Learning to Order & Learning to Correct

Learning to Order & Learning to Correct

a dissertation presented
by
Allen Schmaltz
to
The John A. Paulson School of Engineering and Applied Sciences

in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of
Computer Science

Harvard University
Cambridge, Massachusetts
March 2019

Dissertation advisor: Professor Stuart Shieber Allen Schmaltz

Learning to Order & Learning to Correct

Abstract

We investigate three core tasks in Natural Language Processing that are informative toward building tools for writing assistance—word ordering, grammatical error identification, and grammar correction—shedding new light on old questions and providing new modeling approaches for real-world applications. The first task, word ordering, aims to correctly order a randomly shuffled sentence. Via this diagnostic task, we find evidence that strong surface-level models are at least as effective as the models utilizing explicit syntactic structures for modeling ordering constraints, and we incorporate this insight when approaching the end-user grammar tasks.

An advantage of surface-level models is that additional training data is relatively straightforward to acquire. We perform an analysis of word ordering output with a surface-level model at scale. We find that remaining errors are associated with greater proportions of n-grams unseen in training, highlighting both a path for future improvements in effectiveness and the clear brittleness of such models, with implications for generation models, more generally.

The second task, grammatical error identification, seeks to classify whether or not a sentence contains a grammatical error. The third task, grammar correction, seeks

to transduce a sentence that may or may not have errors into a corrected version. For both of these tasks, we utilize insights from the diagnostic word ordering task and adapt modern sequence models to improve effectiveness over contemporary work.

Modern sequence models for the grammar tasks require significant amounts of data to be effective. In a final section, we propose methods for noising well-formed text from limited amounts of human annotated data. With our proposed data augmentation scheme, we demonstrate that sequence models can be trained with synthetic data to approach the levels of effectiveness of models trained on substantially more human annotated sentences. At the same time, such semi-supervised approaches are still clearly weaker than models trained with very large amounts of annotated data.

Contents

For my parents

Biographical Sketch

Allen Schmaltz completed his core higher education in Evanston, Beijing, Taipei, Ithaca, Stanford, Yokohama, and Cambridge in computer science, the social sciences, and the humanities, with language instruction in Japanese, Chinese, and German.

Acknowledgments

Preparing myself for a field (computational public policy) that does not yet exist at Harvard and Stanford (nor elsewhere) has involved a unique set of administrative and logistical challenges. However, I have been fortunate to have been supported by family, friends, colleagues, faculty members, corporations, and foundations in this endeavor. The work leading up to this dissertation especially benefited from discussions with (but certainly not limited to), in no particular order, Yoon Kim, Sam Wiseman, Yuntian Deng, Justin Chiu, Sebastian Gehrmann, Kelly Zhang, Jiawei Zhou, Zachary Ziegler, Lindsey Brown, Elena Llaudet, Yonatan Belinkov, Gary King, Amy Catalinac, Arthur Spirling, Stuart Shieber, Alexander Rush, Barbara Grosz, Wenxin Jiang, Martin Tanner, Hanspeter Pfister, Kevin Knight, Mercè Crosas, Jonathan Yedidia; and my friends and colleagues at Rakuten, Disney Research, ISI, IQSS, the Department of Government, and Harvard CS; and, of course, Anastasia, Chester, Shonda, Rory, and mom and dad.

I look forward to extending my research to the important area of medicine with colleagues at the School of Public Health in the upcoming months.

Preface

It is our long-term hope, in our broader effort of working toward establishing the new research area of computational public policy, that it will eventually become the norm to analyze the world not as one wishes it to be, but as it is.

Allen Schmaltz

Cambridge, 2018

0

Introduction

0.1 Recent Advances

The principal contemporary ideas, topics, and methods of the field of natural language processing (NLP), and artificial intelligence (AI) more generally, have at their origin the scientific and engineering developments of the 1940's, 1950's, and 1960's.[1]

[1] See Appendix A for additional historical context.

In the 1970's through at least the 1990's, the symbolic/logic-based approach to AI, a paradigm championed by John McCarthy and others, was a central—and perhaps dominant—approach to AI. However, in more recent years, in no small part due to advances in hardware, the pendulum has swung back to a set of approaches closer to the earlier connectionist work of Wiener, McCulloch, Rosenblatt, and others (see Appendix A). A common thread in recent work utilizing neural networks, often referred to as deep learning[2], has been to shift engineering away from hand-annotated features and rules to the engineering of network architectures. This pattern has propagated throughout the subfields of AI, including NLP, with perhaps the most dramatic improvements compared to alternative approaches appearing in the subfield of vision.

As a result of recent developments in modeling, advances in hardware, and the availability of data, effectiveness for some tasks has reached the point of having real-world utility. While true understanding of language seems out-of-reach with the current state and trajectory of the field, at least for the foreseeable future, building useful NLP tools for assisting humans in learning languages and writing text seems within the scope of what is possible. It is towards this end that the research of this dissertation is focused.

In approaching the core NLP tasks at hand for this dissertation, we initially take no particular stance on approaches, instead relying solely on empirical results. On

[2] "Deep learning" is somewhat of a misnomer, as such networks are often relatively shallow in terms of the number of layers used.

this empirical basis, in the first part of this dissertation, we find that neural network based approaches are relatively effective for the target tasks compared to alternative approaches in the literature, at least when large amounts of labeled data are available. They can be relatively effective even when learning only from surface-level lexical signals without the introduction of hand-annotated structures or rules. In particular, by distilling language generation down to a representative diagnostic task, we observe the perhaps surprising result that the ordering constraints in natural language appear to be derivable from the surface-level information alone. We use this insight to approach a series of tasks related to grammar correction that have traditionally been approached with hand-annotated structures and features. At the same time, not surprisingly, when there is a very limited amount of annotated data, priors based on human knowledge may be useful. In the final part of this dissertation, we present a task for which introducing a small number of hand-tuned rules may be helpful when there is a very limited amount of available annotated data.

The modeling approaches of this dissertation that make progress toward real-world applications for writing assistance are based on ideas that have a relatively long lineage. Among the more recent modeling developments, this dissertation makes particular use of sequence-to-sequence models (Cho et al., 2014; Sutskever et al., 2014). Interestingly, these recurrent neural network models bear some conceptual resemblance to the encoder/decoder analogies of transduction at the sentence level to communication systems, as proposed by Yngve and others in the 1950's from an information theoretic

perspective (e.g., Yngve, 1956b). More concretely, these sequence-to-sequence models with attention build on the previous work of the Recursive Auto-Associative Memory architecture of Pollack (1990) and variants thereof, as demonstrated (at a simplified, proof-of-concept scale) for use in translation in the 1990's (Chrisman, 1991; Forcada and Ñeco, 1997).

In closing, the past 70 years of work in NLP (and AI, more broadly) has been an empirical endeavor, making real progress on targeted tasks. However, knowledge of how humans process language remains very limited and the distance to true understanding of language via learned functions remains vast, so a certain vigilance is needed to avoid overcommitting to any one particular paradigm or approach. Rigorous empirical examination is needed of models whose behavior may not be fully understood from a theoretical perspective, but that may nonetheless be deployed in real-world situations. In this dissertation, we examine core natural language tasks in light of recent advances, performing careful empirical experiments to create valid comparisons. It is these types of empirical advances that will reliably (and safely) push the field of NLP forward over the next 70 years.

0.2 Summary of Contributions

We revisit three core tasks in NLP that are informative toward building tools for writing assistance—word ordering, grammatical error identification, and grammar

correction—shedding new light on old questions and providing new modeling approaches for real-world applications. While many such tasks have traditionally been approached by modeling explicit syntactic structures, we instead find compelling evidence with a diagnostic task to suggest that such structures are not strictly necessary and incorporate that insight when approaching the end-user grammatical tasks.

In Chapter 1, we reassess the utility of modeling explicit syntactic annotations for ordering constraints. We do so via the task of word ordering, or linearization, which aims to correctly order the words of a randomly shuffled sentence.

Syntactic trees (i.e., graphs of arbitrary but generally defined structures across acceptable language instances, with terminal vertices consisting of lexical items) received attention from the early days of NLP. More recent efforts have resulted in the construction of a moderately-sized corpus of hand-annotated syntactic trees as part of the Penn Treebank Project of the late 1980's and early 1990's (Marcus et al., 1993). Previous work has suggested that directly modeling such trees from the human annotations is essential for modeling ordering constraints. We demonstrate, instead, that strong surface-level language models are at least as effective as models trained with such explicit syntactic structures. This chapter is based on the work of Schmaltz et al. (2016b).

In the follow-up analysis in Chapter 2, we find that at scale, the surface-level language modeling approach for word ordering is reasonably effective, but clearly model errors are still made at the observed scale. We find that remaining errors are associ-

ated with greater proportions of unseen n-grams in training. This suggests that simply increasing the amount of training data may provide a relatively straightforward path for additional gains in effectiveness, but it also highlights that such models are relatively brittle in the presence of data that diverges from that seen in training.

In Chapter 3, we address the task of grammatical error identification, which is a classification task aimed to determine whether or not a sentence contains a grammatical error. We demonstrate approaches for utilizing modern sequence models for the task. Our models lack explicit syntactic features, building on the results from Chapter 1, differing from much of the related contemporary work on grammatical error tasks (e.g., Kudo et al., 1988, as an early, historical example). This line of work was originally validated in a shared task competition, the details of which appear in Schmaltz et al. (2016a).

We explore the related task of grammar correction in Chapter 4. Here, the task is to generate the corrected version of a sentence that may or may not have errors. We propose ways of adapting sequence models for the task, demonstrating improved effectiveness over existing approaches. This work previously appeared in Schmaltz et al. (2017).

The models proposed in Chapters 3 and 4 require significant amounts of labeled data to be effective. In Chapter 5 we propose a data augmentation method for such models, using limited amounts of annotated data to noise well-formed text. Our method is conceptually similar to our work from Chapter 1, now ordering possible errors in a

sentence, instead of ordering the words in a sentence. We demonstrate that sequence models can be trained with synthetic data to approach models trained on significantly more human annotated labels, ceteris paribus.

We close with proposals for future work and directions.

The core chapters of this work suggest that there may be some potential applications for which NLP models are useful as tools to assist humans in learning languages and assist in generating language. However, true understanding of language via learned functions remains a distant (and perhaps, inherently impossible) goal, toward which existing models are not remotely approaching. Nonetheless, the probability of extant NLP models (not restricted to those presented here) engendering systemic, unintended harm with modern hardware in an interconnected world is non-zero. There are further complications when these approaches are applied to fields outside of the sciences, such as in economics and law, that do not necessarily have precedents in analyzing second- and higher-order risks. In Appendix B, we propose some simple steps for researchers to take in order to encourage a pro-active acknowledgement of potential third-party harms. This is an expansion of the published work of Schmaltz (2018).

1

Word Ordering

In this chapter, we reduce the larger problem of natural language generation, which encompasses correction and error identification, down to the simpler, well-defined representative diagnostic task of work ordering. The analysis of our experiments on this task inform the modeling approaches of subsequent chapters.

Recent work on word ordering has argued that modeling the syntactic structure

of hand-annotated trees is important, or even required, for effectively recovering the order of a sentence. We find, instead, that an n-gram language model with a simple future cost heuristic gives strong results on this task. Furthermore, we show that a long short-term memory (LSTM) language model is even more effective at recovering order, with our basic model outperforming a state-of-the-art syntactic model by 11.5 BLEU points, despite only having access to the surface-level lexical items at training.

1.1 Introduction

The task of word ordering seeks to recover the original word order of a shuffled sentence. It is also referred to as bag generation (Brown et al., 1990), shake-and-bake generation (Brew, 1992), or more recently, linearization, as standardized in a recent line of research as a method useful for isolating the performance of text-to-text generation models (Zhang and Clark, 2011; Liu et al., 2015; Liu and Zhang, 2015; Zhang and Clark, 2015). The predominant argument of the more recent works is that jointly recovering explicit syntactic structure is crucial for determining the correct word order of the original sentence. As such, these methods either generate or rely on given parse structure to reproduce the order.

Independently, Elman (1990) explored linearization in his seminal work on recurrent neural networks. Elman judged the capacity of early recurrent neural networks via, in part, the network's ability to predict word order in simple sentences. He notes,

> The order of words in sentences reflects a number of constraints... Syntactic structure, selective restrictions, subcategorization, and discourse considerations are among the many factors which join together to fix the order in which words occur... [T]here is an abstract structure which underlies the surface strings and it is this structure which provides a more insightful basis for understanding the constraints on word order.... It is, therefore, an interesting question to ask whether a network can learn any aspects of that underlying abstract structure (Elman, 1990).

Recently, recurrent neural networks have reemerged as a powerful tool for learning the latent structure of language. In particular, work on long short-term memory (LSTM) networks for language modeling has yielded improvements in perplexity.

We revisit Elman's question by applying LSTMs to the word ordering task, without any explicit syntactic modeling. We find that language models are in general effective for linearization relative to existing syntactic approaches, with LSTMs in particular outperforming the state-of-the-art by 11.5 BLEU points, with further gains observed when training with additional text and decoding with larger beams.

1.2 Background: Linearization

The task of linearization is to recover the original order of a shuffled sentence. We assume a vocabulary $\mathcal{V}$ and are given a sequence of out-of-order phrases $x_1, \ldots, x_N$, with $x_n \in \mathcal{V}^+$ for $1 \leq n \leq N$. Define M as the total number of tokens (i.e., the sum of the lengths of the phrases). We consider two varieties of the task: (1) Words, where each x_n consists of a single word and $M = N$, and (2) Words+BNPs, where base noun phrases (noun phrases not containing inner noun phrases) are also provided and

$M \geq N$. The second has become a standard formulation in recent literature.

Given input x, we define the output set $\mathcal{Y}$ to be all possible permutations over the N elements of x, where $\hat{y} \in \mathcal{Y}$ is the permutation generating the true order. We aim to find $\hat{y}$, or a permutation close to it. We produce a linearization by (approximately) optimizing a learned scoring function f over the set of permutations, $y^* = \arg\max_{y \in \mathcal{Y}} f(x, y)$.

1.3 Related Work: Syntactic Linearization

Recent approaches to linearization have been based on reconstructing the syntactic structure to produce the word order. Let $\mathcal{Z}$ represent all projective dependency parse trees over M words. The objective is to find $y^*, z^* = \arg\max_{y \in \mathcal{Y}, z \in \mathcal{Z}} f(x, y, z)$ where f is now over both the syntactic structure and the linearization. The current state of the art on the Penn Treebank (PTB) (Marcus et al., 1993), without external data, of Liu et al. (2015) uses a transition-based parser with beam search to construct a sentence and a parse tree. The scoring function is a linear model $f(x, y) = \theta^\top \Phi(x, y, z)$ and is trained with an early update structured perceptron to match both a given order and syntactic tree. The feature function Φ includes features on the syntactic tree. This work improves upon past work which used best-first search over a similar objective (Zhang and Clark, 2011).

In follow-up work, Liu and Zhang (2015) argue that syntactic models yield im-

provements over pure surface n-gram models for the WORDS+BNPS case. This result holds particularly on longer sentences and even when the syntactic trees used in training are of low quality. The n-gram decoder of this work utilizes a single beam, discarding the probabilities of internal, non-boundary words in the BNPs when comparing hypotheses. We revisit this comparison between syntactic models and surface-level models, utilizing a surface-level decoder with heuristic future costs and an alternative approach for scoring partial hypotheses for the WORDS+BNPS case.

