Don't Until the Final Verb Wait: Reinforcement Learning for Simultaneous Machine Translation

Alvin C. Grissom II
and **Jordan Boyd-Graber**
Computer Science
University of Colorado
Boulder, CO
Alvin.Grissom@colorado.edu
Jordan.Boyd.Graber@colorado.edu

He He, John Morgan,
and **Hal Daumé III**
Computer Science and UMIACS
University of Maryland
College Park, MD
{hhe,jjm,hal}@cs.umd.edu

Abstract

We introduce a reinforcement learning-based approach to simultaneous machine translation—producing a translation while receiving input words—between languages with drastically different word orders: from verb-final languages (e.g., German) to verb-medial languages (English). In traditional machine translation, a translator must "wait" for source material to appear before translation begins. We remove this bottleneck by predicting the final verb in advance. We use reinforcement learning to learn when to trust predictions about unseen, future portions of the sentence. We also introduce an evaluation metric to measure expeditiousness and quality. We show that our new translation model outperforms batch and monotone translation strategies.

1 Introduction

We introduce a simultaneous machine translation (MT) system that predicts unseen verbs and uses reinforcement learning to learn when to trust these predictions and when to wait for more input.

Simultaneous translation is producing a partial translation of a sentence before the input sentence is complete, and is often used in important diplomatic settings. One of the first noted uses of human simultaneous interpretation was the Nuremberg trials after the Second World War. Siegfried Ramler (2009), the Austrian-American who organized the translation teams, describes the linguistic predictions and circumlocutions that translators would use to achieve a tradeoff between translation latency and accuracy. The audio recording technology used by those interpreters sowed the seeds of technology-assisted interpretation at the United Nations (Gaiba, 1998).

Performing real-time translation is especially difficult when information that comes early in the target language (the language you're translating *to*) comes late in the source language (the language you're translating *from*). A common example is when translating from a verb-final (SOV) language (e.g., German or Japanese) to a verb-medial (SVO) language, (e.g., English). In the example in Figure 1, for instance, the main verb of the sentence (in **bold**) appears at the end of the German sentence. An offline (or "batch") translation system waits until the end of the sentence before translating anything. While this is a reasonable approach, it has obvious limitations. Real-time, interactive scenarios—such as online multilingual video conferences or diplomatic meetings—require comprehensible partial interpretations *before* a sentence ends. Thus, a significant goal in interpretation is to make the translation as **expeditious** as possible.

We present three components for an SOV-to-SVO simultaneous MT system: a reinforcement learning framework that uses predictions to create expeditious translations (Section 2), a system to predict how a sentence will end (e.g., predicting the main verb; Section 4), and a metric that balances quality and expeditiousness (Section 3). We combine these components in a framework that learns *when* to begin translating sections of a sentence (Section 5).

Section 6 combines this framework with a

ich	bin	mit	dem	Zug	nach	Ulm **gefahren**
I	am	with	the	train	to	Ulm **traveled**
I		*(. . . . . waiting. . . . .)*				**traveled** by train to Ulm

Figure 1: An example of translating from a verb-final language to English. The verb, in **bold**, appears at the end of the sentence, preventing coherent translations until the final source word is revealed.

translation system that produces simultaneous translations. We show that our data-driven system can successfully predict unseen parts of the sentence, learn when to trust them, and outperform strong baselines (Section 7).

While some prior research has approached the problem of simultaneous translation—we review these systems in more detail in Section 8—no current model learns *when* to definitively begin translating chunks of an incomplete sentence. Finally, in Section 9, we discuss the limitations of our system: it only uses the most frequent source language verbs, it only applies to sentences with a single main verb, and it uses an idealized translation system. However, these limitations are not insurmountable; we describe how a more robust system can be assembled from these components.

2 Decision Process for Simultaneous Translation

Human interpreters learn strategies for their profession with experience and practice. As words in the source language are observed, a translator—human or machine—must decide whether and how to translate, while, for certain language pairs, simultaneously predicting future words. We would like our system to do the same. To this end, we model simultaneous MT as a Markov decision process (MDP) and use reinforcement learning to effectively combine predicting, waiting, and translating into a coherent strategy.

2.1 States: What is, what is to come

The **state** s_t represents the current view of the world given that we have seen t words of a source language sentence.[1] The state contains information both about what is known and what is predicted based on what is known. To compare the system to a human translator in a decision-making process, the state is akin to the translator's cognitive state. At any given time, we have *knowledge* (observations) and *beliefs* (predictions) with varying degrees of certainty: that is, the state contains the revealed words $x_{1:t}$ of a sentence; the state also contains predictions about the remainder of the sentence: we predict the next word in the sentence and the final verb.

