathcal{O})$ is encoded by a concatenation $\mathbf{a}$ of three vectors: a one-hot 39-dimensional vector specifying the foreign phrase x, a 39-dimensional binary vector $\boldsymbol{\mathcal{O}}$ indicating the possible English options in $\mathcal{O}$, and a one-hot vector indicating whether the card is EX, MC, or TP.

When reading the history of past interactions, the LSTM input at each time step t concatenates the vector representation $\mathbf{a}_t$ of the current flash card with vectors $\mathbf{a}_{t-1}, \mathbf{y}_{t-1}, \mathbf{f}_{t-1}$ that describe the student's experience in round $t-1$: these respectively encode the previous flash card, the student's response to it (a one-hot 39-dimensional vector), and the resulting feedback (a 39-dimensional binary vector that indicates the remaining options after feedback). Thus, if the student receives no feedback, then $\mathbf{f}_{t-1} = \boldsymbol{\mathcal{O}}_{t-1}$. Indicative feedback sets $\mathbf{f}_{t-1} = \mathbf{y}_{t-1}$ or $\mathbf{f}_{t-1} = \boldsymbol{\mathcal{O}}_{t-1} - \mathbf{y}_t$, according to whether the student was correct or incorrect. Explicit feedback (including for an EX card) sets $\mathbf{f}_{t-1}$ to a one-hot representation of the

Model	Parameters	Accuracy(test)	Cross-Entropy
RNG+CM	$\approx$ 97K	0.449	1.888
LSTM	$\approx$ 25K	0.429	1.992

Table 6.4: Comparison of our best-performing PKT model (RNG+CM) to our LSTM model. On our dataset, PKT outperforms the LSTM both in terms of accuracy and cross-entropy.

correct answer. Thus, $\mathbf{f}_{t-1}$ gives the set of "positive" options that we used in the RG update vector, while $\mathcal{O}_{t-1}$ gives the set of "negative" options, allowing the LSTM to similarly update its hidden state from $\mathbf{h}_{t-1}$ to $\mathbf{h}_t$ to reflect learning.[11]

As in section 6.4.4, we train the parameters by L_2-regularized maximum likelihood, with early stopping on development data. The weights for the LSTM were initialized uniformly at random $\sim U(-\delta, +\delta)$, where $\delta = 0.01$, and RMSProp was used for gradient descent. We settled on a regularization coefficient of 0.002 after a line search. The number of hidden units d was also tuned using line search. Interestingly, a dimensionality of just $d = 10$ performed best on dev data:[12] at this size, the LSTM has *fewer* parameters than our best model.

The result is shown in Table 6.4. These results favor our restricted PKT architecture. We

[11]This architecture is formally able to mimic PKT. We would store $\boldsymbol{\theta}$ in the LSTM's vector of cell activations, and configure the LSTM's "input" and "forget" gates to update this according to (6.2) where $\mathbf{u}_t$ is computed from the input. Observe that each feature in section 6.4.2 has the form $\phi_{ij}(x, y) = \xi_i(x) \cdot \psi_j(y)$. Consider the hidden unit in $\mathbf{h}$ corresponding to this feature, with activation θ_{ij}. By configuring this unit's "output" gate to be $\xi_i(x)$ (where x is the current foreign phrase given in the input), we would arrange for this hidden unit to have output $\xi_i(x) \cdot \theta_{ij}$, which will be multiplied by $\psi_j(y)$ in (6.13) to recover $\theta_{ij} \cdot \phi_{ij}(x, y)$ just as in (6.1). (More precisely, the output would be sigmoid($\xi_i(x) \cdot \theta_{ij}$), but we can evade this nonlinearity if we take the cell activations to be a scaled-down version of $\boldsymbol{\theta}$ and scale up the embeddings $\boldsymbol{\psi}(y)$ to compensate.)

[12]We searched $0.001, 0.002, 0.005, 0.01, 0.02, 0.05$ for the regularization coefficient, and $5, 10, 15, 20, 50, 100, 200$ for the number of hidden units.

acknowledge that the LSTM might perform better when a larger training set was available (which would allow a larger hidden layer), or using a different form of regularization (Srivastava et al., 2014).

Intermediate or hybrid models would of course also be possible. For example, we could predict $p(y \mid a_t)$ via (6.1), defining $\boldsymbol{\theta}_t$ as $\mathbf{h}_t^\top M$, a learned linear function of h_t. This variant would again have access to our hand-designed features $\boldsymbol{\phi}(x, y)$, so that it would know which flash cards were similar. In fact $\boldsymbol{\theta}_t \cdot \boldsymbol{\phi}(x, y)$ in (6.1) equals $\mathbf{h}_t \cdot (M\boldsymbol{\phi}(x, y))$, so M can be regarded as projecting $\boldsymbol{\phi}(x, y)$ down to the LSTM's hidden dimension d, learning how to weight and use these features. In this variant, the LSTM would no longer need to take a_t as part of its input at time t: rather, $\mathbf{h}_t$ (just like $\boldsymbol{\theta}_t$ in PKT) would be a pure representation of the student's knowledge state at time t, capable of predicting y_t for *any* a_t. This setup more closely resembles PKT—or the DKT LSTM of Piech et al. (2015). Unlike the DKT paper, however, it would still predict the student's specific response, not merely whether they were right or wrong.