Additional previous work has also explored n-gram models for the word ordering task. The work of de Gispert et al. (2014) demonstrates improvements over the earlier syntactic model of Zhang et al. (2012) by applying an n-gram language model over the space of word permutations restricted to concatenations of phrases seen in a large corpus. Horvat and Byrne (2014) models the search for the highest probability permutation of words under an n-gram model as a Travelling Salesman Problem; however, direct comparisons to existing works are not provided.

1.4 LM-BASED LINEARIZATION

In contrast to the recent syntax-based approaches, we use an LM directly for word ordering. We consider two types of language models: an n-gram model and a long short-term memory network (Hochreiter and Schmidhuber, 1997). For the purpose of this work, we define a common abstraction for both models. Let $\boldsymbol{h} \in \mathcal{H}$ be the current

Algorithm 1 LM beam-search word ordering

procedure ORDER($x_1 \dots x_N$, K, g)
  $B_0 \leftarrow \langle(\langle\rangle, \{1, \dots, N\}, 0, \boldsymbol{h}_0)\rangle$
  for $m = 0, \dots, M-1$ **do**
    for $k = 1, \dots, |B_m|$ **do**
      $(y, \mathcal{R}, s, \boldsymbol{h}) \leftarrow B_m^{(k)}$
      for $i \in \mathcal{R}$ **do**
        $(s', \boldsymbol{h}') \leftarrow (s, \boldsymbol{h})$
        for word w in phrase x_i **do**
          $s' \leftarrow s' + \log q(w, \boldsymbol{h}')$
          $\boldsymbol{h}' \leftarrow \delta(w, \boldsymbol{h}')$
        $j \leftarrow m + |x_i|$
        $B_j \leftarrow B_j + (y + x_i, \mathcal{R} - i, s', \boldsymbol{h}')$
        keep top-K of B_j by $f(x, y) + g(\mathcal{R})$
  return B_M

state of the model, with $\boldsymbol{h}_0$ as the initial state. Upon seeing a word $w_i \in \mathcal{V}$, the LM advances to a new state $\boldsymbol{h}_i = \delta(w_i, \boldsymbol{h}_{i-1})$. At any time, the LM can be queried to produce an estimate of the probability of the next word $q(w_i, \boldsymbol{h}_{i-1}) \approx p(w_i \mid w_1, \dots, w_{i-1})$. For n-gram language models, $\mathcal{H}$, δ, and q can naturally be defined respectively as the state space, transition model, and edge costs of a finite-state machine.

LSTMs are a type of recurrent neural network (RNN) that are conducive to learning long-distance dependencies through the use of an internal memory cell. Existing work with LSTMs has generated state-of-the-art results in language modeling (Zaremba et al., 2014), along with a variety of other NLP tasks.

In our notation we define $\mathcal{H}$ as the hidden states and cell states of a multi-layer LSTM, δ as the LSTM update function, and q as a final affine transformation and softmax given as $q(*, \boldsymbol{h}_{i-1}; \theta_q) = \text{softmax}(\boldsymbol{W}\boldsymbol{h}_{i-1}^{(L)} + \boldsymbol{b})$ where $\boldsymbol{h}_{i-1}^{(L)}$ is the top hidden

layer and $\theta_q = (\boldsymbol{W}, \boldsymbol{b})$ are parameters. We direct readers to the work of Graves (2013) for a full description of the LSTM update.

For both models, we simply define the scoring function as

$$f(x, y) = \sum_{n=1}^{N} \log p(x_{y(n)} \mid x_{y(1)}, \ldots, x_{y(n-1)})$$

where the phrase probabilities are calculated word-by-word by our language model.

Searching over all permutations $\mathcal{Y}$ is intractable, so we instead follow past work on linearization (Liu et al., 2015) and LSTM generation (Sutskever et al., 2014) in adapting beam search for our generation step. Our work differs from the beam search approach for the WORDS+BNPS case of previous work in that we maintain multiple beams, as in stack decoding for phrase-based machine translation (Koehn, 2010), allowing us to incorporate the probabilities of internal, non-boundary words in the BNPs. Additionally, for both WORDS and WORDS+BNPS, we also include an estimate of future cost in order to improve search accuracy.

Beam search maintains $M + 1$ beams, $B_0, \ldots, B_M$, each containing at most the top-K partial hypotheses of that length. A partial hypothesis is a 4-tuple $(y, \mathcal{R}, s, \boldsymbol{h})$, where y is a partial ordering, $\mathcal{R}$ is the set of remaining indices to be ordered, s is the score of the partial linearization $f(x, y)$, and $\boldsymbol{h}$ is the current LM state. Each step consists of expanding all next possible phrases and adding the next hypothesis to a later beam. The full beam search is given in Algorithm 1.

Model	WORDS	WORDS+BNPS
ZGEN-64	30.9	49.4
ZGEN-64+POS	–	50.8
NGRAM-64 (NO g)	32.0	51.3
NGRAM-64	37.0	54.3
NGRAM-512	38.6	55.6
LSTM-64	40.5	60.9
LSTM-512	42.7	63.2
ZGEN-64+LM+GW+POS	–	52.4
LSTM-64+GW	41.1	63.1
LSTM-512+GW	44.5	65.8

Table 1.1: BLEU score comparison on the PTB test set. Results from previous works (for ZGEN) are those provided by the respective authors, except for the WORDS task. The final number in the model identifier is the beam size, +GW indicates additional Gigaword data. Models marked with +POS are provided with a POS dictionary derived from the PTB training set.

As part of the beam search scoring function we also include a future cost g, an estimate of the score contribution of the remaining elements in $\mathcal{R}$. Together, $f(x, y) + g(\mathcal{R})$ gives a noisy estimate of the total score, which is used to determine the K best elements in the beam. In our experiments we use a very simple unigram future cost estimate, $g(\mathcal{R}) = \sum_{i \in \mathcal{R}} \sum_{w \in x_i} \log p(w)$.

1.5 EXPERIMENTS

Experiments are on PTB with sections 2-21 as training, 22 as validation, and 23 as test[1]. We utilize two UNK types, one for initial upper-case tokens and one for all other low-frequency tokens; end sentence tokens; and start/end tokens, which are

[1]In practice, the results in Liu et al. (2015) and Liu and Zhang (2015) use section 0 instead of 22 for validation (author correspondence).

BNP	g	GW	1	10	64	128	256	512
					LSTM			
•			41.7	53.6	58.0	59.1	60.0	60.6
•	•		47.6	59.4	62.2	62.9	63.6	64.3
•	•	•	48.4	60.1	64.2	64.9	65.6	66.2
			15.4	26.8	33.8	35.3	36.5	38.0
	•		25.0	36.8	40.7	41.7	42.0	42.5
	•	•	23.8	35.5	40.7	41.7	42.9	43.7
					NGRAM			
•			40.6	49.7	52.6	53.2	54.0	54.7
•	•		45.7	53.6	55.6	56.2	56.6	56.6
			14.6	27.1	32.6	33.8	35.1	35.8
	•		27.1	34.6	37.5	38.1	38.4	38.7

Table 1.2: BLEU results on the validation set for the LSTM and NGRAM model with varying beam sizes, future costs, additional data, and use of base noun phrases.

treated as words, to mark BNPs for the WORDS+BNPS task. We also use a special symbol to replace tokens that contain at least one numeric character. We otherwise train with punctuation and the original case of each token, resulting in a vocabulary containing around 16K types from around 1M training tokens.

For experiments marked GW we augment the PTB with a subset of the Annotated Gigaword corpus (Napoles et al., 2012). We follow Liu and Zhang (2015) and train on a sample of 900k Agence France-Presse sentences combined with the full PTB training set. The GW models benefit from both additional data and a larger vocabulary of around 25K types, which reduces unknowns in the validation and test sets.

We compare the models of Liu et al. (2015) (known as ZGEN), a 5-gram LM using modified Kneser-Ney smoothing (NGRAM)[2], and an LSTM. We experiment on the WORDS and WORDS+BNPS tasks, and we also experiment with including future

[2]We use the KenLM Language Model Toolkit (`https://kheafield.com/code/kenlm/`).

costs (g), the Gigaword data (GW), and varying beam size. We retrain ZGEN using publicly available code[3] to replicate published results.

The LSTM model is similar in size and architecture to the medium LSTM setup of Zaremba et al. (2014)[4]. Our implementation uses the Torch[5] framework and is publicly available[6].

We compare the performance of the models using the BLEU metric (Papineni et al., 2002). In generation if there are multiple tokens of identical UNK type, we randomly replace each with possible unused tokens in the original source before calculating BLEU. For comparison purposes, we use the BLEU script distributed with the publicly available ZGEN code.

1.6 RESULTS

Our main results are shown in Table 1.1. On the WORDS+BNPS task the NGRAM-64 model scores nearly 5 points higher than the syntax-based model ZGEN-64. The LSTM-64 then surpasses NGRAM-64 by more than 5 BLEU points. Differences on the WORDS task are smaller, but show a similar pattern. Incorporating Gigaword data further increases the result by another 2 points. Notably, the NGRAM model outperforms the combined result of ZGEN-64+LM+GW+POS from Liu and Zhang

[3] `https://github.com/SUTDNLP/ZGen`

[4] We hypothesize that additional gains are possible via a larger model and model averaging, ceteris paribus.

[5] `http://torch.ch`

[6] `https://github.com/allenschmaltz/word_ordering`

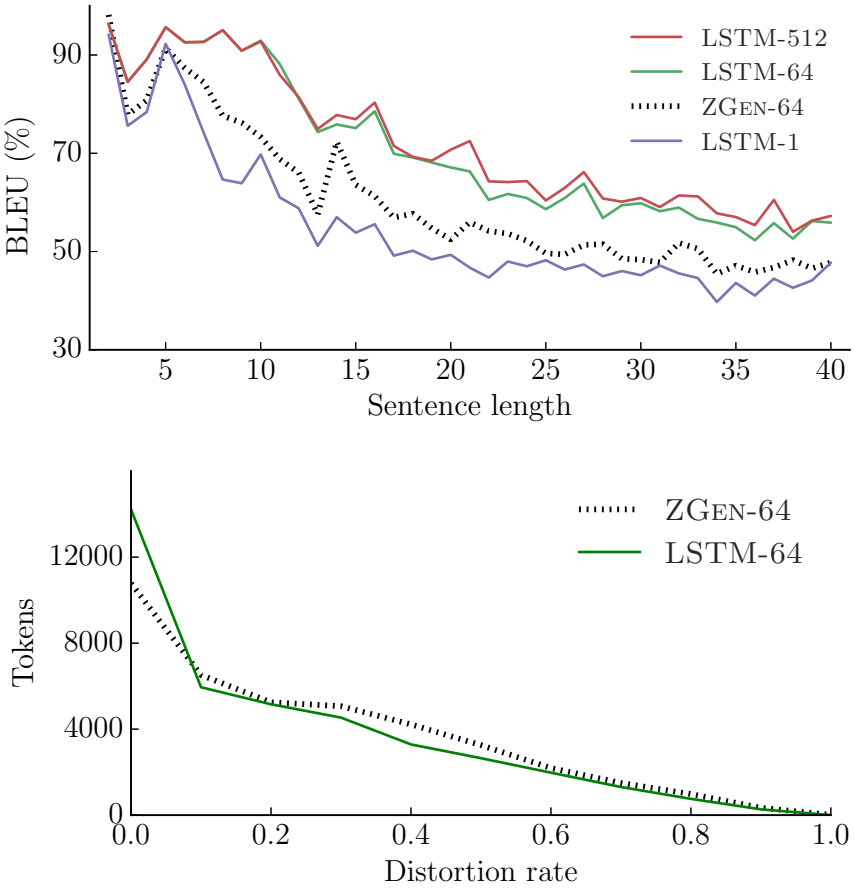

Figure 1.1: Experiments on the PTB validation on the Words+BNPs models: (a) Performance on the set by length on sentences from length 2 to length 40. (b) The distribution of token distortion, binned at 0.1 intervals.

(2015), which uses a 4-gram model trained on Gigaword. We believe this is because the combined ZGen model incorporates the n-gram scores as discretized indicator features instead of using the probability directly.[7] A beam of 512 yields a further improvement at the cost of search time.

[7]In work of Liu and Zhang (2015), with the given decoder, N-grams only yielded a small further improvement over the syntactic models when discretized versions of the LM probabilities were incorporated as indicator features in the syntactic models.

Model	WORDS	WORDS+BNPS
ZGEN-64(z^*)	39.7	64.9
ZGEN-64	40.8	65.2
NGRAM-64	46.1	67.0
NGRAM-512	47.2	67.8
LSTM-64	51.3	71.9
LSTM-512	52.8	73.1

Table 1.3: Unlabeled attachment scores (UAS) on the PTB validation set after parsing and aligning the output. For ZGEN we also include a result using the tree z^* produced directly by the system. For WORDS+BNPS, internal BNP arcs are always counted as correct.

To further explore the impact of search accuracy, Table 1.2 shows the results of various models with beam widths ranging from 1 (greedy search) to 512, and also with and without future costs g. We see that for the better models there is a steady increase in accuracy even with large beams, indicating that search errors are made even with relatively large beams.

One proposed advantage of syntax in linearization models is that it can better capture long-distance relationships. Figure 1.1 shows results by sentence length and distortion, which is defined as the absolute difference between a token's index position in y^* and $\hat{y}$, normalized by M. The LSTM model exhibits consistently better performance than existing syntax models across sentence lengths and generates fewer long-range distortions than the ZGEN model.

Finally, Table 1.3 compares the syntactic fluency of the output. As a lightweight test, we parse the output with the Yara Parser (Rasooli and Tetreault, 2015) and compare the unlabeled attachment scores (UAS) to the trees produced by the syn-

tactic system. We first align the gold (i.e., true, hand-labeled) head to each output token. (In cases where the alignment is not one-to-one, we randomly sample among the possibilities.) The models with no knowledge of syntax are able to recover a higher proportion of gold arcs.

1.7 Conclusion: Word Ordering without Hand-Annotated Syntactic Trees

In this chapter, we have demonstrated that strong surface-level language models recover word order more accurately than the models trained with explicit syntactic annotations appearing in a recent series of papers. Our models in subsequent chapters on grammatical error tasks are informed by this result, and our experiments have implications for the utility of costly syntactic annotations in generation models, more generally, for both high- and low- resource languages and domains.

Acknowledgments

In preparing Schmaltz et al. (2016b), on which this chapter is based, we thank Yue Zhang and Jiangming Liu for assistance in using ZGen, as well as verification of the task setup for a valid comparison. Jiangming Liu also assisted in pointing out a discrepancy in the implementation of an earlier version of our NGram decoder, the resolution of which improved BLEU performance.

2

Word Ordering at Scale

In the previous chapter, we demonstrated that there appears to be no additional word ordering signal derivable from hand-annotated syntactic trees beyond that available from surface-level information, as for example, via smoothed n-gram language models. Furthermore, we observed additional improvements in word ordering via lower

perplexity LSTM language models.[1]

The experiments in Chapter 1 were designed to carefully control various characteristics to be comparable to previous works modeling syntactic information. However, lower perplexity surface-level language models are known to exist by, among other methods, increasing model size and the amount of training data. Additionally, the results in Chapter 1 would seem to suggest that even larger beam sizes might improve ordering effectiveness by reducing search errors.

In this chapter, we examine word ordering via surface-level models toward the practical limits of our available computing resources. We perform a human evaluation on the output with the aim of identifying the remaining ordering errors made by contemporary large-scale surface-level models and to provide guidance for future lines of research, with implications for generation models, more generally.