More formally, we have a prediction at time t of the next source language word that will appear, $n_{t+1}^{(t)}$, and for the final verb, $v^{(t)}$. For example, given the partial observation "`ich bin mit dem`", the state might contain a prediction that the next word, $n_{t+1}^{(t)}$, will be "`Zug`" and that the final verb $v^{(t)}$ will be "`gefahren`".

We discuss the mechanics of next-word and verb prediction further in Section 4; for now, consider these black boxes which, after observing every new source word x_t, make predictions of future words in the source language. This representation of the state allows for a richer set of actions, described below, permitting simultaneous translations that outpace the source language input[2] by predicting the future.

2.2 Actions: What our system can do

Given observed and hypothesized input, our simultaneous translation system must decide when to translate them. This is expressed in the form of four actions: our system can **commit** to a partial translation, predict the **next word** and use it to update the translation, predict the **verb** and use it to update the translation, or **wait** for more words.

We discuss each of these actions in turn before describing how they come together to incrementally translate an entire sentence:

Wait Waiting is the simplest action. It produces no output and allows the system to receive more input, biding its time, so that when it does choose to translate, the translation is based on more information.

Commit Committing produces translation output: given the observed source sentence, produce the best translation possible.

[1] We use t to evoke a discrete version of time. We only allow actions after observing a complete source word.

[2] Throughout, "input" refers to source language input, and "output" refers to target language translation.

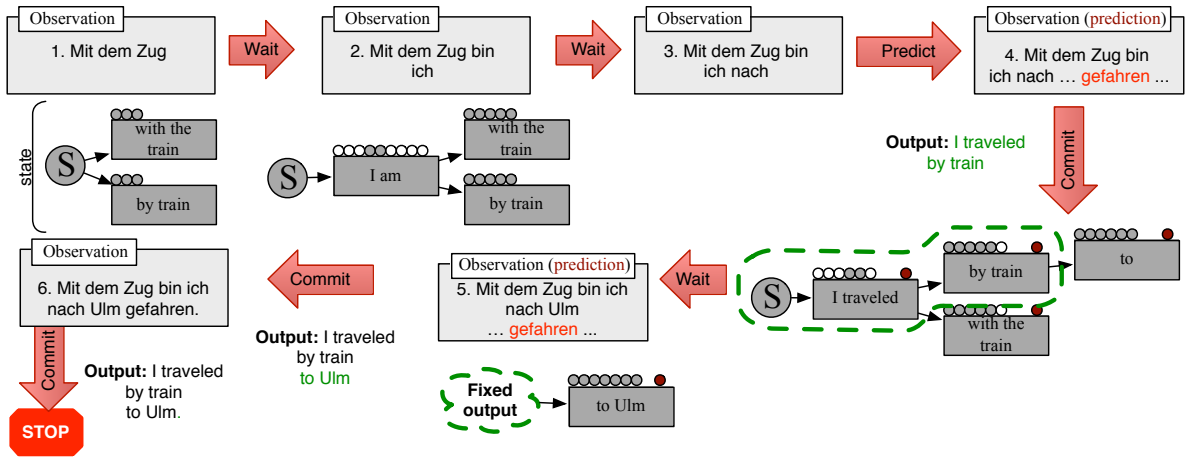

Figure 2: A simultaneous translation from source (German) to target (English). The agent chooses to wait until after (3). At this point, it is sufficiently confident to predict the final verb of the sentence (4). Given this additional information, it can now begin translating the sentence into English, constraining future translations (5). As the rest of the sentence is revealed, the system can translate the remainder of the sentence.

Next Word The **next word** action takes a prediction of the next source word and produces an updated translation based on that prediction, i.e., appending the predicted word to the source sentence and translating the new sentence.

Verb Our system can also predict the source sentence's final verb (the last word in the sentence). When our system takes the **verb** action, it uses its verb prediction to update the translation using the prediction, by placing it at the end of the source sentence.

We can recreate a traditional batch translation system (interpreted temporally) by a sequence of **wait** actions until all input is observed, followed by a **commit** to the complete translation. Our system can **commit** to partial translations if it is confident, but producing a good translation early in the sentence often depends on missing information.