6.7 Conclusion

We have presented a cognitively plausible model that traces a human student's knowledge as he or she interacts with a simple online tutoring system. The student must learn to translate very short inflected phrases from an unfamiliar language into English. Our model assumes that when a student recalls or guesses the translation, he or she is attempting to solve a structured prediction problem of choosing the best translation, based on salient features of the input-output pair. Specifically, we characterize the student's knowledge as a vector of feature weights, which is updated as the student interacts with the system. While the phrasal features memorize the translations of entire input phrases, the other features can pick up on the translations of individual words and sub-words, which are reusable across phrases.

We collected and modeled human-subjects data. We experimented with models using several different update mechanisms, focusing on the student's treatment of negative feedback and the degree to which the student tends to update or forget specific weights in particular contexts. We also found that in comparison to a less constrained LSTM model, we can better fit the human behavior by using weight update schemes that are broadly consistent with schemes used in machine learning.

In the future, we plan to experiment with more variants of the model, including variants that allow noise and personalization. Most important, we mean to use the model for planning which flash cards, feedback, or other stimuli to show next to a given student.

Chapter 7

Conclusion & Future Direction

This thesis introduces the problem of generating macaronic language texts as a foreign language learning paradigm. Adult foreign language learning is a challenging task that requires dedicated time and effort in following a curriculum. We believe the macaronic framework introduced in this thesis allows a student in engage in language learning while simply reading any document. We hope that such instruction will be a valuable addition to the traditional foreign language learning process.

We have made progress towards identifying appropriate data structures to all possible macaronic configurations for a given sentence, devised a method to model the readability and guessability of foreign language words and phrases in macaronic configurations and show how a simple search heuristic can find pedagogically useful macaronic configurations. We have also presented an interaction mechanism for macaronic documents, hopefully leading to improved student engagement while gaining the ability to update the student model based on feedback. Finally, we also studied sequential modeling of student's knowledge as they navigate through a restricted foreign language inflection learning activity.

There are several possible research directions moving forward. We are most interested in improving methods that follow the generic student model based approach as it allows us to create macaronic documents from a wide variety of domains without data collection involving human students. We identify the following limitations currently in the generic student model based approach:

Capturing uncertainty of L2 word embeddings: Word embeddings are points in a subspace. Assigning each L2 word with a single point in that subspace ignores uncertainty associated with that words meaning. This issue might not be very crucial when learning L1 embeddings, as we can assume (at least for frequent words) that we can learn their embeddings from different instances in the training data. However, our incremental L2 learning approach assigns/learns an embedding from (initially) just one exposure. Even subsequent exposures are not batched. Thus, it should be useful to maintain a range (or distribution) of reasonable embeddings for a new L2 word after each exposure, instead of a single point in the embedding space.

A possible approach could be to represent each L2 embedding by a multidimensional Gaussian with a mean vector $\mu \in \mathbb{R}^D$ and variance $\sigma^2 \in \mathbb{R}^D$. Similar ideas have been shown to help word embedding learning (Vilnis and McCallum, 2014; Athiwaratkun and Wilson, 2017) using "word2vec" style objectives (CBOW or skip-gram) (Mikolov et al., 2013). We could also employ the recent reparameterization method (Kingma and Welling, 2013).

Search Heuristics and Planning: We currently employ a simple ”left to right” best-first search heuristic to search for the “best” macaronic configuration for a given sentence. We can explore several alternative search heuristics. One simple alternative is to replace the search ordering from “left to right” to “low-frequency to high-frequency”. That is, we will try to replace low-frequency words in the sentence with their L2 translations before trying high-frequency words. This heuristic should provide more opportunities to replace low-frequency content words at the expense of high-frequency stop words, however, since high-frequency words are more likely to show up in the rest of the document there will be other opportunities for the model to replace these with L2 translations. Pilot experiments with low-frequency to high-frequency (in conjunction with best-first search) outperforms the left to right heuristic in terms of cosine similarity score as defined in §5.5.2.

Our current scheme does not consider the entire document when searching for the best macaronic configuration for a sentence. If the machine teacher knows, for example, that a certain L2 vocabulary item is more guessable in some future part of the document, then it could use the current sentence to teach a different L2 vocabulary item to the student. Thus, looking into the future of the document is a possible future direction of research. Our pilot experiments, using Monte-Carlo tree search to find the best macaronic configuration, also suggest the same. However, with longer look-ahead horizons search takes more time to complete which might hinder “online” search.

Contextual Representations from BERT and KL-Net: The cloze language model used in our generic student model is closely related to sentence representation models such as BERT and ELMo (Devlin et al., 2019; Peters et al., 2018). We could use these pretrained models instead or our cloze language model, however, we would be restricted by the L1 vocabulary used by these large masked language models.