We find that at the scale observed here with the large-scale language model of Dauphin et al. (2017), the basic language modeling approach for word ordering described in Chapter 1 is reasonably effective in human evaluation terms (roughly 70-76 percent of sentences are grammatically and semantically acceptable, depending on the dataset) when accounting for search errors (which are largely attributable to computing resources rather than strictly model deficiencies). Interestingly, the remaining

[1]In unpublished experiments from 2015, we also explored modeling syntactic information with an LSTM language model, by incorporating shift-reduce actions (and slight variants thereof, such as delaying certain actions to allow a type of "look-ahead" for certain lexical items in the stack), trained with annotated dependency trees derived from the Penn Treebank constituency trees, into the language model and using such models for ordering. We observed no gains over otherwise comparable surface-level LSTMs without such syntactic information.

errors are associated with higher proportions of unseen n-grams in the training data, suggesting that future gains may be possible by simply further increasing the amount of training data (and concomitantly, the model size).

Such surface-level models of course have no access to general world knowledge beyond that which is available in the sequential training data[2], and thus, they are still relatively brittle in the face of data that diverges from that seen in training. However, the results in Chapter 1 and this chapter provide evidence to suggest that explicit, hand-annotated syntactic information is not strictly necessary for modeling ordering constraints for grammatical acceptability when large-scale surface-level information is available. In other words, the ordering information available in the trees appears to be derivable from the surface-level information alone.

The associated replication code, output data, and human evaluations presented in this chapter are available at `https://github.com/allenschmaltz/word_ordering_at_scale`.

2.1 Introduction

The experiments of Chapter 1, and follow-up work such as that of Hasler et al. (2017), suggest that other things being equal, surface-level models appear to be at least as effective at modeling ordering constraints as models that utilize hand-annotated syn-

[2]As such, we would assume that even at a scale larger than that considered here, examples in the spirit of Winograd Schemas (Levesque et al., 2012) could be constructed to fool such a model by making use of information outside the set of training sentences.

tactic structures.[3]

At the same time, it is also apparent that such models still make ordering errors when learning from modest-sized training datasets. The number of existing hand-annotated trees is relatively small, but additional training data for language models (i.e., existing well-formed text) is relatively straightforward to acquire. In this chapter, we revisit the basic word ordering setup from Chapter 1 with a significantly larger model, training set, and beam size in order to perform an analysis of the remaining errors made by a particular surface-level model, with implications for future work more generally.

2.2 Model

For purposes of computational efficiency[4], we use the gated convolutional network approach of Dauphin et al. (2017) as our surface-level language model. We use the publicly available pre-trained language model[5], which is similar to the large GCNN-

[3] An important (and hopefully, obvious) distinction is that our results *do not* suggest that language itself is somehow non-hierarchical or non-recursive, but rather that the observed surface-level models are able to capture the ordering constraints without the hand-annotated syntactic trees.

[4] Preliminary experiments with the pre-trained LSTM model of Jozefowicz et al. (2016), available at `https://github.com/tensorflow/models/tree/master/research/lm_1b`, suggested that performing inference with the model would be impractically slow in our academic computing environment. As such, it was not pursued further. That model has a reported single-model perplexity that is similar (within about 2 perplexity points better/lower, albeit with additional computing resources) than the convolutional network language model used for the experiments of this chapter, although even lower perplexities were reported by averaging several models together in an ensemble. Our current computing resources preclude experimentation with ensembles at this scale.

[5] The model is available at `https://github.com/pytorch/fairseq`.

14 BOTTLENECK model of Dauphin et al. (2017). The original model was reportedly trained for 2 weeks on 8 Tesla M40 GPUs, so here we rely on the large, pre-trained model rather than attempting to train the model in our own more constrained academic computing environment.

The model was trained on the Google Billion Words benchmark (GBW) training dataset (Chelba et al., 2013). This dataset contains about 30 million sentences, which is more than 30 times the size of the largest training set from Chapter 1. For training, words appearing fewer than 3 times were replaced with a single UNK type. The model has 796,771,584 parameters and a vocabulary of 793,302. Case information is retained in the tokens.[6]

For the purposes of this chapter, the differences in the architectures of the LSTM language model of Chapter 1 and the model of Dauphin et al. (2017) are primarily only of interest here in that the latter model enables comparatively efficient inference with a large surface-level model over a large vocabulary. We leave to future work the analysis of the significance of the particular network architecture on ordering effectiveness at this scale, setting aside efficiency. However, for reference, we briefly review those aspects of the architecture that make comparatively fast inference possible.

[6]Anecdotal evidence suggests that case provides a useful signal for ordering English with surface-level models, even at scale. Initial experiments with the pre-trained model of Radford et al. (2018) yielded relatively weak results, despite the model being trained on a large corpus and having a relatively low perplexity. A confounder appeared to be the lack of explicit case information accessible by the model, in that the bytepair encoded (Sennrich et al., 2016) vocabulary of the model was lowercased, effectively eliminating some surface-level information in the tokens.

In particular, the gated convolutional network language model of Dauphin et al. (2017) is able to achieve similar perplexities as LSTM models using fewer resources, as the non-recurrent gated convolutions of the model are conducive to parallel processing of the time-dependent tokens. Although the convolutions over the input tokens have a finite context, additional effective context (along with more abstractive features) is obtained by stacking multiple convolutional networks. This has been shown to work competitively well in practice for language modeling relative to LSTMs, which in principle have infinite context, but typically also have, in practice, finite contexts due to the gradient truncations that occur in standard training regimes. However, for beam-search inference, while LSTMs do have a time-dependency, the time-dependent model state can be cached in a straightforward way (at the expense of memory), so we leave to future work the particular efficiency tradeoffs when using such non-recurrent models with large beam sizes, other factors being equal.

The LSTM language model in Chapter 1 obtains a distribution over the vocabulary by performing a softmax over the full vocabulary at each timestep. This is feasible in practice, because the vocabulary is much smaller than that examined here. The much larger LSTM model of Jozefowicz et al. (2016) performs the full softmax computation at inference time over the approximately 800,000-word vocabulary of the GBW dataset, which contributes to relatively slow inference (see Footnote 4).

In contrast, the model of Dauphin et al. (2017) reduces the cost of model predictions incurred by the softmax over the full vocabulary by instead using the adaptive

softmax approach of Grave et al. (2016).[7] A shallow, two-level tree is constructed with the most frequent words and the cluster representations for the remaining words in a root node, with the remaining words in leaf clusters. The probability of a frequently occurring word can be obtained by calculating the probability over the cluster in the root node, which has a much smaller cardinality than the full vocabulary, whereas the probability of a less frequent word can be obtained by multiplying the probability of the associated cluster representation in the root node with the probability of the word in its associated leaf cluster. Additionally, a projection is taken to reduce the model capacity of the clusters associated with the rare words, given the observation that more frequent words (as, for example, in the root node) benefit from higher capacity for adequate modeling, whereas such capacity is wasted on rare words and can be reduced without having an adverse impact on effectiveness.

2.3 Experiments

2.3.1 Task

The basic setup of the word ordering, or linearization, task follows that which was examined in Chapter 1. However, the experiments using base noun phrases in Chapter 1 were presented to provide comparable results to previous works. In practice in real-

[7] An alternative, orthogonal approach for reducing the softmax computation would be to reduce the size of the vocabulary by moving to bytepair encoding or character-level encoding, at the expense of longer sequences. We leave such comparisons to future work.

world settings, such annotations for base noun phrases are not generally available for training and inference, so here we only focus on the WORDS setup.

2.3.2 DATA

Since the examined pre-trained model was trained on the GBW dataset, we analyze the standard test set for that dataset in addition to the Penn Treebank (PTB) data from Chapter 1. Due to the high computational costs at the examined scale, re-ordering large numbers of sentences is prohibitive, so we instead sample smaller sets to re-order.

More specifically, we randomly sample 100 sentences from the validation set of the PTB dataset from Chapter 1, and we randomly sample 100 sentences from the standard test set split of the GBW dataset. We then randomly shuffle the words in each sentence, which becomes the input for the model. The PTB dataset from Chapter 1 has a slightly different tokenization format than the GBW dataset on which the model was trained (e.g., parentheses are not replaced by special symbols), so we convert the tokenization of the PTB dataset to that used by the model and retain that format for subsequent analyses.

2.3.3 PARAMATERS

We create a unigram language model for future costs, analogous to those used in Chapter 1, using the full Google Billion Words benchmark training dataset, resulting

in a total vocabulary of 2,425,340.

We use a beam size of 20,000 for beam search. We chose this size as a reasonable balance for minimizing search errors and the computational costs of our academic computing environment. The final step in the beam search includes the score of the final end-of-sentence token as a lightweight constraint that the task is to construct a single full sentence (as opposed to, for example, not discouraging the construction of a valid shorter sentence followed by a valid prefix of a subsequent sentence).

2.3.4 Human Evaluation

After re-ordering the fully shuffled sentences, we analyze the output. We mark whether or not the original source sentence is in fact a full sentence (as opposed to, for example, a headline). For each original sentence and its re-ordered counterpart, we check for grammatical and semantic acceptability. We also mark whether or not the original sentence and its re-ordered counterpart have the same meaning. The evaluations at a semantic level are, of course, subjective, given the limited context of a single sentence, which we address further in the analysis section below. Our basic standard for acceptability is whether or not a sentence would require re-writing to appear in a similar news article domain (not unlike the proofreading and copywriting standards for grammatical corrections for particular domains, such as scientific articles and student essays, appearing in subsequent chapters).

2.4 Results

2.4.1 Data

For reference and context, we calculate the perplexity of the model on the source PTB and GBW sentences. Perplexity is the following transformation of the average log probability of held-out data (i.e., non-training data) often used as an evaluation measure for language models:

$$e^{\frac{1}{N}\sum_{i=1}^{N} -\ln p(x_i|x_1,\ldots,x_{i-1})}$$

where each x_i is one of N tokens in the data, including a special end of sentence token appearing at the end of every sentence. (A start of sentence token is also given as input to the model, but it is not scored as part of the perplexity calculation.) We follow the practice of treating each sentence as an independent sequence (i.e., model state is not shared across sentence boundaries).[8]

Using the model vocabulary and tokenization and the aforementioned specification, the perplexity of the 100 PTB sentences is about 63.5, and the perplexity of the 100

[8]The choice of vocabulary and tokenization, whether or not to share state across sentence boundaries, and the distributional similarity of the evaluation text to the training text can all have a significant impact on the perplexity and must be held constant for a valid comparison across models. The model of Dauphin et al. (2017) was externally validated in the original paper by comparisons on the Google Billion Words benchmark. In providing perplexity measures here, we are interested in the relative difference between that of the PTB and GBW sentences to illustrate the relative inference challenge of the two datasets for the model.

GBW sentences is about 31.0. (To reiterate, note that these are perplexities over the original, non-shuffled source sentences.) The higher perplexity on the PTB sentences indicates greater model uncertainty over these sentences relative to that of the GBW sentences. Although both datasets are in the news domain, the PTB sentences contain more tokens outside the model vocabulary, and qualitatively, the PTB sentences have a greater focus on financial-themed phrases than the GBW sentences. As such, the PTB sentences may diverge slightly farther from the distribution of the original GBW training sentences. (Recall that the corresponding PTB training sentences were not used to train the model.)

2.4.2 Search

The search space for ordering a bag, or multiset, of words is large.[9] The number of possible permutations is given by the multinomial coefficient, where the denominator accounts for repeated tokens. (Without repetitions, the number of permutations is simply a factorial of the number of tokens in the sentence.)

For example, let us consider the final sentence in Table 2.5, for which the model generates an acceptable ordering. The bag of words given to the model is {`'after', 'me', '.', 'ought', 'free', 'to', 'told', 'then', 'I', 'saying', 'I',`

[9]That humans can successfully search this space in reasonably short time spans, as in language learning games such as `http://reorder.coffee/`, which was created by the author of this work, remains a curiosity and area for future research. However, at this stage, too little is known about the brain and human processing to make any meaningful comparisons on the matter with the statistical approaches examined here.

Dataset	Average Length	BLEU
PTB	23.2	55.1
GBW	25.4	61.8

Table 2.1: Summary statistics and automatic evaluations of the re-ordered (after shuffling) versions of all 100 PTB sentences and all 100 GBW sentences.

`'more', 'to', 'go', 'He', 'careful', 'be', 'was'`}. There are 18 tokens and 16 types (`'I'` and `'to'` are repeated). Thus, the number of possible sequences is

$$\binom{18}{1,1,1,1,1,2,1,1,2,1,1,1,1,1,1,1} = \frac{18!}{1!1!1!1!1!2!1!1!2!1!1!1!1!1!1!1} \approx 1.6 \times 10^{15}.$$

In other words, the search space is very large, even for relatively short sentences.

In our analysis, we are particularly interested in analyzing model errors, and as such, we are primarily interested in re-ordered sentences that do not exhibit obvious search errors, which in principle could be lessened by increasing the beam size. While it is difficult, in general, to determine whether the global maximum probability ordering under the model has been found, we can at least check whether a proposed ordering scores greater than or equal to the original source sentence. We refer to re-ordered sequences that have probabilities less than the original source sentence as exhibiting "source-relative" (S.R.) search error.

Dataset	Average Length	Proportion Identical to Source	Proportion Acceptable	Proportion w/o S.-R. Search Error	Number of Sentences
PTB	23.7	0.126	0.589	0.737	95
GBW	25.4	0.266	0.596	0.660	94

Table 2.2: Summary statistics and human evaluations of the subset of the re-ordered (after shuffling) versions of PTB sentences and GBW sentences for which the original source sentence was evaluated as being a regular sentence (e.g., not a headline) that is grammatically and semantically acceptable. The total number of such source sentences among the 100 PTB sentences is 95, and among the 100 GBW sentences is 94. Source-relative (S.-R.) search error is defined in the main text. The proportion acceptable is the proportion of re-ordered sentences evaluated as being grammatically and semantically acceptable.

2.4.3 Acceptability

For context, we run the same BLEU script from Chapter 1 to automatically evaluate the sentences. In Table 2.1, we see that the overall BLEU scores for the GBW sentences tends to be higher than that of the PTB sentences, which is an outcome that tracks the perplexity values noted above. However, for the purposes of this chapter, we are primarily interested in the human evaluations, to which we now turn.

As shown in Table 2.2, among the source sentences, we find that 95 of the 100 PTB sentences are full sentences that are acceptable in terms of grammar and meaning, and 94 of the GBW sentences are full sentences that are acceptable in terms of grammar and meaning. The remaining excluded sentences tend to be headlines, or otherwise incomplete sentences. We restrict our subsequent analyses to the re-ordered sentences that correspond to the acceptable source sentences.

Of the 95 acceptable source PTB sentences, about 73.7 percent do not exhibit

Among Re-ordered Sentences w/o S.-R. Search Error				
Dataset	Average Length	Proportion Identical to Source	Proportion Acceptable	Number of Sentences
PTB	21.1	0.171	0.7	70
GBW	20.8	0.403	0.758	62

Table 2.3: Summary statistics and human evaluations of the subset of the re-ordered (after shuffling) versions of PTB sentences and GBW sentences for which the original source sentence was evaluated as being a regular sentence (e.g., not a headline) that is grammatically and semantically acceptable AND for which the log-probability of the re-ordered sentence is greater than or equal to that of the original source (i.e., sentences without source-relative, S.-R., search error). The total number of such source sentences among the 100 PTB sentences is 70, and among the 100 GBW sentences is 62. The proportion acceptable is the proportion of re-ordered sentences evaluated as being grammatically and semantically acceptable.

source-relative search error, and of the 94 acceptable source GBW sentences, about 66.0 percent do not exhibit source-relative search error. We subsequently only consider the subset of re-ordered sentences without source-relative search error for which the corresponding source sentence is acceptable. As shown in Table 2.3, among the PTB subset, we find that 70 percent are acceptable, with about 17 percent exactly matching the original source. Among the GBW subset, we find that about 76 percent are acceptable, with about 40 percent exactly matching the original source.