2.3 Translation Process

Having described the state, its components, and the possible actions at a state, we present the process in its entirety. In Figure 2, after each German word is received, the system arrives at a new *state*, which consists of the source input, target translation so far, and predictions of the unseen words. The translation system must then take an *action* given information about the current state. The action will result in receiving and translating more source words, transitioning the system to the next state. In the example, for the first few source-language words, the translator lacks the confidence to produce any output due to insufficient information at the state. However, after State 3, the state shows high confidence in the predicted verb "gefahren". Combined with the German input it has observed, the system is sufficiently confident to act on that prediction to produce English translation.

2.4 Consensus Translations

Three straightforward actions—**commit**, **next word**, and **verb**—all produce translations. These rely black box access to a translation (discussed in detail in Section 6): that is, given a source language sentence fragment, the translation system produces a target language sentence fragment.

Because these actions can happen more than once in a sentence, we must form a single consensus translation from all of the translations that we might have seen. If we have only one translation or if translations are identical, forming the consensus translation is trivial. But how should we resolve conflicting translations?

Any time our system chooses an action that

produces output, the observed input (plus extra predictions in the case of **next-word** or **verb**), is passed into the translation system. That system then produces a complete translation of its input fragment.

Any new words—i.e., words whose target index is greater than the length of any previous translation—are appended to the previous translation.[3] Table 1 shows an example of forming these consensus translations.

Now that we have defined how states evolve based on our system's actions, we need to know how to select which actions to take. Eventually, we will formalize this as a learned policy (Section 5) that maps from states to actions. First, however, we need to define a reward that measures how "good" an action is.

3 Objective: What is a good simultaneous translation?

Good simultaneous translations must optimize two objectives that are often at odds, i.e., producing translations that are, in the end, accurate, and producing them in pieces that are presented expeditiously. While there are existing automated metrics for assessing translation quality (Papineni et al., 2002; Banerjee and Lavie, 2005; Snover et al., 2006), these must be modified to find the necessary compromise between translation quality and expeditiousness. That is, a good metric for simultaneous translation must achieve a balance between translating chunks early and translating accurately. All else being equal, maximizing either goal in isolation is trivial: for accurate translations, use a **batch** system and wait until the sentence is complete, translating it all at once; for a maximally expeditious translation, create **monotone** translations, translating each word as it appears, as in Tillmann et al. (1997) and Pytlik and Yarowsky (2006). The former is not simultaneous at all; the latter is mere word-for-word replacement and results in awkward, often unintelligible translations of distant language pairs.

Once we have predictions, we have an expanded array of possibilities, however. On one extreme, we can imagine a **psychic** translator—one that can completely translate an imagined sentence after one word is uttered—as an unobtainable system. On the other extreme is a standard **batch** translator, which waits until it has access to the utterer's complete sentence before translating anything.

Again, we argue that a system can improve on this by *predicting* unseen parts of the sentence to find a better tradeoff between these conflicting goals. However, to evaluate and optimize such a system, we must measure where a system falls on the continuum of accuracy versus expeditiousness.

Consider partial translations in a two-dimensional space, with time (quantized by the number of source words seen) increasing from left to right on the x axis and the BLEU score (including brevity penalty against the reference length) on the y axis. At each point in time, the system may add to the consensus translation, changing the precision (Figure 3). Like an ROC curve, a good system will be high and to the left, optimizing the area under the curve: the ideal system would produce points as high as possible immediately. A translation which is, in the end, accurate, but which is less expeditious, would accrue its score more slowly but outperform a similarly expeditious system which nevertheless translates poorly.

An idealized psychic system achieves this, claiming all of the area under the curve, as it would have a perfect translation instantly, having no need of even waiting for future input.[4] A batch system has only a narrow (but tall) sliver to the right, since it translates nothing until all of the words are observed.

Formally, let Q be the score function for a partial translation, $\boldsymbol{x}$ the sequentially revealed source words $x_1, x_2, \ldots, x_T$ from time step 1 to T, and $\boldsymbol{y}$ the partial translations $y_1, y_2, \ldots, y_T$, where T is the length of the source language input. Each incremental translation y_t has a BLEU-n score with respect to a reference $\boldsymbol{r}$. We apply the usual BLEU brevity penalty to all the incremental translations (initially empty) to

[3] Using constrained decoding to enforce consistent translation prefixes would complicate our method but is an appealing extension.

[4] One could reasonably argue that this is not ideal: a fluid conversation requires the prosody and timing between source and target to match exactly. Thus, a psychic system would provide too much information too quickly, making information exchange unnatural. However, we take the information-centric approach: more information faster is better.