Modified softmax layer in cloze model: Our incremental approach to learning the embeddings of novel L2 words is scored using cosine similarity (§5.5.1). However, the initial cloze model (§§ 5.2.1 and 5.2.2) does not take this particular scoring into account. When training the cloze model on L1 data and during incremental L2 word learning, the norms of the embeddings are not constrained, leading to common words having larger norms. This creates a mismatch between how we learn L1 embeddings and L2 embeddings (incrementally) and how we score them. While it is unclear if this dramatically changes the resulting macaronic configurations, a possible solution could be to use cosine similarity to obtain logits during L1 learning and during incremental L2 learning. This would encourage the initial cloze model to restrict the norms of word embeddings to be close to one.

Phrasal Translations: Finally, the one-to-one lexical translation setup is a limitation as it only affords for teaching single L2 words and not phrases. Additionally, does it expose the student to word order differences between the L1 and L2. There are two main challenges when moving to non-lexical teaching. To address this limitation we would first have to consider different scoring functions to guide the macaronic configuration search. Currently, the scoring function (§5.5.1) straight-forward for lexical translation case but not for scoring L1-L2 phrase pairs especially when also considering word-order differences between an L1 and L2 phrase. Consequently, we need to address how we represent L2 "knowledge"

Markup	He gave a talk about how education $\overset{de}{\underline{\text{and school kills creativity.}}}$
Prediction	He gave a talk about how education **und schulen kreativität tötet .**
Markup	$\overset{de}{\underline{\text{It was sombody who was trying}}}$ to ask a question about Javascript.
Prediction	**Es war jemand , der versuchte ,** to ask a question about Javascript .
Markup	$\overset{de}{\underline{\text{We were standing on the edge}}}$ of thousands of acres of $\overset{de}{\underline{\text{cotton}}}$.
Prediction	**Wir standen am rande** of thousands of acres of **baumwolle** .
Markup	And we're building upon innovations of $\overset{de}{\underline{\text{generations who went before us.}}}$
Prediction	And we're building upon innovations of **generationen , die vor uns gingen .**

Table 7.1: Examples of inputs and predicted outputs by our experimental NMT model trained to generate macaronic language sentences using annotations on the input sequence. We see that the macaronic language translations are able to correctly order German portions of the sentences, especially at the sentence ending. The source-features have also been learned by the NMT model and translations are faithful to the markup. The case, tokenization and italics added in post.

in our generic student model. Merely, using L2 word embeddings would be insufficient. One possible research endeavor is to not only learn new L2 word embeddings but also learn L2 recurrent parameters in an incremental fashion. That is, we can learn a entirely new L2 cloze language model. To score this cloze model we could use held out fully L2 sentences, perhaps from the remainder of the current document. Apart from redesigning the generic student model and incremental learning paradigm to enable phrasal L2 learning, we also have to alter how we generate the macaronic data structure that can support phrasal macaronic configurations. Creating the back-end macaronic data structure using statistical machine translation §1.4 may result in translations that are not accurate. Neural Machine Translation may provide better partial translations which could be used to generate the required back-end data structure. We find that we are able to generate fluent macaronic translations by tagging tokens in the input sequence (which is fully in L1) with either a `Translate` or `No-Translate` tag. Table 7.1 shows examples of generated En-De (L1-L2) macaronic sentences.

Longitudinal User Modelling: In this thesis, experiments involving human students were conducted on relatively short time-frames. Modelling long term learning and forgetting patterns in a macaronic learning setup would lead to better configurations as the machine teacher can account for student's forgetting patterns. Such experiments, however, would exhibit high variation and would require larger number of participants. Generally longer studies also exhibit poor participant retention.

Bibliography

Ahn, Luis von (2013). “Duolingo: Learn a Language for Free While Helping to Translate the Web”. In: *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, pp. 1–2.

Alishahi, Afra, Afsaneh Fazly, and Suzanne Stevenson (2008). “Fast mapping in word learning: What probabilities tell us”. In: *Proceedings of the twelfth conference on computational natural language learning*. Association for Computational Linguistics, pp. 57–64.

Athiwaratkun, Ben and Andrew Wilson (2017). “Multimodal Word Distributions”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1645–1656.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov (2017). “Enriching Word Vectors with Subword Information”. In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146. URL: `https://www.aclweb.org/anthology/Q17-1010`.

Bojar, Ondřej, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi (2015). “Findings of the 2015 Workshop on Statistical Machine Translation”. In: *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pp. 1–46. URL: `http://aclweb.org/anthology/W15-3001`.

Burstein, Jill, Joel Tetreault, and Nitin Madnani (2013). “The E-Rater Automated Essay Scoring System”. In: *Handbook of Automated Essay Evaluation: Current Applications and New Directions*. Ed. by Mark D. Shermis. Routledge, pp. 55–67.

Carey, Susan and Elsa Bartlett (1978). “Acquiring a single new word.” In:

Chen, Tao, Naijia Zheng, Yue Zhao, Muthu Kumar Chandrasekaran, and Min-Yen Kan (July 2015). “Interactive Second Language Learning from News Websites”. In: *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*. Beijing, China: Association for Computational Linguistics, pp. 34–42. URL: `https://www.aclweb.org/anthology/W15-4406`.

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (Oct. 2014). “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734. URL: `http://www.aclweb.org/anthology/D14-1179`.