Table 2.4 includes examples of sentences that were re-ordered by the model and marked as problematic. The first contains the phrase "arranging peace tragic results", which is grammatically questionable. In the second example, we see that the model has scrambled the full name of the firm. The third example illustrates our standard of marking re-ordered sentences as not acceptable if they would need to be re-written in the given news domain. In this case, while there could conceivably be a market

Examples of Problematic Re-ordered Sentences w/o S.-R. Search Error	
O	The U.S. is in the habit of arranging peace settlements and then washing its hands over the tragic results .
R	The U.S. is in the habit of washing its hands over the settlements and then arranging peace tragic results .
O	Here are price trends on the world 's major stock markets , as calculated by Morgan Stanley Capital International Perspective , Geneva .
R	Here are the world 's major stock markets as calculated by Geneva Capital , Morgan Stanley , International Perspective on price trends .
O	But in a meeting after the market closed yesterday , Shearson executives decided not to go ahead with the stock-market ad .
R	But in a meeting after the stock-market market closed yesterday , Shearson executives decided not to go ahead with the ad .
O	John Connolly , whose two-year stint as head coach ended with the defeat , was confident Australian rugby was on the right track and the Wallabies would remain a force in years to come .
R	Australian head coach John Connolly , whose two-year stint with the Wallabies ended in defeat , was confident rugby was on the right track and would remain a force as the years to come .
O	The statement says both missiles launched and flew without trouble but the system 's sea-based X-band radar did not perform as expected and the interceptor missed its target .
R	The system 's sea-based X-band radar and its interceptor missiles did not perform as expected but both flew without trouble and missed the target the launched statement says .
O	Queen Elizabeth is formally queen of Australia despite the country 's independence from Britain and the governor general acts as the monarch 's representative in Australia .
R	Queen Elizabeth is the queen 's representative in Australia and the monarch formally acts as governor of Australia despite the country 's general independence from Britain .

Table 2.4: The first three examples are from PTB and the final three are from GBW, for the original sentence (O) and the re-ordered counterpart (R). These re-ordered examples are grammatically and/or semantically problematic, even though they do not exhibit source-relative search error. Primary errors are marked in red.

for stock-markets (for example, as part of an article introducing a new concept), in which case the sentence would otherwise be grammatically and semantically acceptable, we marked the sentence as not acceptable because it is sufficiently unlikely that such a sentence would appear in a *Wall Street Journal* news article. This is an inherently subjective decision, but this subjectivity does not impact our core conclusions with respect to the hand-annotated trees. When in doubt, we aim to err on the side of marking a sentence as not acceptable, as in this case.

The fourth example makes a straightforward preposition mistake, and the fifth sentence is awkward in general, with the phrase "launched statement" sufficiently unusual to mark the sentence as not acceptable. The final sentence is another example in which given sufficient context, the sentence might be considered acceptable. However, we mark the sentence as not semantically acceptable, as Queen Elizabeth is not her own representative in Australia, and she is not the governor of Australia.

2.4.4 Semantic Comparisons between Acceptable Sentences

There is no particular reason to assume that such a model, when given a bag of words without additional context, would be able to consistently reconstruct a sentence with the same meaning as the original. A human would also not be expected to be able to do so (although, of course, in some cases, one meaning might be more likely to be chosen than others). That is, in fact, the case observed in our human evaluation of the output, with many sentences otherwise marked as grammatically and semantically

		Examples of Acceptable Re-ordered Sentences w/o S.-R. Search Error
O	I	Several analysts said Malaysia and Singapore had the biggest losses because they are relatively open to rapid cash flows .
R	I	Several analysts said Singapore and Malaysia had the biggest losses because they are relatively open to rapid cash flows .
O	S	" These increases were partly offset by lower trading-related " income , the bank said .
R	S	" These increases were partly offset by lower trading-related income , " the bank said .
O	D	They said the exchange 's trading procedures , personnel , equipment and links with other exchanges couldn 't have performed better .
R	D	They said other exchanges couldn 't have performed better with the exchange 's trading links , equipment , procedures and personnel .
O	I	Five US citizens accused of plotting jihad , attempting to join Al-Qaeda and planning terrorist attacks are due to appear in court in Pakistan 's Punjab province on Tuesday .
R	I	Five US citizens accused of plotting jihad , planning terrorist attacks and attempting to join Al-Qaeda are due to appear in court on Tuesday in Pakistan 's Punjab province .
O	S	He said it appeared to be a football issue and that it was " downright despicable " for anyone to try and associate the Apprentice Boys with the trouble .
R	S	He said it was " downright despicable " for anyone to try and associate it with the Apprentice Boys and that the trouble appeared to be a football issue .
O	D	He then told me I was free to go after saying I ought to be more careful .
R	D	He then told me I ought to be more careful after saying I was free to go .

Table 2.5: The first three examples are from PTB and the final three are from GBW, for the original sentence (O) and the re-ordered counterpart (R). These re-ordered examples are grammatically and semantically acceptable, and they are marked as semantically identical (I), similar (S), or different (D) than the original source. The gradient from semantically identical to different is subjective (see text for a further discussion).

Among Acceptable Re-ordered Sentences w/o S.-R. Search Error				
Dataset	Number of Sentences	Semantics (Source vs. Re-ordered) Identical	Similar	Different
PTB	49	0.306	0.204	0.49
GBW	47	0.638	0.064	0.298

Table 2.6: Proportions of re-ordered sentences that have identical, similar, or different meanings as the original source sentences, among those re-ordered sentences that are grammatically and semantically acceptable without source-relative search error. The gradient from semantically identical to different is subjective, as noted in the text.

acceptable having different meanings than the original sentences.

Table 2.5 includes examples of re-ordered sentences without source-relative search error that were marked as acceptable. Note that for illustrative purposes, the semantically identical examples in Table 2.5 were chosen to not have the same order as the original source; however, in most cases when a re-ordered sentence is marked as having the exact same meaning as the corresponding source, it exactly matches the order of the original source. Table 2.6 includes summary statistics of the proportions of re-ordered sentences that have identical, similar, or different meanings as the original source sentences, among the aforementioned subset.

The relative meaning comparisons are very noisy and subjective.[10] However, the takeaway is that the results in the aggregate suggest that it is not uncommon for the re-ordered sentences to have different meanings than the original sentences, even if

[10]Formulating and articulating a standard for semantic comparisons would be a project onto itself. We speculate that semantic annotations at the sentence level may suffer from some of the same consistency, scale, and signal problems as hand-annotated syntactic structures, and such approaches may not be particularly effective for downstream tasks on the margin at scale.

they are grammatically acceptable.

2.5 Additional Analysis: Frequency Distribution

At the analyzed scale, the model still makes errors. Are there systematic errors that are being made? In particular, continuing the theme of the earlier work considered in Chapter 1, are there systematic errors that suggest that adding explicit syntactic rules could help resolve? Setting aside for the moment the difficulty of annotating at scale and the challenge of introducing hand-annotated structures that take into account all edge cases (i.e., not inadvertently introducing errors), and that such signal was not helpful in the controlled experiments of Chapter 1, is there otherwise reason to believe such signal could be useful in closing the gap on the remaining errors? The following analysis would tend to cast doubt on that possibility, at least when large-scale surface-level information is available.

One way to analyze whether the errors suggest that adding syntactic rules could help is by looking at whether there appear to be systematic errors (by which the syntactic rules could push the model toward the correct structure that is not captured by the model), or whether the errors are instead more simply the result of unseen ngrams in the training set. The latter seems to be the case.

Table 2.7 shows the average per-sentence proportions of n-grams from the source sentences that appear in training, separated by whether the corresponding re-ordered

Among Re-ordered Sentences w/o S.-R. Search Error						
Dataset, grouped by acceptability	Avg. Length	Avg. Proportion of Covered Source N-grams				
		Unigrams*	Bigrams	Trigrams	4-grams	5-grams
PTB, acceptable	18.9	0.995	0.958	0.785	0.553	0.398
PTB, not acceptable	26.3	0.993	0.932	0.696	0.432	0.278
GBW, acceptable	18.3	1	0.987	0.853	0.660	0.534
GBW, not acceptable	28.6	1	0.949	0.741	0.480	0.301

Table 2.7: Average lengths and average per-sentence proportions of n-grams from the source sentences that appear in training. The values here are for the subset of re-ordered sentences without source-relative search error. *The unigram value is the average proportion of tokens in the LM vocabulary (i.e., non-unknown tokens).

sentences are marked as acceptable or not acceptable. We see that the means are consistently lower for the sentences marked not acceptable.

Among the re-ordered sentences without source-relative search error, none of the corresponding PTB source sentences have 5-grams that are fully covered in the training data. The maximum 5-gram coverage among the acceptable sentences is about 0.864, and 0.5 for the not acceptable sentences. Among the GBW sentences, the maximum 5-gram coverage among the not acceptable sentences is about 0.471. However, among the GBW source sentences corresponding to the acceptable re-ordered sentences, three have 5-grams that are fully covered in the training data (i.e., coverage proportions of 1), and of these, one of the sentences appears verbatim in the training sentences. (All three of these sentences were re-ordered by the model to exactly match the original source.) Thus, the evaluation sentences rarely exactly match those seen in training, but along with the observed average coverage means (see Table 2.7), the aforementioned observations suggest that the GBW sentences may indeed be

Among Re-ordered Sentences w/o S.-R. Search Error & Length <= mean length						
Dataset, grouped by acceptability	Avg. Length	Avg. Proportion of Covered Source N-grams				
		Unigrams*	Bigrams	Trigrams	4-grams	5-grams
PTB, acceptable	15.1	0.991	0.961	0.79	0.575	0.444
PTB, not acceptable	17.3	0.992	0.909	0.709	0.483	0.345
GBW, acceptable	13.4	1	0.988	0.871	0.702	0.597
GBW, not acceptable	16.0	1	0.934	0.658	0.484	0.323

Table 2.8: Average lengths and average per-sentence proportions of n-grams from the source sentences that appear in training, for sentences less than or equal to the mean length for the respective dataset. The values here are for the subset of re-ordered sentences without source-relative search error. *The unigram value is the average proportion of tokens in the LM vocabulary (i.e., non-unknown tokens).

more similar to the training data than the PTB sentences. This may contribute to the observed greater model effectiveness on the GBW sentences.

The proportions in Table 2.7 control for sentence length in so far as they are averages over per-sentence proportions of covered n-grams, but we may nonetheless wonder whether the differences between the acceptable and not acceptable proportions are largely attributable to length. The not acceptable sentences are clearly longer on average.

Tables 2.8 and 2.9 break the proportions down into two groups separated by the mean length. Additionally, Figures 2.1 and 2.2 provide a view of sentence length vs. the proportion of source n-grams. With these views we see that the basic pattern holds for both shorter and longer sentences, with problematic sentences tending to have higher proportions of n-grams that were unseen in training. Based on these results, we might expect to see further gains in effectiveness by simply further increas-

Among Re-ordered Sentences w/o S.-R. Search Error & Length > mean length						
Dataset, grouped by acceptability	Avg. Length	Avg. Proportion of Covered Source N-grams				
		Unigrams*	Bigrams	Trigrams	4-grams	5-grams
PTB, acceptable	25.5	1	0.953	0.776	0.514	0.319
PTB, not acceptable	29.9	0.993	0.941	0.691	0.412	0.251
GBW, acceptable	25.5	1	0.985	0.827	0.599	0.442
GBW, not acceptable	31.8	1	0.953	0.762	0.478	0.296

Table 2.9: Average lengths and average per-sentence proportions of n-grams from the source sentences that appear in training, for sentences greater than the mean length for the respective dataset. The values here are for the subset of re-ordered sentences without source-relative search error. *The unigram value is the average proportion of tokens in the LM vocabulary (i.e., non-unknown tokens).

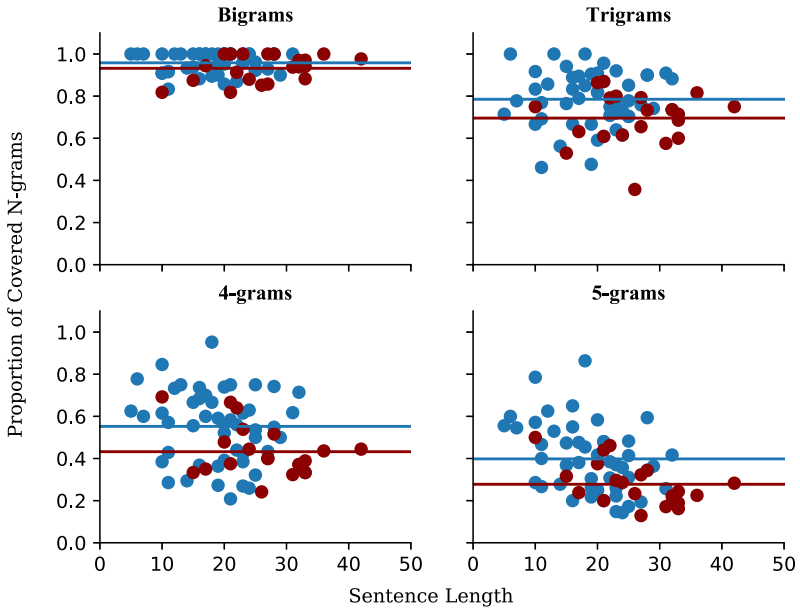

Figure 2.1: Sentence length vs. proportion of covered source n-grams (i.e., proportion of source n-grams that appear in training), for the re-ordered PTB sentences without source-relative search error. The horizontal lines represent the mean proportions of covered source n-grams. Grammatically and semantically acceptable re-ordered sentences are blue, and not acceptable re-ordered sentences are red.

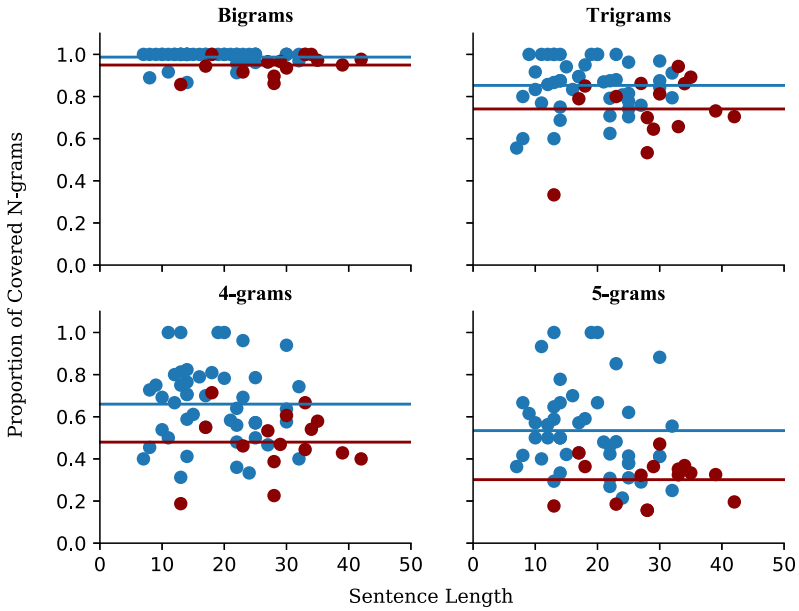

Figure 2.2: Sentence length vs. proportion of covered source n-grams (i.e., proportion of source n-grams that appear in training), for the re-ordered GBW sentences without source-relative search error. The horizontal lines represent the mean proportions of covered source n-grams. Grammatically and semantically acceptable re-ordered sentences are blue, and not acceptable re-ordered sentences are red.

ing the amount of training data.