Pos	Input	Intermediate	Consensus
1			
2	Er	He_1	He_1
3	Er wurde *gestaltet*	It_1 was_2 designed_3	He_1 was_2 designed_3
4		It_1 was_2 designed_3	He_1 was_2 designed_3
5	Er wurde gestern renoviert	It_1 was_2 renovated_3 yesterday_4	He_1 was_2 designed_3 yesterday_4

Table 1: How intermediate translations are combined into a consensus translation. Incorrect translations (e.g., "he" for an inanimate object in position 3) and incorrect predictions (e.g., incorrectly predicting the verb *gestaltet* in position 5) are kept in the consensus translation. When no translation is made, the consensus translation remains static.

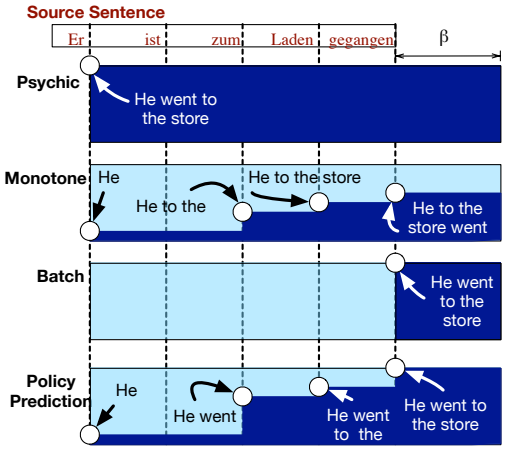

Figure 3: Comparison of LBLEU (the area under the curve given by Equation 1) for an impossible psychic system, a traditional batch system, a monotone (German word order) system, and our prediction-based system. By correctly predicting the verb "gegangen" (to go), we achieve a better overall translation more quickly.

obtain **latency-bleu** (LBLEU),

$$Q(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{T} \sum_t \text{BLEU}(y_t, r) + T \cdot \text{BLEU}(y_T, r) \quad (1)$$

The LBLEU score is a word-by-word integral across the input source sentence. As each source word is observed, the system receives a reward based on the BLEU score of the partial translation. LBLEU, then, represents the sum of these T rewards at each point in the sentence. The score of a simultaneous translation is the sum of the scores of all individual segments that contribute to the overall translation.

We multiply the final BLEU score by T to ensure good final translations in learned systems to compensate for the implicit bias toward low latency.[5]

4 Predicting Verbs and Next Words

The **next** and **verb** actions depend on predictions of the sentence's next word and final verb; this section describes our process for predicting verbs and next words given a partial source language sentence.

The prediction of the next word in the source language sentence is modeled with a left-to-right language model. This is (naïvely) analogous to how a human translator might use his own "language model" to guess upcoming words to gain some speed by completing, for example, collocations before they are uttered. We use a simple bigram language model for next-word prediction. We use Heafield et al. (2013).

For verb prediction, we use a generative model that combines the prior probability of a particular verb v, $p(v)$, with the likelihood of the source context at time t given that verb (namely, $p(x_{1:t} \mid v)$), as estimated by a smoothed Kneser-Ney language model (Kneser and Ney, 1995). We use Pauls and Klein (2011). The prior probability $p(v)$ is estimated by simple relative frequency estimation. The context, $x_{1:t}$, consists of all words observed. We model $p(x_{1:t} \mid v)$ with verb-specific n-gram language models. The predicted verb $v^{(t)}$ at time t is then:

$$\arg\max_v p(v) \prod_{i=1}^{t} p(x_i \mid v, x_{i-n+1:i-1}) \quad (2)$$

[5] One could replace T with a parameter, β, to bias towards different kinds of simultaneous translations. As $\beta \to \infty$, we recover batch translation.

where $x_{i-n+1:i-1}$ is the $n-1$-gram context. To narrow the search space, we consider only the 100 most frequent final verbs, where a "final verb" is defined as the sentence-final sequence of verbs and particles as detected by a German part-of-speech tagger (Toutanova et al., 2003).[6]

5 Learning a Policy

We have a framework (states and actions) for simultaneous machine translation and a metric for assessing simultaneous translations. We now describe the use of reinforcement learning to learn a **policy**, a mapping from states to actions, to maximize LBLEU reward.

We use imitation learning (Abbeel and Ng, 2004; Syed et al., 2008): given an optimal sequence of actions, learn a generalized policy that maps states to actions. This can be viewed as a cost-sensitive classification (Langford and Zadrozny, 2005): a state is represented as a feature vector, the loss corresponds to the quality of the action, and the output of the classifier is the action that should be taken in that state.

In this section, we explain each of these components: generating an optimal policy, representing states through features, and learning a policy that can generalize to new sentences.