Church, Kenneth Ward and Patrick Hanks (Mar. 1990). "Word Association Norms, Mutual Information, and Lexicography". In: *Computational Linguistics* 16.1, pp. 22–29. URL: http://dl.acm.org/citation.cfm?id=89086.89095.

Clarke, Linda K. (Oct. 1988). "Invented versus Traditional Spelling in First Graders' Writings: Effects on Learning to Spell and Read". In: *Research in the Teaching of English*, pp. 281–309. URL: http://www.jstor.org.proxy1.library.jhu.edu/stable/40171140.

Constantino, Rebecca, Sy-Ying Lee, Kyung-Sook Cho, and Stephen Krashen (1997). "Free Voluntary Reading as a Predictor of TOEFL Scores." In: *Applied Language Learning* 8.1, pp. 111–18.

Corbett, Albert T and John R Anderson (1994). "Knowledge tracing: Modeling the acquisition of procedural knowledge". In: *User modeling and user-adapted interaction* 4.4, pp. 253–278.

Daumé III, Hal (Aug. 2004). "Notes on CG and LM-BFGS Optimization of Logistic Regression". URL: http://hal3.name/megam/.

Daumé III, Hal (2007). "Frustratingly Easy Domain Adaptation". In: *Proceedings of ACL*. Prague, Czech Republic.

Daumé III, Hal (June 2007). "Frustratingly Easy Domain Adaptation". In: *Proceedings of ACL*, pp. 256–263. URL: http://www.aclweb.org/anthology/P07-1033.

Deutschlandfunk (2016). *nachrichtenleicht*. http://www.nachrichtenleicht.de/. Accessed: 2015-09-30. URL: www.nachrichtenleicht.de.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (June 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. URL: https://www.aclweb.org/anthology/N19-1423.

Dorr, Bonnie J. (Dec. 1994). "Machine Translation Divergences: A Formal Description and Proposed Solution". In: *Computational Linguistics* 20.4, pp. 597–633. URL: http://aclweb.org/anthology/J/J94/J94-4004.pdf.

Dreyer, Markus and Jason Eisner (Aug. 2009). "Graphical Models Over Multiple Strings". In: *Proceedings of EMNLP*. Singapore, pp. 101–110. URL: http://cs.jhu.edu/~jason/papers/\#dreyer-eisner-2009.

Du, Xinya, Junru Shao, and Claire Cardie (July 2017). "Learning to Ask: Neural Question Generation for Reading Comprehension". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 1342–1352. URL: https://www.aclweb.org/anthology/P17-1123.

Duchi, John, Elad Hazan, and Yoram Singer (2011). “Adaptive subgradient methods for online learning and stochastic optimization”. In: *Journal of Machine Learning Research* 12.Jul, pp. 2121–2159.

Dupuy, B and J McQuillan (1997). “Handcrafted books: Two for the price of one”. In: *Successful strategies for extensive reading*, pp. 171–180.

Elley, Warwick B and Francis Mangubhai (1983). “The impact of reading on second language learning”. In: *Reading research quarterly*, pp. 53–67.

Gentry, J. Richard (Nov. 2000). “A Retrospective on Invented Spelling and a Look Forward”. In: *The Reading Teacher* 54.3, pp. 318–332. URL: `http://www.jstor.org.proxy1.library.jhu.edu/stable/20204910`.

González-Brenes, José, Yun Huang, and Peter Brusilovsky (2014). “General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge”. In: *Proceedings of the 7th International Conference on Educational Data Mining*. University of Pittsburgh, pp. 84–91.

Grammarly (2009). *Grammarly*. `https://app.grammarly.com`. Accessed: 2019-02-20.

Heafield, Kenneth, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn (Aug. 2013). “Scalable Modified Kneser-Ney Language Model Estimation”. In: *Proceedings of ACL*. Sofia, Bulgaria, pp. 690–696. URL: `http://kheafield.com/professional/edinburgh/estimate\_paper.pdf`.

Heilman, Michael and Nitin Madnani (2012). “ETS: Discriminative Edit Models for Paraphrase Scoring”. In: **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*. Montréal, Canada: Association for Computational Linguistics, pp. 529–535. URL: `https://www.aclweb.org/anthology/S12-1076`.

Heilman, Michael and Noah A Smith (2010). “Good question! statistical ranking for question generation”. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 609–617.

Hermjakob, Ulf, Jonathan May, Michael Pust, and Kevin Knight (July 2018). “Translating a Language You Don’t Know In the Chinese Room”. In: *Proceedings of ACL 2018, System Demonstrations*. Melbourne, Australia: Association for Computational Linguistics, pp. 62–67. URL: `https://www.aclweb.org/anthology/P18-4011`.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural Computation* 9.8, pp. 1735–1780.

Hopfield, J. J. (1982). “Neural networks and physical systems with emergent collective computational abilities”. In: *Proceedings of the National Academy of Sciences of the USA*. Vol. 79. 8, pp. 2554–2558.

Hu, Chang, Benjamin B Bederson, and Philip Resnik (2010). “Translation by iterative collaboration between monolingual users”. In: *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pp. 54–55.

Hu, Chang, Benjamin B Bederson, Philip Resnik, and Yakov Kronrod (2011). “Monotrans2: A new human computation system to support monolingual translation”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1133–1136.