On the other hand, although the sample size is on the small size, it is less clear that there are particular syntactic structures that are systematically challenging for the model. For example, the model makes a preposition error in the fourth re-ordered sentence in Table 2.4; however, there are a number of cases in which the re-ordered sentences correctly handle prepositions. At the same time, it is obvious that the model is relatively brittle to data that diverges from what was seen in training, and the model has no general world knowledge beyond that seen in training, which are some of the

limitations that we now further highlight.

2.6 Limitations

Surface-level language models, such as those examined here and in Chapter 1, are clearly brittle in the face of data that diverges from that seen in training, and of course, they have no general world knowledge beyond that encoded in the sequential training data. Thus, it is perhaps not surprising that there are observed instances of re-ordered sentences that are roughly grammatically acceptable but clearly not semantically acceptable, at least without qualifying context that would be unlikely in the given news domains. Adversarial examples could likely be readily constructed that would be challenging for such models on a semantic level.

Additionally, we assume that the remaining search errors could be resolved in practice with additional computing resources (as, for example, re-ordering with a beam size greater than 20,000), but we leave to future work to verify this. The sample size here is on the small side, restricted to the news domain (generally construed), so it would also be of interest to analyze additional sentences, including those from alternative domains.

Finally, while the hand-annotated syntactic rules do not appear to be particularly necessary as ordering constraints when large-scale surface-level information is available, other types of hand-written rules and feature engineering may well be useful for

NLP applications in practice, or when the amount of surface-level information is very limited.[11]

2.7 Conclusion: What Remains to be Ordered at the Sentence Level?

The word ordering task is a diagnostic task designed to identify whether or not additional structures are needed for modeling ordering constraints. Based on the results of this chapter and Chapter 1, it appears that given adequate computing resources, the observed surface-level models are able to capture some amount of the natural hierarchy and structure in language at the sentence level, without explicitly providing such structure in training annotations.[12] It appears that the networks are reasonably effective at encoding the ordering constraints at scale.

Our error analyses suggest that such models would further benefit from additional data. Training on additional data is already a practical option in some computing environments, as contemporary NLP models have already been trained in industrial settings with additional data. For example, the deep bi-directional model referred to as BERT, which has shown promising results as a generalized pre-training substrate

[11]For example, replacing a computational algebra system, as part of an NLP pipeline, with a surface-level model trained on large amounts of examples may well not be the right approach given the limitations of such models. It is not clear whether at a very large scale, such models would be able to back-out the rules of a computer algebra system by frequency information alone. The generalization ability may turn out to be too limited and/or require a scale of training instances that may remain infeasible, and in any case, the stochasticity may be an undesirable property in such a domain. In other words, such models do not necessarily generalize to other tasks.

[12]The human engineering is instead pushed to the construction of the model architecture.

for various NLP tasks, is trained on about 3,300 million words (Devlin et al., 2018), which is roughly four times the size of the GBW dataset.[13] Yet, the training dataset for BERT is still presumably a small fraction of the existing text in English.

At the same time, it is clear that such models are not robust against semantic errors and are generally relatively brittle to data not seen in training.

Acknowledgments

The work in this chapter was made possible by the public release of the pre-trained model of Dauphin et al. (2017).

[13]Note that the BERT model is a surface-level model but not a language model, as it does not provide a probability distribution over sentences. As such, it was not considered for the experiments here. More generally, it remains an open research question whether such models or language models are more (or similarly) effective pre-training substrates at scale, when controlling for various experimental conditions.

3

Error Identification

In the previous chapters, we presented evidence to suggest that hand-annotated syntactic trees may not yield additional signal for modeling ordering constraints beyond that which is already present in the surface-level lexical items, at least in the context of modern language modeling approaches. Based on this result, here and in subsequent chapters, we approach grammatical error correction tasks with surface-level

models. In this chapter, we present an approach for using sequence-to-sequence models, a type of conditional language model, for error identification, a binary classification task identifying whether or not a sentence has a grammatical error. In Chapter 4, we present additional approaches for adapting these models to the task of correction. The results presented in this chapter were externally validated in a shared task competition held in 2016, as noted below, in which our final model was the most effective based on the shared task metrics.

3.1 Introduction

Grammatical error identification, or determining where or not a sentence has an error, has potential utility as a component of a writing support and learning tool.[1] Much of the existing work on the related task of grammatical error correction has made use of hand-tuned rules and features that augment data-driven approaches, or individual classifiers for human-designated subsets of errors. Given a large, annotated dataset of scientific journal articles, we instead propose a fully data-driven approach for this problem, inspired by recent work in neural machine translation and more generally, sequence-to-sequence learning (Sutskever et al., 2014; Bahdanau et al., 2014; Cho et al., 2014).

[1]For learners at certain levels, it may actually be more helpful (for pedagogical purposes) to indicate errors at the level of the sentence rather than at lower levels of granularity, such as at the word or phrase level. In Chapter 6, we briefly suggest some possible avenues for future research to empirically analyze this with end-user studies.

The Automated Evaluation of Scientific Writing (AESW) 2016 dataset is a collection of nearly 10,000 scientific journal articles (over 1 million sentences) published between 2006 and 2013 and annotated with corrections by professional, native English-speaking editors. The goal of the associated AESW Shared Task is to identify whether or not a given unedited source sentence was corrected by the editor (that is, whether a given source sentence has one or more grammatical errors, broadly construed).

We present an attention-based sequence-to-sequence model for the task, and we demonstrate the use of a character-aware version of the model, building on the character-aware language modeling work of Kim et al. (2016). Our final model, as well as an ensemble of a generative sequence-to-sequence model and a discriminative CNN classifier, established the highest-performing public baseline for the binary prediction task submitted to the AESW 2016 Shared Task.

3.2 Background

Recent work in natural language processing has shown strong results in sequence-to-sequence transformations using recurrent neural network models (Cho et al., 2014; Sutskever et al., 2014). Grammar correction and error identification can be cast as a sequence-to-sequence translation problem, in which an unedited (*source*) sentence is "translated" into a corrected (*target*) sentence in the same language. Using this framework, sentence-level error identification then simply reduces to an equality check

between the source and target sentences.

The goal of the AESW shared task is to identify whether a particular sentence needs to be edited (contains a "grammatical" error, broadly construed[2]). The dataset consists of sentences taken from academic articles annotated with corrections by professional editors. Annotations are described via insertions and deletions, which are marked with start and end tags, which we refer to as "diffs".

Tokens to be deleted are surrounded with the deletion start tag <del> and the deletion end tag </del> and tokens to be inserted are surrounded with the insertion start tag <ins> and the insertion end tag </ins>. Replacements (as shown in Figure 3.1) are represented as deletion-insertion pairs. Unlike the related CoNLL-2014 Shared Task (Ng et al., 2014) data, errors are not labeled with fine-grained types (article or determiner error, verb tense error, etc.).

More formally, we assume a vocabulary $\mathcal{V}$ of natural language word types (some of which have orthographic errors) and a set $\mathcal{Q}$ ={<ins>,</ins>,~~,~~} of annotation tags. Given a sentence $\mathbf{s} = [s_1, \ldots, s_I]$, where $s_i \in \mathcal{V}$ is the i-th token of the sentence of length I, we seek to predict whether or not the gold, annotated target sentence $\mathbf{t} = [t_1, \ldots, t_J]$, where $t_j \in \mathcal{Q} \cup \mathcal{V}$ is the j-th token of the annotated sentence of length J, is identical to $\mathbf{s}$. We are given both $\mathbf{s}$ and $\mathbf{t}$ for supervised training. At test time, we are only given access to sequence $\mathbf{s}$. We learn to predict sequence $\mathbf{t}$.

[2]Some insertions and deletions in the shared task data represent stylistic choices, not all of which are necessarily recoverable given the sentence or paragraph context. For the purposes here, we refer to all such edits as "grammatical" errors.

Evaluation of this binary prediction task for the shared task is via the F_1-score, where the positive class is that indicating an error is present in the sentence (that is, where $\mathbf{s} \neq \mathbf{t}$).

Evaluation is at the sentence level, but the paragraph-level context for each sentence is also provided. The paragraphs, themselves, are shuffled so that full article context is not available. A coarse academic field category is also provided for each paragraph. Our models described in this chapter do not make use of the paragraph context nor the field category, and they treat each sentence independently. We explore modeling the field category in the next chapter.

Further information about the task is available in the AESW 2016 Shared Task report (Daudaravicius et al., 2016a).

3.3 Related Work

AESW 2016 was the first shared task focusing on sentence-level binary error identification, whereas previous work and shared tasks focused on the related tasks of intra-sentence identification and correction of errors. Until recently, standard hand-annotated grammatical error datasets were not available, complicating comparisons and limiting the choice of methods used. Given the lack of a large hand-annotated corpus at the time, Park and Levy (2011) demonstrated the use of the EM algorithm for parameter learning of a noise model using error data without corrections, perform-

ing evaluation on a much smaller set of sentences hand-corrected by Amazon Mechanical Turk workers.

More recent work has emerged as a result of a series of grammatical error shared tasks, starting with the Helping Our Own (HOO) Pilot Shared Task run in 2011, which focused on a diverse set of errors in a small dataset (Dale and Kilgarriff, 2011), and the subsequent HOO 2012 Shared Task, which focused on the automated detection and correction of preposition and determiner errors (Dale et al., 2012). The CoNLL-2013 Shared Task (Ng et al., 2013)[3] focused on the correction of a limited set of five error types in essays by second-language learners of English at the National University of Singapore. The follow-up CoNLL-2014 Shared Task (Ng et al., 2014)[4] focused on the full generation task of correcting all errors in essays by second-language learners.

As with machine translation (MT), evaluation of the full generation task is still an open research area, but a subsequent human evaluation ranked the output from the CoNLL-2014 Shared Task systems (Napoles et al., 2015). The system of Felice et al. (2014) ranked highest, utilizing a combination of a rule-based system and phrase-based MT, with re-ranking via a large web-scale language model. Of the non-MT based approaches, the Illinois-Columbia system was a strong performer, combining several classifiers trained for specific types of errors (Rozovskaya et al., 2014). Since

[3] `http://www.comp.nus.edu.sg/~nlp/conll13st.html`
[4] `http://www.comp.nus.edu.sg/~nlp/conll14st.html`

our submission to the AESW Shared Task, translation-based approaches to correction have become standard approaches, at least in the academic literature. Additional works on the correction task appearing after the AESW Shared Task are discussed in the next chapter.

3.4 Models

We use an end-to-end approach that does not have separate components for candidate generation or re-ranking that make use of hand-tuned rules or explicit syntax, nor do we employ separate classifiers for human-differentiated subsets of errors, unlike some previous work for the related task of grammatical error correction.

We next introduce two approaches for the task of sentence-level grammatical error identification: A binary classifier and a sequence-to-sequence model that is trained for correction but can also be used for identification as a side-effect.

3.4.1 Baseline Convolutional Neural Net

To establish a baseline, we follow past work that has shown strong performance with convolutional neural nets (CNNs) across various domains for sentence-level classification (Kim, 2014; Zhang and Wallace, 2015). We utilize the one-layer CNN architecture of Kim (2014) with the publicly available[5] word vectors trained on the Google News dataset, which contains about 100 billion words (Mikolov et al., 2013). We ex-

[5] `https://code.google.com/archive/p/word2vec/`

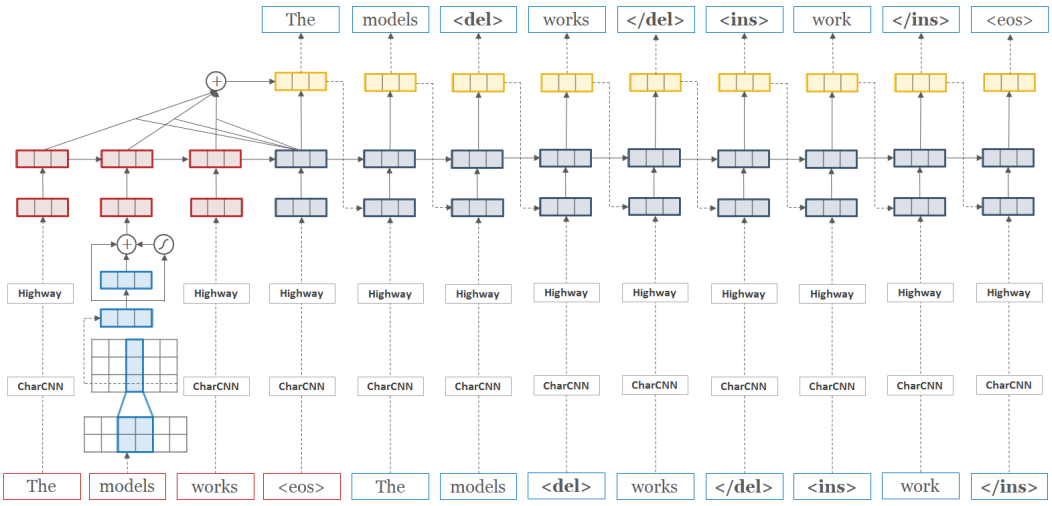

Figure 3.1: An illustration of the CHARCNN model architecture translating an example source sentence into the corrected target with a single word replacement. A CNN (here, with three filters of width two) is applied over character embeddings to obtain a fixed dimensional representation of a word, which is given to a highway network (in light blue, above). Output from the highway network is used as input to a LSTM encoder/decoder. At each step of the decoder, its hidden state is interacted with the hidden states of the encoder to produce attention weights (for each word in the encoder), which are used to obtain the context vector via a convex combination. The context vector is combined with the decoder hidden state through a one layer MLP (yellow), after which an affine transformation followed by a softmax is applied to obtain a distribution over the next word/tag. The MLP layer (yellow) is used as additional input (via concatenation) for the next time step. Generation continues until the <eos> symbol is generated.

periment with keeping the word vectors static (CNN-STATIC) and fine-tuning the vectors (CNN-NONSTATIC). The CNN models only have access to sentence-level labels and are not given correction-level diff annotations, since these binary classification models have no means of generating the diffs at test time.

3.4.2 Sequence-to-Sequence Model

We propose two sequence-to-sequence[6] architectures for this task. Our word-based architecture (Word) is similar to that of Luong et al. (2015). Our character-based models (CharCNN) still make predictions at the word-level, but use a CNN and a highway network over characters instead of word embeddings as the input to the encoder and decoder, as depicted in Figure 3.1. We follow past work (Sutskever et al., 2014; Luong et al., 2015) in stacking multiple recurrent neural networks (RNNs), specifically Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks, as used in Chapter 1, in both the encoder and decoder.

Here, we model the probability of the target given the source, $p(\mathbf{t} \mid \mathbf{s})$, with an *encoder* neural network that summarizes the source sequence and a *decoder* neural network that generates a distribution over the target words and diff tags at each step given the source.

We start by describing the basic encoder and decoder architectures in terms of the Word model, and then we describe the CharCNN model departures from Word.

[6]Sequence-to-sequence models are also referred to as encoder-decoder models in the literature. Here, we generally refer to the overall model as sequence-to-sequence, which consists of an encoder and decoder, which may differ in their individual architectures.