5.1 Optimal Policies

Because we will eventually learn policies via a classifier, we must provide training examples to our classifier. These training examples come from an **oracle policy** π^* that demonstrates the optimal sequence—i.e., with maximal LBLEU score—of actions for each sequence. Using dynamic programming, we can determine such actions for a fixed translation model.[7] From this oracle policy, we generate training examples for a supervised classifier. State s_t is represented as a tuple of the observed words $x_{1:t}$, predicted verb $v^{(t)}$, and the predicted word $n_{t+1}^{(t)}$. We represent the state to a classifier as a feature vector $\phi(x_{1:t}, n_{t+1}^{(t)}, v^{(t)})$.

[6]This has the obvious disadvantage of ignoring morphology and occasionally creating duplicates of common verbs that have may be associated with multiple particles; nevertheless, it provides a straightforward verb to predict.

[7]This is possible for the limited class of consensus translation schemes discussed in Section 2.4.

5.2 Feature Representation

We want a feature representation that will allow a classifier to generalize beyond the specific examples on which it is trained. We use several general classes of features: features that describe the input, features that describe the possible translations, and features that describe the quality of the predictions.

Input We include both a bag of words representation of the input sentence as well as the most recent word and bigram to model word-specific effects. We also use a feature that encodes the length of the source sentence.

Prediction We include the identity of the predicted verb and next word as well as their respective probabilities under the language models that generate the predictions. If the model is confident in the prediction, the classifier can learn to more so trust the predictions.

Translation In addition to the state, we include features derived from the possible actions the system might take. This includes a bag of words representation of the target sentence, the score of the translation (decreasing the score is undesirable), the score of the current consensus translation, and the difference between the current and potential translation scores.

5.3 Policy Learning

Our goal is to learn a classifier that can accurately mimic the oracle's choices on previously unseen data. However, at test time, when we run the learned policy classifier, the learned policy's state distribution may deviate from the optimal policy's state distribution due to imperfect imitation, arriving in states not on the oracle's path. To address this, we use SEARN (Daumé III et al., 2009), an iterative imitation learning algorithm. We learn from the optimal policy in the first iteration, as in standard supervised learning; in the following iterations, we run an interpolated policy

$$\pi_{k+1} = \epsilon\pi_k + (1-\epsilon)\pi^*, \tag{3}$$

with k as the iteration number and ϵ the mixing probability. We collect examples by asking the policy to label states on its path. The interpolated policy will execute the optimal action with probability $1-\epsilon$ and the learned

policy's action with probability ϵ. In the first iteration, we have $\pi_0 = \pi^*$.

Mixing in the learned policy allows the learned policy to slowly change from the oracle policy. As it trains on these no-longer-perfect state trajectories, the state distribution at test time will be more consistent with the states used in training.

SEARN learns the policy by training a cost-sensitive classifier. Besides providing the optimal action, the oracle must also assign a cost to an action

$$\mathcal{C}(a_t, \boldsymbol{x}) \equiv Q(\boldsymbol{x}, \pi^*(x_t)) - Q(\boldsymbol{x}, a_t(x_t)), \quad (4)$$

where $a_t(x_t)$ represents the translation outcome of taking action a_t. The cost is the regret of not taking the optimal action.

6 Translation System

The focus of this work is to show that given an effective batch translation system and predictions, we can learn a policy that will turn this into a simultaneous translation system. Thus, to separate translation errors from policy errors, we perform experiments with a nearly optimal translation system we call an *omniscient* translator.

More realistic translation systems will naturally lower the objective function, often in ways that make it difficult to show that we can effectively predict the verbs in verb-final source languages. For instance, German to English translation systems often drop the verb; thus, predicting a verb that will be ignored by the translation system will not be effective.

The omniscient translator translates a source sentence correctly once it has been fed the appropriate source words as input. There are two edge cases: empty input yields an empty output, while a complete, correct source sentence returns the correct, complete translation. Intermediate cases—where the input is either incomplete or incorrect—require using an alignment. The omniscient translator assumes as input a reference translation r, a partial source language input $x_{1:t}$ and a corresponding partial output y. In addition, the omniscient translator assumes access to an *alignment* between r and x. In practice, we use the HMM aligner (Vogel et al., 1996; Och and Ney, 2003).

We first consider incomplete but correct inputs. Let $y = \tau(x_{1:t})$ be the translator's output given a partial source input $x_{1:t}$ with translation y. Then, $\tau(x_{1:t})$ produces all target words y_j if there is a source word x_i in the input aligned to those words—i.e., $(i, j) \in a_{x,y}$—*and* all preceding target words can be translated. (That translations must be contiguous is a natural requirement for human recipients of translations). In the case where y_j is unaligned, the closest aligned target word to y_j that has a corresponding alignment entry is aligned to x_i; then, if x_i is present in the input, y_j appears in the output. Thus, our omniscient translation system will always produce the correct output given the correct input.