Huang, Yun, J Guerra, and Peter Brusilovsky (2016). “Modeling Skill Combination Patterns for Deeper Knowledge Tracing”. In: *Proceedings of the 6th Workshop on Personalization Approaches in Learning Environments (PALE 2016)*. 24th Conference on User Modeling, Adaptation and Personalization. Halifax, Canada.

Huckin, Thomas and James Coady (1999). “Incidental vocabulary acquisition in a second language”. In: *Studies in Second Language Acquisition* 21.2, pp. 181–193.

Kann, Katharina, Ryan Cotterell, and Hinrich Schütze (Apr. 2017). “Neural Multi-Source Morphological Reinflection”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 514–524. URL: `https://www.aclweb.org/anthology/E17-1049`.

Khajah, Mohammad, Rowan Wing, Robert Lindsey, and Michael Mozer (2014a). “Integrating latent-factor and knowledge-tracing models to predict individual differences in learning”. In: *Proceedings of the 7th International Conference on Educational Data Mining*.

Khajah, Mohammad M, Yun Huang, José P González-Brenes, Michael C Mozer, and Peter Brusilovsky (2014b). “Integrating knowledge tracing and item response theory: A tale of two frameworks”. In: *Proceedings of Workshop on Personalization Approaches in Learning Environments (PALE 2014) at the 22th International Conference on User Modeling, Adaptation, and Personalization*. University of Pittsburgh, pp. 7–12.

Kim, Haeyoung and Stephen Krashen (1998). “The Author Recognition and Magazine Recognition Tests, and Free Voluntary Rereading as Predictors of Vocabulary Development in English as a Foreign Language for Korean High School Students.” In: *System* 26.4, pp. 515–23.

Kim, Yoon, Yacine Jernite, David Sontag, and Alexander M. Rush (2016). “Character-aware neural language models”. In: *Thirtieth AAAI Conference on Artificial Intelligence*.

Kingma, Diederik P. and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

Kingma, Diederik P and Max Welling (2013). “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114*.

Knowles, Rebecca, Adithya Renduchintala, Philipp Koehn, and Jason Eisner (Aug. 2016). “Analyzing Learner Understanding of Novel L2 Vocabulary”. In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. To appear. Berlin, Germany: Association for Computational Linguistics, pp. 126–135. URL: `http://www.aclweb.org/anthology/K16-1013`.

Koedinger, K. R., P. I. Pavlick Jr., J. Stamper, T. Nixon, and S. Ritter (2011). "Avoiding Problem Selection Thrashing with Conjunctive Knowledge Tracing". In: *Proceedings of the 4th International Conference on Educational Data Mining*. Eindhoven, NL, pp. 91–100.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst (2007). "Moses: Open Source Toolkit for Statistical Machine Translation". In: *Proceedings of ACL: Short Papers*, pp. 177–180. URL: `http://www.aclweb.org/anthology/P/P07/P07-2045`.

Krashen, S. (1993). "How Well do People Spell?" In: *Reading Improvement* 30.1. URL: `http://www.sdkrashen.com/content/articles/1993\_how\_well\_do\_people\_spell.pdf`.

Krashen, Stephen (1989). "We acquire vocabulary and spelling by reading: Additional evidence for the input hypothesis". In: *The Modern Language Journal* 73.4, pp. 440–464.

Krashen, Stephen D (2003). *Explorations in language acquisition and use*. Heinemann Portsmouth, NH.

Labutov, Igor and Hod Lipson (June 2014). "Generating Code-switched Text for Lexical Learning". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 562–571. URL: https://www.aclweb.org/anthology/P14-1053.

Lee, Jung In and Emma Brunskill (2012). "The Impact on Individualizing Student Models on Necessary Practice Opportunities." In: *International Educational Data Mining Society*.

Lee, Sy-Ying and Stephen Krashen (1996). "Free Voluntary Reading and Writing Competence in Taiwanese High School Students". In: *Perceptual and Motor Skills* 83.2, pp. 687–690. eprint: https://doi.org/10.2466/pms.1996.83.2.687. URL: https://doi.org/10.2466/pms.1996.83.2.687.

Lee, Yon Ok, Stephen D Krashen, and Barry Gribbons (1996). "The effect of reading on the acquisition of English relative clauses". In: *ITL-International Journal of Applied Linguistics* 113.1, pp. 263–273.

Leitner, Sebastian (1972). *So lernt man lernen: der Weg zum Erfolg*. Herder, Freiburg.

Lingua.ly (2013). *Lingua.ly*. https://lingua.ly/. Accessed: 2016-04-04.

Litman, Diane (2016). "Natural Language Processing for Enhancing Teaching and Learning". In: *Proceedings of AAAI*.

Madnani, Nitin, Michael Heilman, Joel Tetreault, and Martin Chodorow (2012). "Identifying High-Level Organizational Elements in Argumentative Discourse". In: *Proceedings of NAACL-HLT*. Association for Computational Linguistics, pp. 20–28.

Merity, Stephen, Caiming Xiong, James Bradbury, and Richard Socher (2016). "Pointer sentinel mixture models". In: *arXiv preprint arXiv:1609.07843*.