Encoder

The encoder reads the source sentence and outputs a sequence of vectors, associated with each word in the sentence, which will be selectively accessed during decoding via a soft attentional mechanism. We use an LSTM network to obtain the hidden states $\mathbf{h}_i^s \in \mathbb{R}^n$ for each time step i,

$$\mathbf{h}_i^s = \text{LSTM}(\mathbf{h}_{i-1}^s, \mathbf{x}_i^s).$$

For the Word models, $\mathbf{x}_i^s \in \mathbb{R}^m$ is the word embedding for s_i, the i-th word in the source sentence. (The analogue for the CharCNN models is discussed below.) The output of the encoder is the sequence of hidden state vectors $[\mathbf{h}_1^s, \ldots, \mathbf{h}_I^s]$. The initial hidden state of the encoder is set to zero (i.e. $\mathbf{h}_0^s \leftarrow \mathbf{0}$).

Decoder

The decoder is another LSTM that produces a distribution over the next target word/tag given the source vectors $[\mathbf{h}_1^s, \ldots, \mathbf{h}_I^s]$ and the previously generated target words/tags $\mathbf{t}_{<j} = [t_1, \ldots t_j]$. Let

$$\mathbf{h}_j^t = \text{LSTM}(\mathbf{h}_{j-1}^t, \mathbf{x}_j^t)$$

be the summary of the target sentence up to the j-th word, where $\mathbf{x}_j^t$ is the word embedding for t_j in the Word models. The current target hidden state $\mathbf{h}_j^t$ is combined

with each of the memory vectors in the source to produce attention weights as follows,

$$u_{j,i} = \mathbf{h}_j^t \cdot \mathbf{W}_\alpha \mathbf{h}_i^s$$

$$\alpha_{j,i} = \frac{\exp u_{j,i}}{\sum_{k \in [1,I]} \exp u_{j,k}}$$

The source vectors are multiplied with the respective attention weights, summed, and interacted with the current decoder hidden state $\mathbf{h}_j^t$ to produce a *context* vector $\mathbf{c}_j$,

$$\mathbf{v}_j = \sum_{i \in [1,I]} \alpha_{j,i} \mathbf{h}_i^s$$

$$\mathbf{c}_j = \tanh(\mathbf{W}[\mathbf{v}_j; \mathbf{h}_j^t])$$

The probability distribution over the next word/tag is given by applying an affine transformation to $\mathbf{c}_j$ followed by a softmax,

$$p(t_{j+1} \,|\, \mathbf{s}, \mathbf{t}_{<j}) = \text{softmax}(\mathbf{U}\mathbf{c}_j + \mathbf{b})$$

Finally, as in Luong et al. (2015), we feed $\mathbf{c}_j$ as additional input to the decoder for the next time step by concatenating it with $\mathbf{x}_j^t$, so the decoder equation is modified to,

$$\mathbf{h}_j^t = \text{LSTM}(\mathbf{h}_{j-1}^t, [\mathbf{x}_j^t; \mathbf{c}_{j-1}])$$

This allows the decoder to have knowledge of previous (soft) alignments at each time step. The decoder hidden state is initialized with the final hidden state of the encoder (i.e. $\mathbf{h}_0^t \leftarrow \mathbf{h}_I^s$).

Character Convolutional Neural Network

For the CharCNN models, instead of a word embedding, our input for each word in the source/target sentence is an output from a character-level convolutional neural network (depicted in Figure 3.1). Our character model closely follows that of Kim et al. (2016).

Suppose word s_i is composed of characters $[p_1, \ldots, p_l]$. We concatenate the character embeddings to form the matrix $\mathbf{P}_i \in \mathbb{R}^{c \times l}$, where the k-th column corresponds to the character embedding for p_k (of dimension c).

We then apply a convolution between $\mathbf{P}_i$ and a *filter* $\mathbf{H} \in \mathbb{R}^{c \times w}$ of width w, after which we add a bias and apply a nonlinearity to obtain a *feature map* $\mathbf{f}_i \in \mathbb{R}^{l-w+1}$. The k-th element of $\mathbf{f}_i$ is given by,

$$\mathbf{f}_i[k] = \tanh(\langle \mathbf{P}_i[*, k : k + w - 1], \mathbf{H} \rangle + b)$$

where $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{Tr}(\mathbf{A}\mathbf{B}^T)$ is the Frobenius inner product and $\mathbf{P}_i[*, k : k + w - 1]$ is the

k-to-$(k+w-1)$-th column of $\mathbf{P}_i$. Finally, we take the *max-over-time*

$$z_i = \max_k \mathbf{f}_i[k]$$

as the feature corresponding to filter $\mathbf{H}$. We use multiple filters $\mathbf{H}_1, \ldots \mathbf{H}_h$ to obtain a vector $\mathbf{z}_i \in \mathbb{R}^h$ as the representation for a given source/target word or tag. We have separate character-level convolutional neural networks for the encoder and decoder.

HIGHWAY NETWORK Instead of replacing the word embedding $\mathbf{x}_i$ with $\mathbf{z}_i$, we feed $\mathbf{z}_i$ through a *highway network* (Srivastava et al., 2015). Whereas a multilayer perceptron produces a new set of features via the following transformation (given input $\mathbf{z}$),

$$\hat{\mathbf{z}} = f(\mathbf{W}\mathbf{z} + \mathbf{b})$$

a highway network instead computes,

$$\hat{\mathbf{z}} = \mathbf{r} \odot f(\mathbf{W}\mathbf{z} + \mathbf{b}) + (\mathbf{1} - \mathbf{r}) \odot \mathbf{z}$$

where f is a non-linearity (in our models, ReLU), $\odot$ is the element-wise multiplication operator, and $\mathbf{r} = \sigma(\mathbf{W}_r\mathbf{z} + \mathbf{b}_r)$ and $\mathbf{1} - \mathbf{r}$ are called the *transform* and *carry* gates. We feed $\mathbf{z}_i$ into the highway network to obtain $\hat{\mathbf{z}}_i$, which is used to replace the input word embeddings in both the encoder and the decoder.

Inference

Exact inference is computationally infeasible for the sequence-to-sequence models given the combinatorial explosion of possible output sequences, but we follow past work in neural machine translation (NMT) using beam search. We do not constrain the generation process of words outside insertion tags to words in the source, and each low-frequency holder token generated in the target sentence is replaced with the source token associated with the maximum attention weight. We use a beam size of 10 for all models, with the exception of one of the models in the final system combination, for which we use a beam of size 5, as noted in Section 3.6.

Note that this model generates corrections, but we are only interested in determining the existence of any error at the sentence-level. As such, after beam decoding, we simply check for whether there were any corrections in the target. However, we found that decoding in this way under-predicts sentence-level errors. It is therefore important to calibrate the weights associated with corrections, which we discuss in Section 3.5.3.

3.5 Experiments

3.5.1 Data

The AESW task data differs from previous grammatical error datasets in terms of scale and genre. To the best of our knowledge, the AESW dataset is the first large-scale, publicly available professionally edited dataset of academic, scientific writing. The training set consists of 466,672 sentences with edits and 722,742 sentences without edits, and the development set contains 57,340 sentences with edits and 90,106 sentences without. The raw training and development datasets are provided as annotated sentences, **t**, from which the **s** sequences may be deterministically derived (by simply removing the diff tags). There are 143,802 sentences in the Shared Task test set with hidden gold labels, which serve directly as **s** sequences.

As part of pre-processing, we treat each sentence independently, discarding paragraph context (which sentences, if any, were present in the same paragraph) and domain information, which is a coarse grouping by the field of the original journal (Engineering, Computer Science, Mathematics, Chemistry, Physics, etc.). We generate Penn Treebank style tokenizations of the input. Case is maintained and digits are not replaced with holder symbols. The vocabulary is restricted to the 50,000 most common tokens, with remaining low frequency tokens replaced with a special <unk> token. The CharCNN model can encode but not decode over open vocabularies and

hence we do not have any <unk> tokens on the source side of those models. For all of the sequence-to-sequence models, we replace the low-frequency target symbols during inference as discussed above in Section 3.4.2.

For development against the provided data with labels, we set aside a 10,000 sentence sample from the original development set for tuning, and use the remaining 137,446 sentences for validation[7]. The sequence-to-sequence models are given all 466,672 pairs of **s** and **t** sequences with edits, augmented with varying numbers of pairs without edits. The CharCNN+sample and Word+sample models are given a random sample of 200,000 pairs without edits for a total of 666,672 pairs of **s** and **t** sequences. The CharCNN+all and Word+all models are given all 722,742 sentences without edits for a total of 1,189,414 pairs of **s** and **t** sequences. For some of the final testing models, we also train with the development sentences. In these latter cases, all sequence pairs are used. In training all of the sequence-to-sequence models, as indicated in Section 3.5.2, we drop sentences exceeding 50 tokens in length.

We also experimented with creating corrected versions of sentences for the CNN. The binary CNN classifiers are given 1,656,086 single-sentence training examples, of which 722,742 are error-free examples (in which $\mathbf{s} = \mathbf{t}$), and the remaining examples are constructed by removing the tags from the annotated sentences, **t**, to create tag-free examples that contain errors (466,672 instances) and additional error-free exam-

[7]Note that the number of sentences in the final development set without labels posted on CodaLab (`http://codalab.org`) differed from that originally posted on the AESW 2016 Shared Task website with labels.

ples (466,672 instances).

3.5.2 Training

Training (along with testing) of all models was conducted on GPUs. Our models were implemented with the Torch[8] framework.

CNN

Architecture and training approaches were informed by past work on sentence-level classification using CNNs (Kim, 2014; Zhang and Wallace, 2015). A limited grid search on the development set determined our use of filter windows of width 3, 4, and 5 and 1000 feature maps. We trained for 10 epochs. Training otherwise followed the approach of the correspondingly named CNN-static and CNN-nonstatic models of Kim (2014).

Sequence-to-Sequence

Initial parameter settings (including architecture decisions such as the number of layers and embedding and hidden state sizes) were informed by concurrent work in neural machine translation and existing work such as that of Sutskever et al. (2014) and Luong et al. (2015). We used 4-layer LSTMs with 1000 hidden units in each layer. We trained for 14 epochs with a batch size of 64 and a maximum sequence length of

[8] `http://torch.ch`

50. The parameters for the WORD model were uniformly initialized in $[-0.1, 0.1]$, and those of the CHARCNN model were uniformly initialized in $[-0.05, 0.05]$. The L_2-normalized gradients were constrained to be ≤ 5. Our learning rate schedule started the learning rate at 1 and halved the learning rate after each epoch beyond epoch 10, or once the validation set perplexity no longer improved. The WORD model used 1000-dimensional word embeddings. For CHARCNN, the character embeddings were 25-dimensional, the filter width was 6, the number of feature maps was 1000, and 2 highway layers were used. The maximum word length was 35 characters for training CHARCNN. Note that we do not reverse the source ($\mathbf{s}$) sequences, unlike some previous NMT work. Following the work of Zaremba et al. (2014), we employed dropout with a probability of 0.3 between the LSTM layers.

3.5.3 TUNING

Post-hoc tuning was performed on the 10k held-out portion of the development set. In terms of maximizing the F_1-score, this post-hoc tuning was important for these models, without which precision was high and recall was low.

For the CNN models, after training, we tuned the decision boundary to maximize the F_1-score on the held-out tuning set. Analogously, for the sequence-to-sequence models, after training the models, we tuned the bias weights (given as input to the final softmax layer generating the words/tags distribution) associated with the four annotation tags via a simple grid search by iteratively running beam search on the

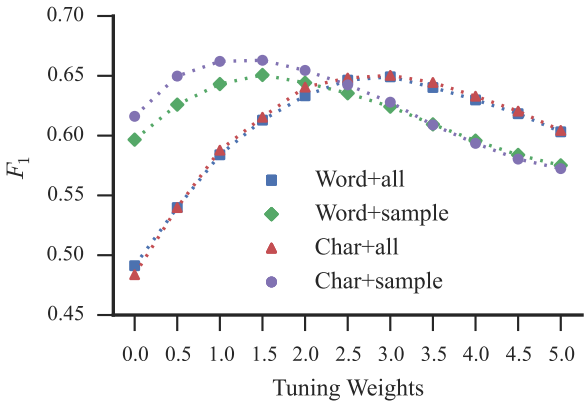

Figure 3.2: F_1 scores for varying values applied additively to the bias weights of the four annotation tags on the held-out 10k tuning subset.

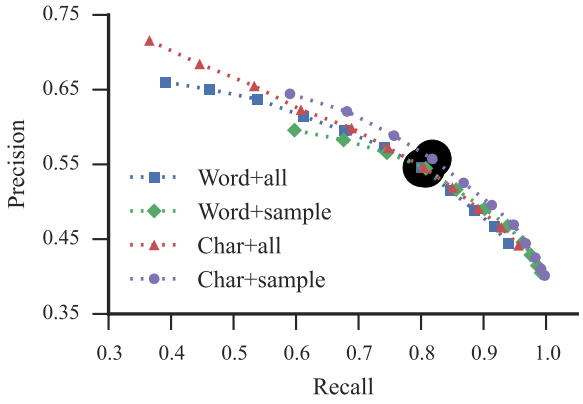

Figure 3.3: Precision vs. recall trade-off as the bias weights associated with the four annotation tags are varied on the held-out 10k tuning subset. The points yielding maximum F_1 scores are highlighted with black circles.

tuning set. Due to the relatively high expense of decoding, we employed a coarse grid search in which the bias weights of the four annotation tags were uniformly varied.

Model	Data	Precision	Recall	F_1
RANDOM	N/A	0.3885	0.4992	0.4369
CNN-STATIC	Training+word2vec	0.5349	0.7586	0.6274
CNN-NONSTATIC	Training+word2vec	0.5365	0.7758	0.6343
WORD+ALL	Training	0.5399	0.7882	0.6408
WORD+SAMPLE	Training	0.5394	0.8024	0.6451
CHARCNN+ALL	Training	0.5400	0.8048	0.6463
CHARCNN+SAMPLE	Training	0.5526	0.8126	0.6579

Table 3.1: Experimental results on the development set excluding the held-out 10k tuning subset.

Model	Data	Precision	Recall	F_1
RANDOM	N/A	0.3921	0.5981	0.4736
WORD+ALL	Training	0.5343	0.7577	0.6267
WORD+SAMPLE	Training	0.5335	0.7699	0.6303
CHARCNN+ALL	Training	0.5351	0.7749	0.6330
CHARCNN+SAMPLE	Training	0.5469	0.7803	0.6431

Table 3.2: Results on the official development set. Here, RANDOM was provided by the Shared Task organizers.

Model	Data	Precision	Recall	F_1
RANDOM	N/A	0.3607	0.6004	0.4507
KNOWLET	–	0.6241	0.3685	0.4634
NTNU-YZU	–	0.6717	0.3805	0.4858
HITS	–	0.3765	0.948	0.5389
UW-SU	–	0.4145	0.8201	0.5507
NTNU-YZU	–	0.5025	0.7785	0.6108
CHARCNN+SAMPLE	Training	0.5112	0.7841	0.6189
COMBINATION	Training+Dev+word2vec	0.5444	0.7413	0.6278

Table 3.3: Final submitted results on the Shared Task test set. COMBINATION was our final submitted system. RANDOM was provided by the Shared Task organizers. For comparison, we have included the other team submissions from National Taiwan Normal University and Yuan Ze University (NTNU-YZU), the University of Washington and Stanford University (UW-SU), HITS (HITS), and Knowlet (KNOWLET). Teams were allowed to designate up to two final submissions. (The CHARCNN model trained on the combined training and development set had not finished training by the Shared Task deadline. As such, it was not submitted, but the partially trained model was included in COMBINATION.)