However, our learned policy can make wrong predictions, which can produce partial translations y that do *not* match the reference. In this event, an incorrect source word $\tilde{x}_i$ produces incorrect target words $\tilde{y}_j$, for all j: $(i, j) \in a_{x,y}$. These $\tilde{y}_j$ are sampled from the IBM Model 1 lexical probability table multiplied by the source language model $\tilde{y}_j \sim \text{Mult}(\theta_{\tilde{x}_i}) p_{LM}(\tilde{\boldsymbol{x}})$.[8] Thus, even if we predict the correct verb using a **next word** action, it will be in the wrong position and thus generate a translation from the lexical probabilities. Since translations based on Model 1 probabilities are generally inaccurate, the omniscient translator will do very well when given correct input but will produce very poor translations otherwise.

7 Experiments

In this section, we describe our experimental framework and results from our experiments. From aligned data, we derive an omniscient translator. We use monolingual data in the source language to train the verb predictor and the next word predictor. From these features, we compute an optimal policy from which we train a learned policy.

7.1 Data sets

For translation model and policy training, we use data from the German-English Parallel "de-news" corpus of radio broadcast news (Koehn, 2000), which we lower-cased and stripped of

[8] If a policy chooses an incorrect unaligned word, it has no effect on the output. Alignments are position-specific, so "wrong" refers to position and type.

punctuation. A total of 48,601 sentence pairs are randomly selected for building our system. Of these, we use 70% (34,528 pairs) for training word alignments.

For training the translation policy, we restrict ourselves to sentences that end with one of the 100 most frequent verbs (see Section 4). This results in a data set of 4401 training sentences and 1832 test sentences from the de-news data. We did this to narrow the search space (from thousands of possible, but mostly very infrequent, verbs).

We used 1 million words of news text from the Leipzig Wortschatz (Quasthoff et al., 2006) German corpus to train 5-gram language models to predict a verb from the 100 most frequent verbs.

For next-word prediction, we use the 18,345 most frequent German bigrams from the training set to provide a set of candidates in a language model trained on the same set. We use frequent bigrams to reduce the computational cost of finding the completion probability of the next word.

7.2 Training Policies

In each iteration of SEARN, we learn a multi-class classifier to implement the policy. The specific learning algorithm we use is AROW (Crammer et al., 2013). In the complete version of SEARN, the cost of each action is calculated as the highest expected reward starting at the current state minus the actual roll-out reward. However, computing the full roll-out reward is computationally very expensive. We thus use a surrogate binary cost: if the predicted action is the same as the optimal action, the cost is 0; otherwise, the cost is 1. We then run SEARN for five iterations. Results on the development data indicate that continuing for more iterations yields no benefit.

7.3 Policy Rewards on Test Set

In Figure 4, we show performance of the optimal policy *vis-à-vis* the learned policy, as well as the two baseline policies: the batch policy and the monotone policy. The x-axis is the percentage of the source sentence seen by the model, and the y-axis is a smoothed average of the reward as a function of the percentage of the sentence revealed. The monotone policy's performance is close to the optimal policy for

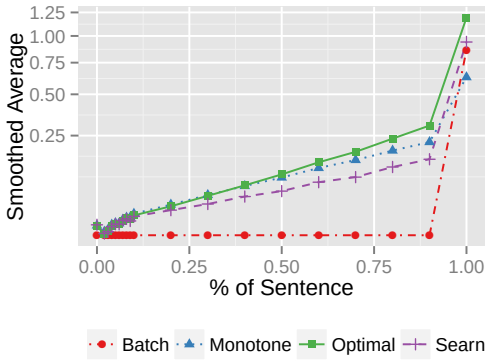

Figure 4: The final reward of policies on German data. Our policy outperforms all baselines by the end of the sentence.

Figure 5: Histogram of actions taken by the policies.

the first half of the sentence, as German and English have similar word order, though they diverge toward the end. Our learned policy outperforms the monotone policy toward the end and of course outperforms the batch policy throughout the sentence.

Figure 5 shows counts of actions taken by each policy. The batch policy always commits at the end. The monotone policy commits at each position. Our learned policy has an action distribution similar to that of the optimal policy, but is slightly more cautious.