Mikolov, Tomas, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur (2010). “Recurrent neural network based language model.” In: *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pp. 1045–1048.

Mikolov, Tomas, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky (2011). “RNNLM—Recurrent Neural Network Language Modeling Toolkit”. In: *Proc. of the 2011 ASRU Workshop*, pp. 196–201.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013). “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*, pp. 3111–3119.

Mikolov, Tomas, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin (May 2018). “Advances in Pre-Training Distributed Word Representations”. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. Miyazaki, Japan: European Languages Resources Association (ELRA). URL: `https://www.aclweb.org/anthology/L18-1008`.

Mitkov, Ruslan and Le An Ha (2003). “Computer-aided generation of multiple-choice tests”. In: *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing-Volume 2*. Association for Computational Linguistics, pp. 17–22.

Mousa, Amr and Björn Schuller (Apr. 2017). “Contextual Bidirectional Long Short-Term Memory Recurrent Neural Network Language Models: A Generative Approach to

Sentiment Analysis”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain, pp. 1023–1032. URL: `https://www.aclweb.org/anthology/E17-1096`.

Murphy, Kevin P., Yair Weiss, and Michael I. Jordan (1999). “Loopy belief propagation for approximate inference: An empirical study”. In: *Proceedings of UAI*. Morgan Kaufmann Publishers Inc., pp. 467–475.

Nelson, Mark (2007). *The Alpheios Project*. `http://alpheios.net/`. Accessed: 2016-04-05.

Nivre, Joakim, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, et al. “Universal Dependencies v1: A Multilingual Treebank Collection.” In:

OneThirdStories (2018). *OneThirdStories*. `https://onethirdstories.com/`. Accessed: 2019-02-20.

Özbal, Gözde, Daniele Pighin, and Carlo Strapparava (2014). “Automation and Evaluation of the Keyword Method for Second Language Learning”. In: *Proceedings of ACL (Volume 2: Short Papers)*, pp. 352–357.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). “GLoVe: Global Vectors for Word Representation”. In: *Proceedings of EMNLP*. Vol. 14, pp. 1532–1543. URL: `http://www.aclweb.org/anthology/D14-1162`.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (June 2018). “Deep Contextualized Word Representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association* *for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 2227–2237. URL: https://www.aclweb.org/anthology/N18-1202.

Philips, Lawrence (Dec. 1990). “Hanging on the Metaphone”. In: *Computer Language* 7.12.

Piech, Chris, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein (2015). “Deep Knowledge Tracing”. In: *Advances in Neural Information Processing Systems*, pp. 505–513.

Posner, Michael I (1989). *Foundations of cognitive science*. MIT press Cambridge, MA.

Preacher, K. J. (May 2002). *Calculation for the test of the difference between two independent correlation coefficients [Computer software]*. URL: http://www.quantpsy.org/corrtest/corrtest.htm.

Press, Ofir and Lior Wolf (Apr. 2017). “Using the Output Embedding to Improve Language Models”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain, pp. 157–163. URL: https://www.aclweb.org/anthology/E17-2025.

Rafferty, Anna N and Christopher D Manning (2008). “Parsing Three German Treebanks: Lexicalized and Unlexicalized Baselines”. In: *Proceedings of the Workshop on Parsing German*. Association for Computational Linguistics, pp. 40–46.

Recht, Benjamin, Christopher Re, Stephen Wright, and Feng Niu (2011). “Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent”. In: *Advances in Neural Information Processing Systems*, pp. 693–701.

Renduchintala, Adithya, Philipp Koehn, and Jason Eisner (Aug. 2017). "Knowledge Tracing in Sequential Learning of Inflected Vocabulary". In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada, pp. 238–247. URL: `https://www.aclweb.org/anthology/K17-1025`.

Renduchintala, Adithya, Philipp Koehn, and Jason Eisner (Aug. 2019a). "Simple Construction of Mixed-Language Texts for Vocabulary Learning". In: *Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*. Florence. URL: `http://cs.jhu.edu/~jason/papers/\#renduchintala-et-al-2019`.

Renduchintala, Adithya, Philipp Koehn, and Jason Eisner (Nov. 2019b). "Spelling-Aware Construction of Macaronic Texts for Teaching Foreign-Language Vocabulary". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 6438–6443. URL: `https://www.aclweb.org/anthology/D19-1679`.

Renduchintala, Adithya, Rebecca Knowles, Philipp Koehn, and Jason Eisner (Aug. 2016a). "Creating Interactive Macaronic Interfaces for Language Learning". In: *Proceedings of ACL-2016 System Demonstrations*. Berlin, Germany, pp. 133–138. URL: `https://www.aclweb.org/anthology/P16-4023`.

Renduchintala, Adithya, Rebecca Knowles, Philipp Koehn, and Jason Eisner (Aug. 2016b). "User Modeling in Language Learning with Macaronic Texts". In: *Proceedings of the*

54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany, pp. 1859–1869. URL: `https://www.aclweb.org/anthology/P16-1175`.