3.6 Results

Results on the development set, excluding the 10k tuning set, appear in Table 3.1. Here (and elsewhere) Random is the result of randomly assigning a sentence to one of the binary classes. For the CNN classifiers, fine-tuning the word2vec embeddings improves performance. The sequence-to-sequence models improve over the CNN classifiers, even though the latter are provided with additional data (via word2vec). The character-based models yield minor improvements over the word-based models.

For consistency here, we kept the beam size at 10 across models, but subsequent analysis revealed that increasing the beam from 5 to 10 had a negligible effect on overall performance.

Tuning results appear in Figures 3.2 and 3.3, illustrating the importance of adjusting the bias weights associated with the annotation tags in balancing precision and recall to maximize the F_1 score. The models trained on all sequence pairs without edits, CharCNN+all and Word+all, perform particularly poorly without tuning these bias weights, yielding F_1 scores near that of Random before tuning, which corresponds to a weight of 0.0 in Figure 3.2.

The official development set posted on CodaLab differed slightly from the original development set provided with labels, so we include those results in Table 3.2 for the sequence-to-sequence models. Here, evaluation is performed on the CodaLab server, as with the final test submission. The overall relative performance pattern is similar

to that of the original development set.

A comparison of our results with other shared task submissions appears in Table 3.3. (Teams were allowed to submit up to two results.) Our submission, COMBINATION was a simple majority vote at the system level (for each test sentence) of 5 models[9]: (1) a CNN-NONSTATIC model trained with the concatenation of the training and development sets (and using word2vec); (2) a WORD model trained on all sequence pairs in the training and development sets with a beam size of 10 for decoding; (3,4) a CHARCNN+SAMPLE model trained on the training set, decoding the test set twice, each time with different weight biases (the two highest performing via the grid search over the tuning set) with a beam size of 10; and (5) a CHARCNN model trained on all sequence pairs in the training and development sets, with training suspended at epoch 9 (out of 14) and a beam size of 5 to meet the Shared Task deadline. For reference, we also include the CodaLab evaluation for just the CHARCNN+SAMPLE model trained on the training set with a beam size of 10, with the bias weights being those that generated the highest F_1-score on the 10k tuning set.

3.7 DISCUSSION

For this binary prediction task, the CHARCNN+SAMPLE model performs well, both in terms of effectiveness on the test set relative to other submissions, as well as on the

[9]The choice of models was limited to those that were trained and tuned in time for the Shared Task deadline.

development set relative to the WORD models and the CNN classifiers. It is possible these minor improvements in the F_1-score are due to the ability of the CHARCNN models to capture some types of orthographic errors beyond that of the word-level models.

The empirical results suggest that simply adding additional already correct source-target pairs when training the sequence-to-sequence models for the binary prediction task may not boost effectiveness, ceteris paribus, as seen in comparing the effectiveness of CHARCNN+SAMPLE vs WORD+SAMPLE, and CHARCNN+ALL vs WORD+ALL.

One benefit of the sequence-to-sequence models is that they can be used to generate corrections (and identify locations of intra-sentence errors) for end-users. We explore this further in the next chapter. However, the added generation capabilities of the sequence-to-sequence models comes at the expense of considerably longer training and testing times compared to the CNN classifiers.

We found that post-hoc tuning provides a straightforward means of tuning the precision-recall trade-off for these models, and we speculate (but leave to future work for investigation) that in practice, end-users might prefer greater emphasis placed on precision over recall.

3.8 Conclusion: Sentence-Level Grammatical Error Identification as Sequence-to-Sequence Correction

In this chapter, we have demonstrated the application of sequence-to-sequence models for the error identification task, the effectiveness of which was externally validated in a shared task relative to other approaches, including grammar-based approaches. Our models do not make use of hand-tuned rules, are not trained with explicit syntactic annotations, and do not make use of individuals classifiers designed for human-designated subsets of errors, unlike most related models in the then contemporary literature for grammar correction related tasks. At the time of submission in 2016 to the AESW Shared Task competition, the application of an end-to-end neural network based approach for a grammar correction related task was a novel contribution, and attention-based sequence-to-sequence models have subsequently become standard approaches and baselines for such tasks.

Acknowledgments

This chapter is based on Schmaltz et al. (2016a). The Institute for Quantitative Social Science (IQSS) and the Harvard Initiative for Learning and Teaching (HILT) supported earlier, related research that led to our participation in the AESW Shared Task in 2016. Jeffrey Ling contributed a torch-based CNN implementation of Kim (2014), which formed the basis for the presented CNN models.

4

Error Correction

In the previous chapter, we presented an early application of sequence-to-sequence modeling for the sentence-level grammatical error identification task. In this chapter, we shift our attention to adapting such models to the related task of sentence correction. In a series of controlled experiments for the correction task, we find that character-based models are generally more effective than word-based models and

models that encode subword information via convolutions, and that modeling the output data as a series of diffs improves effectiveness over standard approaches. Our strongest sequence-to-sequence model improves over our strongest phrase-based statistical machine translation model, with access to the same data, by 6 M^2 (0.5 GLEU) points. Additionally, in the data environment of the standard CoNLL-2014 setup, we demonstrate that modeling (and tuning against) diffs yields similar or better M^2 scores with simpler models and/or significantly less data than alternative sequence-to-sequence approaches.

4.1 Introduction

The task of *sentence correction* is to convert a natural language sentence that may or may not have errors into a corrected version. The task is envisioned as a component of a learning tool or writing-assistant, and it has seen increased interest since 2011 driven by a series of shared tasks (Dale and Kilgarriff, 2011; Dale et al., 2012; Ng et al., 2013, 2014).

Most recent work on language correction has focused on the data provided by the CoNLL-2014 shared task (Ng et al., 2014), a set of corrected essays by second-language learners. The CoNLL-2014 data consists of only around 60,000 sentences, and as such, competitive systems have made use of large amounts of corrected text without annotations, and in some cases lower-quality crowd-annotated data, in addition

to the shared data. In this data environment, it has been suggested that statistical phrase-based machine translation (MT) with task-specific features is the state-of-the-art for the task (Junczys-Dowmunt and Grundkiewicz, 2016), outperforming word- and character-based sequence-to-sequence models (Yuan and Briscoe, 2016; Xie et al., 2016; Ji et al., 2017), phrase-based systems with neural features (Chollampatt et al., 2016b,a), re-ranking output from phrase-based systems (Hoang et al., 2016), and combining phrase-based systems with classifiers trained for hand-picked subsets of errors (Rozovskaya and Roth, 2016).

We revisit the comparison across translation approaches for the correction task in light of the Automated Evaluation of Scientific Writing (AESW) 2016 dataset, the correction dataset containing over 1 million sentences introduced in Chapter 3 for the identification task, holding constant the training data across approaches.

Experiments demonstrate that pure character-level sequence-to-sequence models are more effective on AESW than word-based models and models that encode sub-word information via convolutions over characters, and that representing the output data as a series of diffs, as with the binary classification task in Chapter 3, significantly increases effectiveness on this task. Our strongest character-level model achieves statistically significant improvements over our strongest phrase-based statistical machine translation model by 6 M^2 (0.5 GLEU) points, with additional gains when including domain information. Furthermore, in the partially crowd-sourced data environment of the standard CoNLL-2014 setup in which there are compara-

tively few professionally annotated sentences, we find that tuning against the tags marking the diffs yields similar or superior effectiveness relative to existing sequence-to-sequence approaches despite using significantly less data, with or without using secondary models. The associated code is publicly available at `https://github.com/allenschmaltz/grammar`.

4.2 Task

We follow recent work and treat the task of sentence correction as translation from a source sentence (the unedited sentence) into a target sentence (a corrected version in the same language as the source). We do not make a distinction between grammatical and stylistic corrections. The basic setup of the task is similar to that of the identification task in Chapter 3; however, recall that the identification task is a binary classification task.

We assume a vocabulary $\mathcal{V}$ of natural language word types (some of which have orthographic errors). Given a sentence $\mathbf{s} = [s_1 \cdots s_I]$, where $s_i \in \mathcal{V}$ is the i-th token of the sentence of length I, we seek to predict the corrected target sentence $\mathbf{t} = [t_1 \cdots t_J]$, where $t_j \in \mathcal{V}$ is the j-th token of the corrected sentence of length J. We are given both $\mathbf{s}$ and $\mathbf{t}$ for supervised training in the standard setup. At test time, we are only given access to sequence $\mathbf{s}$. We learn to predict sequence $\mathbf{t}$, which can be identical to $\mathbf{s}$. We propose a variant of this basic formulation, modeling diffs in the target

sentences, in Section 4.3.2, below.

4.3 Methods

4.3.1 Sequence-to-Sequence Model

We explore word and character variants of the sequence-to-sequence framework. We use a standard word-based model (Word), similar to that of Luong et al. (2015), as well as a model that uses a convolutional neural network and a highway network over characters (CharCNN), based on the work of Kim et al. (2016), instead of word embeddings as the input to the encoder and decoder. With both of these models, predictions are made at the word level. These models follow the basic architecture of the corresponding models for the identification task in Chapter 3. We also consider the use of bidirectional versions of these encoders (+Bi).

Our character-based model (Char+Bi) follows the architecture of the Word+Bi model, but the input and output consist of characters rather than words. In this case, the input and output sequences are converted to a series of characters and whitespace delimiters. The output sequence is converted back to $\mathbf{t}$ prior to evaluation.

The Word models encode and decode over a closed vocabulary (of the 50k most frequent words); the CharCNN models encode over an open vocabulary and decode over a closed vocabulary; and the Char models encode and decode over an open vocabulary.

Our contribution is to investigate the impact of sequence-to-sequence approaches (including those not considered in previous work) in a series of controlled experiments, holding the data constant. In doing so, we demonstrate that on a large, professionally annotated dataset, the most effective sequence-to-sequence approach can significantly outperform a state-of-the-art SMT system without augmenting the sequence-to-sequence model with a secondary model to handle low-frequency words (Yuan and Briscoe, 2016) or an additional model to improve precision or intersecting a large language model (Xie et al., 2016). We also demonstrate improvements over these previous sequence-to-sequence approaches on the CoNLL-2014 data and competitive results with Ji et al. (2017), despite using significantly less data.

4.3.2 Additional Approaches

The standard formulation of the correction task is to model the output sequence as $\mathbf{t}$ above. Here, we also propose modeling the diffs between $\mathbf{s}$ and $\mathbf{t}$. The diffs are modeled as in Chapter 3, the basic setup of which we briefly review here. The diffs are provided in-line within $\mathbf{t}$ and are described via annotation tags marking the starts and ends of insertions and deletions, with replacements represented as deletion-insertion pairs, as in the following example selected from the training set: "Some key points are worth <del> emphasiz </del> <ins> emphasizing </ins> .". Here, "emphasiz" is replaced with "emphasizing". The models, including the Char model, treat each tag as a single, atomic token.

The diffs enable a means of tuning the sequence-to-sequence model's propensity to generate corrections by modifying the probabilities generated by the decoder for the 4 diff tags, which we examine with the CoNLL data. We include four bias parameters associated with each diff tag, and run a grid search between 0 and 1.0 to set their values based on the tuning set.

It is possible for models with diffs to output invalid target sequences (for example, inserting a word without using a diff tag). To fix this, a deterministic post-processing step is performed (greedily from left to right) that returns to source any non-source tokens outside of insertion tags.[1] Diffs are removed prior to evaluation. We indicate models that *do not* incorporate target diff annotation tags with the designator −DIFFS.

The AESW dataset provides the paragraph context and a journal domain (a classification of the document into one of nine subject categories) for each sentence.[2] For the sequence-to-sequence models we propose modeling the input and output sequences with a special initial token representing the journal domain (+DOM).[3]

[1] Alternatively, the output can be constrained to the semantics of the diff tags when decoding. In practice, we did not see a significant difference in the output, based on the measured metrics, between constrained decoding and post-hoc fixing.

[2] The paragraphs are shuffled for purposes of obfuscation, so document-level context is not available.

[3] In our experiments, we do not include additional paragraph context features, since the underlying AESW data appears to have been collected such that nearly all paragraphs (including those containing a single sentence) contain at least one error; thus, modeling paragraph information provides additional signal that seems unlikely to reflect real-world environments.

Model	GLEU		M^2	
	Dev	Test	Dev	Test
No Change	89.68	89.45	00.00	00.00
SMT–DIFFS+M^2	90.44	–	38.55	–
SMT–DIFFS+BLEU	90.90	–	37.66	–
WORD+BI–DIFFS	91.18	–	38.88	–
CHAR+BI–DIFFS	91.28	–	40.11	–
SMT+BLEU	90.95	90.70	38.99	38.31
WORD+BI	91.34	91.05	43.61	42.78
CHARCNN	91.23	90.96	42.02	41.21
CHAR+BI	**91.46**	**91.22**	**44.67**	**44.62**
WORD+DOM	91.25	–	43.12	–
WORD+BI+DOM	91.45	–	44.33	–
CHARCNN+BI+DOM	91.15	–	40.79	–
CHARCNN+DOM	91.35	–	43.94	–
CHAR+BI+DOM	**91.64**	**91.39**	**47.25**	**46.72**

Table 4.1: AESW development/test set correction results. GLEU and M^2 differences on test are statistically significant via paired bootstrap resampling (Koehn, 2004; Graham et al., 2014) at the 0.05 level, resampling the full set 50 times.

4.4 Experiments

4.4.1 Data

AESW (Daudaravicius, 2016; Daudaravicius et al., 2016b) consists of sentences taken from academic articles annotated with corrections by professional editors used for the AESW shared task, as noted in Chapter 3. The training set contains 1,182,491 sentences, of which 460,901 sentences have edits. We set aside a 9,947 sentence sample from the original development set for tuning (of which 3,797 contain edits), and use

the remaining 137,446 sentences as the dev set[4] (of which 53,502 contain edits). The test set contains 146,478 sentences.[5]

As part of preprocessing, the sentences from the AESW XML are converted to Penn Treebank-style tokenization. Case is maintained and digits are not replaced with holder symbols for the sequence-to-sequence models. For the SMT models, the truecasing[6] and tokenization pipeline of the publicly available code is used. For consistency, all model output and all reference files are converted to cased Moses-style tokenization prior to evaluation.

The primary focus of the present study is conducting controlled experiments on the AESW dataset, but we also investigate results on the CoNLL-2014 shared task data in light of recent neural results (Ji et al., 2017) and to serve as a baseline of comparison against existing sequence-to-sequence approaches (Yuan and Briscoe, 2016; Xie et al., 2016). We use the common sets of public data appearing in past work for training: the National University of Singapore (NUS) Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013) and the publicly available Lang-8 data (Tajiri et al., 2012; Mizumoto et al., 2012). The Lang-8 dataset of corrections is large[7] but is crowd-

[4]The dev set contains 13,562 unique deletion types, 29,952 insertion types, and 39,930 replacement types.

[5]The aforementioned training and tuning sizes of the AESW dataset are those after dropping sentences exceeding 126 tokens on the source or target side (in source sequences or target sequences with diff annotation tags) from the raw AESW dataset. All evaluation metrics on the development and test set are on the data without filtering based on sentence lengths.

[6]Here, the truecase language model is created from the training $\mathbf{t}$ sequences (or where applicable, the target with diffs).