7.4 What Policies Do

Figure 6 shows a policy that, predicting incorrectly, still produces sensible output. The policy correctly intuits that the person discussed

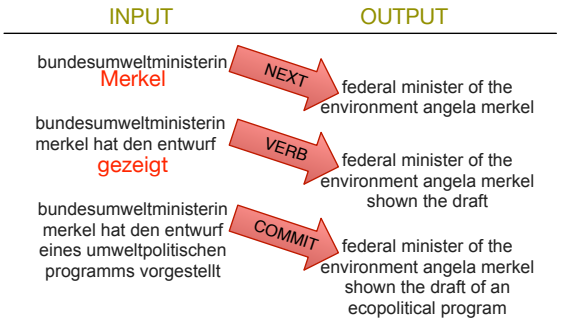

Figure 6: An imperfect execution of a learned policy. Despite choosing the wrong verb "gezeigt" (showed) instead of "vorgestellt" (presented), the translation retains the meaning.

is Angela Merkel, who was the environmental minister at the time, but the policy uses an incorrectly predicted verb. Because of our poor translation model (Section 6), it renders this word as "shown", which is a poor translation. However, it is still comprehensible, and the overall policy is similar to what a human would do: intuit the subject of the sentence from early clues and use a more general verb to stand in for a more specific one.

8 Related Work

Just as MT was revolutionized by statistical learning, we suspect that simultaneous MT will similarly benefit from this paradigm, both from a systematic system for simultaneous translation and from a framework for learning how to incorporate predictions.

Simultaneous translation has been dominated by rule and parse-based approaches (Mima et al., 1998a; Ryu et al., 2006). In contrast, although Verbmobil (Wahlster, 2000) performs incremental translation using a statistical MT module, its incremental decision-making module is rule-based. Other recent approaches in speech-based systems focus on waiting until a pause to translate (Sakamoto et al., 2013) or using word alignments (Ryu et al., 2012) between languages to determine optimal translation units.

Unlike our work, which focuses on prediction and learning, previous strategies for dealing with SOV-to-SVO translation use rule-based methods (Mima et al., 1998b) (for instance, passivization) to buy time for the translator to hear more information in a spoken context—or use phrase table and reordering probabilities to decide where to translate with less delay (Fujita et al., 2013). Oda et al. (2014) is the most similar to our work on the translation side. They frame word segmentation as an optimization problem, using a greedy search and dynamic programming to find segmentation strategies that maximize an evaluation measure. However, unlike our work, the direction of translation was from *from* SVO to SVO, obviating the need for verb prediction. Simultaneous translation is more straightforward for languages with compatible word orders, such as English and Spanish (Fügen, 2008).

To our knowledge, the only attempt to specifically predict verbs or any late-occurring terms (Matsubara et al., 2000) uses pattern matching on what would today be considered a small data set to predict English verbs for Japanese to English simultaneous MT.

Incorporating verb predictions into the translation process is a significant component of our framework, though n-gram models strongly prefer highly frequent verbs. Verb prediction might be improved by applying the insights from psycholinguistics. Ferreira (2000) argues that verb lemmas are required early in sentence production—prior to the first noun phrase argument—and that multiple possible syntactic hypotheses are maintained in parallel as the sentence is produced. Schriefers et al. (1998) argues that, in simple German sentences, non-initial verbs do not need lemma planning at all. Momma et al. (2014), investigating these prior claims, argues that the abstract relationship between the internal arguments and verbs triggers *selective* verb planning.

9 Conclusion and Future Work

Creating an effective simultaneous translation system for SOV to SVO languages requires not only translating partial sentences, but also effectively predicting a sentence's verb. Both elements of the system require substantial refinement before they are usable in a real-world system.

Replacing our idealized translation system is the most challenging and most important next step. Supporting multiple translation hypotheses and incremental decoding (Sankaran

et al., 2010) would improve both the efficiency and effectiveness of our system. Using data from human translators (Shimizu et al., 2014) could also add richer strategies for simultaneous translation: passive constructions, reordering, etc.

Verb prediction also can be substantially improved both in its scope in the system and how we predict verbs. Verb-final languages also often place verbs at the end of clauses, and also predicting these verbs would improve simultaneous translation, enabling its effective application to a wider range of sentences. Instead predicting an *exact* verb early (which is very difficult), predicting a semantically close or a more general verb might yield interpretable translations.

A natural next step is expanding this work to other languages, such as Japanese, which not only has SOV word order but also requires tokenization and morphological analysis, perhaps requiring sub-word prediction.