Rodrigo, Victoria, Jeff McQuillan, and Stephen Krashen (1996). "Free Voluntary Reading and Vocabulary Knowledge in Native Speakers of Spanish". In: *Perceptual and Motor Skills* 83.2, pp. 648–650. eprint: `https://doi.org/10.2466/pms.1996.83.2.648`. URL: `https://doi.org/10.2466/pms.1996.83.2.648`.

Schacter, D. L. (1989). "Memory". In: *Foundations of Cognitive Science*. Ed. by M. I. Postner. MIT Press, pp. 683–725.

Settles, Burr and Brendan Meeder (Aug. 2016). "A Trainable Spaced Repetition Model for Language Learning". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. Berlin, Germany: Association for Computational Linguistics, pp. 1848–1858. URL: `http://www.aclweb.org/anthology/P16-1174`.

Smolensky, Paul (1986). "Information Processing in Dynamical Systems: Foundations of Harmony Theory". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Ed. by D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. Vol. 1: Foundations. Cambridge, MA: MIT Press/Bradford Books, pp. 194–281.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15, pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber (2015). "Highway networks". In: *arXiv preprint arXiv:1505.00387*.

Stokes, Jeffery, Stephen D Krashen, and John Kartchner (1998). "Factors in the acquisition of the present subjunctive in Spanish: The role of reading and study". In: *ITL-International Journal of Applied Linguistics* 121.1, pp. 19–25.

Swych (2015). *Swych*. http://swych.it/. Accessed: 2019-02-20.

Tieleman, Tijmen and Geoffrey Hinton (2012). "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2.

Vilnis, Luke and Andrew McCallum (2014). "Word Representations via Gaussian Embedding". In: *CoRR* abs/1412.6623.

Vygotsky, Lev (2012). *Thought and Language (Revised and Expanded Edition)*. MIT Press.

Weide, R. (1998). *The CMU pronunciation dictionary, release 0.6*.

Wieting, John, Mohit Bansal, Kevin Gimpel, and Karen Livescu (Nov. 2016). "Charagram: Embedding Words and Sentences via Character n-grams". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pp. 1504–1515. URL: https://www.aclweb.org/anthology/D16-1157.

Wikimedia Foundation (2016). *Simple English Wikipedia*. Retrieved from https://dumps.wikimedia.org/simplewiki/20160407/ 8-April-2016.

Wikipedia (2016). *Leichte Sprache — Wikipedia, Die freie Enzyklopädie*. [Online; accessed 16-March-2016]. URL: `https://de.wikipedia.org/wiki/Leichte\_Sprache`.

Wood, David, Jerome S. Bruner, and Gail Ross (1976). "The role of tutoring in problem solving". In: *Journal of Child Psychology and Psychiatry* 17.2, pp. 89–100.

Wu, Dekai (1997). "Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora". In: *Computational Linguistics* 23.3, pp. 377–404.

Xu, Yanbo and Jack Mostow (2012). "Comparison of Methods to Trace Multiple Subskills: Is LR-DBN Best?" In: *Proceedings of the 5th International Conference on Educational Data Mining*, pp. 41–48.

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le (2019). "Xlnet: Generalized autoregressive pretraining for language understanding". In: *Advances in neural information processing systems*, pp. 5753–5763.

Zeiler, Matthew D (2012). "ADADELTA: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701*.

Zhang, Haoran, Ahmed Magooda, Diane Litman, Richard Correnti, Elaine Wang, LC Matsmura, Emily Howe, and Rafael Quintana (2019). "eRevise: Using Natural Language Processing to Provide Formative Feedback on Text Evidence Usage in Student Writing". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 9619–9625.

Zhou, Jingguang and Zili Huang (2018). “Recover missing sensor data with iterative imputing network”. In: *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

Zolf, Flak (1945). *On Stranger Land: Pages of a Life*.

Zolf, Flak and Martin Green (2003). *On Foreign Soil: Tales of a Wandering Jew*. Benchmark Publishing.

Vita

Adithya Renduchintala

arendu.github.io

adi.r@jhu.edu

INTERESTS

I am broadly interested in problems at the intersection of (Deep) Machine Learning, Neural Machine Translation, Natural Language Processing, & User Modeling

EDUCATION

PhD, Computer Science 2013 - 2020

Johns Hopkins University, Baltimore, MD

MS, Computer Science 2010 - 2012

University of Colorado, Boulder, CO

VITA

MS, Electrical Engineering, Arts Media and Engineering 2005-2008

Arizona State University, Tempe, AZ

BE, Electrical Engineering 2001-2005

Anna University, SRM Engineering College, Chennai, India

PUBLICATIONS

1. Spelling-Aware Construction of Macaronic Texts for Teaching Foreign-Language Vocabulary, **Adithya Renduchintala**, Philipp Koehn and Jason Eisner. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing

2. Simple Construction of Mixed-Language Texts for Vocabulary Learning, **Adithya Renduchintala**, Philipp Koehn and Jason Eisner. Annual Meeting of the Association for Computational Linguistics (ACL) Workshop on Innovative Use of NLP for Building Educational Applications, 2019.