Acknowledgments

We thank the anonymous reviewers, as well as Yusuke Miyao, Naho Orita, Doug Oard, and Sudha Rao for their insightful comments. This work was supported by NSF Grant IIS-1320538. Boyd-Graber is also partially supported by NSF Grant CCF-1018625. Daumé III and He are also partially supported by NSF Grant IIS-0964681. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the International Conference of Machine Learning*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics.

Koby Crammer, Alex Kulesza, and Mark Dredze. 2013. Adaptive regularization of weight vectors. *Machine Learning*, 91(2):155–187.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning Journal (MLJ)*.

Fernanda Ferreira. 2000. Syntax in language production: An approach using tree-adjoining grammars. *Aspects of language production*, pages 291–330.

Christian Fügen. 2008. *A system for simultaneous translation of lectures and speeches*. Ph.D. thesis, KIT-Bibliothek.

Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2013. Simple, lexicalized choice of translation timing for simultaneous speech translation. *INTERSPEECH*.

Francesca Gaiba. 1998. *The origins of simultaneous interpretation: The Nuremberg Trial*. University of Ottawa Press.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the Association for Computational Linguistics*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for n-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. IEEE.

Philipp Koehn. 2000. German-english parallel corpus "de-news".

John Langford and Bianca Zadrozny. 2005. Relating reinforcement learning performance to classification performance. In *Proceedings of the International Conference of Machine Learning*.

Shigeki Matsubara, Keiichi Iwashima, Nobuo Kawaguchi, Katsuhiko Toyama, and Yasuyoshi Inagaki. 2000. Simultaneous Japanese-English interpretation based on early predition of English verb. In *Symposium on Natural Language Processing*.

Hideki Mima, Hitoshi Iida, and Osamu Furuse. 1998a. Simultaneous interpretation utilizing example-based incremental transfer. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 855–861. Association for Computational Linguistics.

Hideki Mima, Hitoshi Iida, and Osamu Furuse. 1998b. Simultaneous interpretation utilizing example-based incremental transfer. In *Proceedings of the Association for Computational Linguistics*.

Shota Momma, Robert Slevc, and Colin Phillips. 2014. The timing of verb selection in english active and passive sentences.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the Association for Computational Linguistics*, June.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics*.

Adam Pauls and Dan Klein. 2011. Faster and smaller n-gram language models. In *Proceedings of the Association for Computational Linguistics*.

Brock Pytlik and David Yarowsky. 2006. Machine translation for languages lacking bitext via multilingual gloss transduction. In *5th Conference of the Association for Machine Translation in the Americas (AMTA)*, August.

Uwe Quasthoff, Matthias Richter, and Christian Biemann. 2006. Corpus portal for search in monolingual corpora. In *International Language Resources and Evaluation*, pages 1799–1802.

Siegfried Ramler and Paul Berry. 2009. *Nuremberg and Beyond: The Memoirs of Siegfried Ramler from 20th Century Europe to Hawai'i*. Booklines Hawaii Limited.

Koichiro Ryu, Shigeki Matsubara, and Yasuyoshi Inagaki. 2006. Simultaneous english-japanese spoken language translation based on incremental dependency parsing and transfer. In *Proceedings of the Association for Computational Linguistics*.

Koichiro Ryu, Shigeki Matsubara, and Yasuyoshi Inagaki. 2012. Alignment-based translation unit for simultaneous japanese-english spoken dialogue translation. In *Innovations in Intelligent Machines–2*, pages 33–44. Springer.

Akiko Sakamoto, Nayuko Watanabe, Satoshi Kamatani, and Kazuo Sumita. 2013. Development of a simultaneous interpretation system for face-to-face services and its evaluation experiment in real situation.

Baskaran Sankaran, Ajeet Grewal, and Anoop Sarkar. 2010. Incremental decoding for phrase-based statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation*.

H Schriefers, E Teruel, and RM Meinshausen. 1998. Producing simple sentences: Results from picture–word interference experiments. *Journal of Memory and Language*, 39(4):609–632.

Hiroaki Shimizu, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Collection of a simultaneous translation corpus for comparative analysis. In *International Language Resources and Evaluation*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*.

Umar Syed, Michael Bowling, and Robert E. Schapire. 2008. Apprenticeship learning using linear programming. In *Proceedings of the International Conference of Machine Learning*.

Christoph Tillmann, Stephan Vogel, Hermann Ney, and Alex Zubiaga. 1997. A dp-based search using monotone alignments in statistical translation. In *Proceedings of the Association for Computational Linguistics*.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 173–180.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*.

Wolfgang Wahlster. 2000. *Verbmobil: foundations of speech-to-speech translation*. Springer.