3. Pretraining by Backtranslation for End-to-End ASR in Low-Resource Settings, Matthew Wiesner, **Adithya Renduchintala**, Shinji Watanabe, Chunxi Li, Najim Dehak and Sanjeev Khudanpur, Interspeech 2019

4. A Call for prudent choice of Subword Merge Operations, Shuoyang Ding, **Adithya Renduchintala**, and Kevin Duh. Machine Translation Summit 2019.

5. Character-Aware Decoder for Translation into Morphologically Rich Languages, **Adithya Renduchintala***, Pamela Shapiro*, Kevin Duh and Philipp Koehn. Machine Translation Summit 2019.

6. The JHU/KyotoU Speech Translation System for IWSLT 2018, Hirofumi Inaguma, Xuan Zhang, Zhiqi Wang, **Adithya Renduchintala**, Shinji Watanabe and Kevin Duh. The International Workshop on Spoken Language Translation 2018 (IWSLT)

7. Multi-Modal Data Augmentation for End-to-End ASR **Adithya Renduchintala**, Shuoyang Ding, Matthew Wiesner and Shinji Watanabe, Interspeech 2018. Best Student Paper Award (3/700+)

8. ESPnet: End-to-End Speech Processing Toolkit, Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, **Adithya Renduchintala** and Tsubasa Ochiai, Interspeech 2018.

9. Knowledge Tracing in Sequential Learning of Inflected Vocabulary, **Adithya Renduchintala**, Philipp Koehn and Jason Eisner, Conference on Computational Natural Language Learning (CoNLL), 2017.

10. User Modeling in Language Learning with Macaronic Texts, **Adithya Renduchintala**, Rebecca Knowles, Philipp Koehn, and Jason Eisner. Annual Meeting of the

* Equal contribution

Association for Computational Linguistics (ACL) 2016.

11. Creating interactive macaronic interfaces for language learning, **Adithya Renduchintala**, Rebecca Knowles, Philipp Koehn, and Jason Eisner. Annual Meeting of the Association for Computational Linguistics (ACL) Demo Session 2016.

12. Analyzing learner understanding of novel L2 vocabulary, Rebecca Knowles, **Adithya Renduchintala**, Philipp Koehn, and Jason Eisner, Conference on Computational Natural Language Learning (CoNLL), 2016.

13. Algerian Arabic-French Code-Switched Corpus, Ryan Cotterell, **Adithya Renduchintala**, Naomi P. Saphra and Chris Callison-Burch. An LREC-2014 Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools. 2014.

14. Using Machine Learning and HL7 LOINC DO for Classification of Clinical Documents, **Adithya Renduchintala**, Amy Zhang, Thomas Polzin, G. Saadawi. American Medical Informatics Association (AMIA) 2013.

15. Collaborative Tagging and Persistent Audio Conversations, Ajita John, Shreeharsh Kelkar, Ed Peebles, **Adithya Renduchintala**, Doree Seligmann Web 2.0 and Social Software Workshop in Conjunction with ECSCW. 2007.

16. Designing for persistent Audio Conversations in the Enterprise, **Adithya Renduchintala**, Ajita John, Shreeharsh Kelkar, and Doree Duncan-Seligmann. Design for User Experience. 2007.

VITA

EXPERIENCE

Facebook AI, Menlo Park, CA **2020-Present**

Research Scientist

Working on problems related to Neural Machine Translation.

Johns Hopkins University, Baltimore, MD **2013-2020**

Research Assistant

Designed and evaluated AI foreign language teaching systems. Also worked on Machine Translation and End-to-End Speech Recognition.

Duolingo, Pittsburgh, PA **Summer 2017**

Research Intern

Prototyped a Chatbot system that detects and corrects word-ordering errors made by language learners. Explored spelling-error robustness of compositional word embeddings

M*Modal, Pittsburgh, PA **2012-2013**

NLP Engineer

Developed SVM based clinical document classification system. Worked on feature engineering for statistical models (Document Classification, Entity Detection, Tokenization, Chunking)

Rosetta Stone, Boulder, CO **2008-2012**

Software Developer

Designed, prototyped and evaluated speech recognition based games and applications for language learning. Prototyped a image-to-concept relation visualization tool for second language vocabulary learning.

Avaya, Lincroft NJ **Summer 2007**

Research Scientist Intern

Developed an interactive graph based visualization tool to explore and annotate conference calls in enterprises.

Arizona State University, Tempe AZ **2006-2008**

Research Assistant, Arts Media & Engineering

Designed and prototyped systems for serendipitous interactions in distributed workplaces.

TEACHING

1. Intro. to Human Language Technology, Teaching Assistant Fall, 2019
2. Neural Machine Translation Lab Session, JSALT Summer School Summer 2018
3. Machine Translation, Teaching Assistant Spring 2016
4. Intro. to Programming for Scientists & Engineers, Teaching Assistant Fall 2013

PROGRAMMING

Advanced: Python (numpy, scipy, scikit-learn)

Proficient: Java, C/C++, Javascript, Jquery, NodeJs

Deep Learning Frameworks: PyTorch (Advanced), MxNet, Tensorflow

Deep Learning Toolkits: OpenNMT, Fairseq, ESPNet, Sockeye

NATIONALITY

Indian, Permanent US Resident

Updated 08/28/2020