[7]about 1.4 million sentences after filtering

sourced[8] and is thus of a different nature than the professionally annotated AESW and NUCLE datasets. We use the revised CoNLL-2013 test set as a tuning/dev set and the CoNLL-2014 test set (without alternatives) for testing. We do not make use of the non-public Cambridge Learner Corpus (CLC) (Nicholls, 2003), which contains over 1.5 million sentence pairs.

For training, we used the copy of the Lang-8 corpus distributed in the repo for the code of Junczys-Dowmunt and Grundkiewicz (2016): `https://github.com/grammatical/baselines-emnlp2016`. We filtered the Lang-8 data to remove duplicates and target sentences containing emoticon text, informal colloquial words (e.g., "haha", "lol", "yay"), and non-ascii characters. Target sentences not starting with a capital letter were dropped, as were target sentences not ending in a period, question mark, exclamation mark, or quotation mark. (Target sentences ending in a parenthesis were dropped as they often indicate informal additional comments from the editor.) In the combined NUCLE and Lang-8 training set, source sentences longer than 79 tokens and target sentences longer than 100 tokens were dropped. This resulted in a training set with 1,470,992 sentences. Diffs were created using the Python class difflib.SequenceMatcher.

[8]derived from the Lang-8 language-learning website

4.4.2 Evaluation

We follow past work and use the Generalized Language Understanding Evaluation (GLEU) (Napoles et al., 2016) and MaxMatch (M^2) metrics (Dahlmeier and Ng, 2012b).

4.4.3 Parameters

All our models, implemented with OpenNMT (Klein et al.), are 2-layer LSTMs with 750 hidden units. For the Word model, the word embedding size is also set to 750, while for the CharCNN and Char models we use a character embedding size of 25. The CharCNN model has a convolutional layer with 1000 filters of width 6 followed by max-pooling, which is fed into a 2-layer highway network.

For the Char model, the L_2-normalized gradients were constrained to be ≤ 1 (instead of ≤ 5 with the other models), and our learning rate schedule started the learning rate at 0.5 (instead of 1 for the other models) for stable training. The maximum sequence length of 421 was used for models given character sequences, which was equivalent to the maximum sequence length of 126 used for models given word sequences. The maximum sequence lengths were increased by 1 for the models with the +dom features.

For AESW, the Word+Bi model contains around 144 million parameters, the CharCNN+Bi model around 79 million parameters, and the Char+Bi model around 25 million parameters.

For the sequence-to-sequence models, the closed vocabularies were restricted to the 50,000 most common tokens, and a single special <unk> token was used for all remaining low frequency tokens. An <unk> token generated in the target sentence by the WORD and CHARCNN models was replaced with the source token associated with the maximum attention weight. The "open" vocabularies were only limited to the space of characters seen in training.

The sequence-to-sequence models were decoded with a beam size of 10.

4.4.4 BASELINE: STATISTICAL MACHINE TRANSLATION

As a baseline of comparison, we experiment with a phrase-based machine translation approach (SMT) shown to be state-of-the-art for the CoNLL-2014 shared task data in previous work (Junczys-Dowmunt and Grundkiewicz, 2016), which adds task specific features and the M^2 metric as a scorer to the Moses statistical machine translation system. The SMT model follows the training, parameters, and dense and sparse task-specific features that generate state-of-the-art results for CoNLL-2014 shared task data, as implemented in publicly available code.[9] However, to compare models against the same training data, we remove language model features associated with external data.[10] We experiment with tuning against M^2 ($+M^2$) and BLEU (+BLEU).

[9]SRI International provided access to SRILM (Stolcke, 2002) for running Junczys-Dowmunt and Grundkiewicz (2016)

[10]We found that including the features and data associated with the large language models of Junczys-Dowmunt and Grundkiewicz (2016), created from Common Crawl text filtered against the NUCLE corpus, *hurt* effectiveness for the phrase-based models. This is likely a reflection of the domain specific nature of the academic text and LaTeX holder symbols ap-

Models trained with diffs were only tuned with BLEU, since the tuning pipeline from previous work is not designed to handle removing such annotation tags prior to M^2 scoring.

Following the work of Junczys-Dowmunt and Grundkiewicz (2016), for dense features, we used the stateless edit distance features and the stateful Operation Sequence Model (OSM) of Durrani et al. (2013)[11]. Since for our controlled data experiments we removed the language model features associated with external data, we did not use the word-class language model feature, so for the sparse features, we used the set of edit operations on "words with left/right context of maximum length 1 on words" (set "E0C10" from the original paper), instead of those dependent on word classes.

The training and tuning splits for the phrase-based machine translation models were the same as for the sequence-to-sequence models. For tuning, we used Batch-Mira, setting the background corpus decay rate to 0.001, as in previous work. As in previous work, we repeated the tuning process multiple times (in this case, 5 times) and averaged the final weight vectors.

Decoding of the SMT models used the same approach of Junczys-Dowmunt and Grundkiewicz (2016) (i.e., the open-source Moses decoder run with the cube pruning search algorithm).

---

pearing in the text. Here, we conduct controlled experiments without introducing additional domain-specific monolingual data.

[11]The OSM features use the SRI Language Modeling Toolkit (SRILM) (Stolcke, 2002).

Replacement Error Type (out of 39,930) – Frequency relative to training							
Model	Punctuation	Articles	Other				
			> 100	$[5, 100]$	$[2, 5)$	1	0
Raw frequency in dev	11507	1691	6788	8974	2271	1620	7079
Number of unique instances	371	367	215	2918	1510	1242	5819
SMT+BLEU	56.03	16.41	44.57	36.17	39.46	31.93	0.00
Word+Bi	56.13	18.58	55.38	44.33	18.79	6.38	0.77
Word+Bi+dom	56.87	19.16	59.02	44.57	19.70	4.42	2.01
CharCNN+dom	55.64	13.37	57.34	41.83	28.99	16.74	7.09
Char+Bi	58.71	28.40	55.34	44.59	28.98	24.48	14.14
Char+Bi+dom	58.93	27.64	59.32	46.08	32.82	26.48	18.66

Table 4.2: Micro $F_{0.5}$ scores on replacement errors on the dev set. Errors are grouped by 'Punctuation', 'Article', and 'Other'. 'Other' errors are further broken down based on frequency buckets on the training set, with errors grouped by the frequency in which they occur in the training set.

4.5 Results and Analysis: AESW

Table 4.1 shows the full set of experimental results on the AESW development and test data.

The Char+Bi+dom model is stronger than the Word+Bi+dom and Char-

	Deletions	Insertions	Replacements
SMT+BLEU	46.56	31.48	42.21
Word+Bi	47.75	38.31	46.02
Word+Bi+dom	47.78	39.00	47.29
CharCNN+dom	48.30	39.57	46.24
Char+Bi	49.05	37.17	48.55
Char+Bi+dom	50.20	42.51	50.39

Table 4.3: Micro $F_{0.5}$ scores across error types

CNN+DOM models by 2.9 M^2 (0.2 GLEU) and 3.3 M^2 (0.3 GLEU), respectively. The sequence-to-sequence models were also more effective than the SMT models, as shown in Table 4.1. We find that training with target diffs is beneficial across all models, with an increase of about 5 M^2 points for the WORD+BI model, for example. Adding +DOM information slightly improves effectiveness across models.

We analyzed deletion, insertion, and replacement error types. Table 4.2 compares effectiveness across replacement errors. The seven columns of Table 4.2 are Micro $F_{0.5}$ scores for the errors within each frequency grouping. There are a total of 39,916 replacement changes. The replacements are grouped in regard to the changes within the opening and closing deletion tags and subsequent opening and closing insertion tags, as follows: (1) whether the replacement involves (on the deletion and/or insertion side) a single punctuation symbol (comma, colon, period, hyphen, apostrophe, quotation mark, semicolon, exclamation, question mark); (2) whether the replacement involves (on the deletion and/or insertion side) a single article (a, an, the); (3) non-article, non-punctuation grouped errors with frequency greater than 100 in the gold training data; (4) non-article, non-punctuation grouped errors with frequency less than or equal to 100 and greater than or equal to 5; (5) non-article, non-punctuation grouped errors with frequency less than 5 and greater than or equal to 2; (6) non-article, non-punctuation grouped errors with frequency equal to 1; (7) non-article, non-punctuation grouped errors that never occurred in the training data. Note that the large number of unique instances occurring for the "punctuation" and "articles"

classes are a result of the large number of errors that can occur on the non-article, non-punctuation side of the replacement. The Micro $F_{0.5}$ scores are calculated by treating each individual error (rather than the agglomerated classes here) as binary classifications.

We found the CharCNN+Bi models were less effective than CharCNN variants in terms of GLEU and M^2, and the strongest CharCNN models were eclipsed by the Word+Bi models in terms of the GLEU and M^2 scores. However, Table 4.2 shows CharCNN+dom is stronger on lower frequency replacements than Word models. The Char+Bi+dom model is relatively strong on article and punctuation replacements, as well as errors appearing with low frequency in the training set and overall across deletion and insertion error types, which are summarized in Table 4.3.

Errors never occurring in training

The comparatively high Micro $F_{0.5}$ score (18.66) for the Char+Bi+dom model on replacement errors (Table 4.2) never occurring in training is a result of a high precision (92.65) coupled with a low recall (4.45). This suggests some limited capacity to generalize to items not seen in training. A selectively chosen example is the replacement from "discontinous" to "discontinuous", which never occurs in training. However, similar errors of low edit distance also occur once in the dev set and never in training, but the Char+Bi+dom model never correctly recovers many of these errors, and many of the correctly recovered errors are minor changes in capitalization or hyphen-

ation.

Error frequency

About 39% of the AESW training sentences have errors, and of those sentences, on average, 2.4 words are involved in changes in deletions, insertions, or replacements (i.e., the count of words occurring between diff tags) per sentence. In the NUCLE data, about 37% of the sentences have errors, of which on average, 5.3 words are involved in changes. On the AESW dev set, if we only consider the 9545 sentences in which 4 or more words are involved in a change (average of 5.8 words in changes per sentence), the Char+Bi model is still more effective than SMT+BLEU, with a GLEU score of 67.21 vs. 65.34. The baseline GLEU score (No Change) is 60.86, reflecting the greater number of changes relative to the full dataset (cf. Table 4.1).

Re-annotation

The AESW dataset only provides 1 annotation for each sentence, so we perform a small re-annotation of the data to gauge effectiveness in the presence of multiple annotations. We collected 3 outputs (source, gold, and generated sentences from the Char+Bi+dom model) for 200 randomly sampled sentences, re-annotating to create 3 new references for each sentence. The GLEU scores for the 200 original source, Char+Bi+dom, and original gold sentences evaluated against the 3 new references were 79.79, 81.72, and 84.78, respectively, suggesting that there is still progress to be

	Precision	Recall	$F_{0.5}$
WORD+BI–DIFFS	65.36	6.19	22.45
WORD+BI, before tuning	72.34	0.97	4.60
WORD+BI, after tuning	46.66	15.35	33.14

Table 4.4: M^2 scores on the CoNLL-2013 set.

made on the task relative to human levels of annotation.

4.6 Results and Analysis: CoNLL

Table 4.4 shows the results on the CoNLL dev set, and Table 4.6 contains the final test results.

Since the CoNLL data does not contain enough data for training neural models, previous works add the crowd-sourced Lang-8 data; however, this data is not professionally annotated. Since the distribution of corrections differs between the dev/test and training sets, we need to tune the precision and recall.

As shown in Table 4.4, WORD+BI effectiveness increases significantly by tuning the weights[12] assigned to the diff tags on the CoNLL-2013 set[13]. Note that we are tuning the weights on this same CoNLL-2013 set. Without tuning, the model very rarely generates a change, albeit with a high precision. After tuning, it exceeds the effectiveness of WORD+BI–DIFFS. For tuning on the dev set, a coarse grid search between 0 and 1.0 was used to set the four bias parameters associated with each diff tag.

[12]In contrast, in early experiments on AESW, tuning yielded negligible improvements.

[13]The single model with highest M^2 score was then run on the test set. Here, a single set is used for tuning and dev.

Bias parameter	Precision	Recall	$F_{0.5}$
0.0	72.34	0.97	4.60
0.1	69.74	1.51	6.96
0.2	72.00	2.57	11.23
0.3	69.05	4.14	16.68
0.4	67.19	6.08	22.31
0.5	61.03	8.76	27.82
0.6	51.75	11.41	30.31
0.7	**46.66**	**15.35**	**33.14**
0.8	40.01	18.68	32.57
0.9	34.49	22.08	31.00
1.0	30.17	24.90	28.94

Table 4.5: M^2 scores on the CoNLL-2013 dev set for the WORD+BI model.

(Training was performed without re-weighting.) The bias parameter (in this case 0.7) yielding the highest M^2 score on the decoded dev set was chosen for use in evaluation of the final test set. The M^2 scores across the tuning runs on the dev set for the WORD+BI model are shown in Table 4.5.

The comparatively low effectiveness of WORD+BI–DIFFS seen in Table 4.4 is consistent with past sequence-to-sequence approaches utilizing data augmentation, additional annotated data, and/or secondary models to achieve competitive levels of effectiveness.

Table 4.6 shows that WORD+BI is within 0.2 M^2 of Ji et al. (2017), despite using over 1 million fewer sentence pairs, and exceeds the M^2 scores of Xie et al. (2016) and Yuan and Briscoe (2016) without the secondary models of those systems. We hypothesize that further gains are possible utilizing the CLC data and moving to the

	Data	M^2
Yuan and Briscoe (2016)	CLC*	39.90
Xie et al. (2016)	NUCLE, Lang-8, Common Crawl LM	40.56
Ji et al. (2017)	NUCLE, Lang-8, CLC*	41.53
Word+Bi–diffs	NUCLE, Lang-8	35.73
Word+Bi	NUCLE, Lang-8	41.37

Table 4.6: M^2 scores on the CoNLL-2014 test set and data used for recent sequence-to-sequence based systems. Results for previous works are those reported by the original authors. *CLC is proprietary.

character model. (The character model is omitted here due to the long training time of about 4 weeks.) Notably, SMT systems (with LMs) are still more effective than reported sequence-to-sequence results, as in Ji et al. (2017), on CoNLL.[14]

4.7 Conclusion: Adapting Sequence Models for Sentence Correction

Our experiments demonstrate that on a large, professionally annotated dataset, a sequence-to-sequence character-based model of diffs can lead to considerable effectiveness gains over a state-of-the-art SMT system with task-specific features, ceteris

[14]For reference, the reported M^2 results of the carefully optimized SMT system of Junczys-Dowmunt and Grundkiewicz (2016) trained on NUCLE and Lang-8, with parameter vectors averaged over multiple runs, with a Wikipedia LM is 45.95 and adding a Common Crawl LM is 49.49. We leave to future work the intersection of a LM for the CoNLL environment and more generally, whether these patterns hold in the presence of additional monolingual data.

paribus. Furthermore, in the crowd-sourced environment of the CoNLL data, in which there are comparatively few professionally annotated sentences in training, modeling diffs enables a means of tuning that improves the effectiveness of sequence-to-sequence models for the task.

Acknowledgments

This chapter is based on Schmaltz et al. (2017). We thank the researchers, engineers, and staff of Rakuten for feedback, advice, and computing support.

5

Error Ordering

Translation models for grammatical error tasks such as correction and identification, as presented in the previous chapters, depend on large amounts of hand-annotated data. For many languages and domains, such annotations do not exist and are expensive to acquire. We propose a data augmentation approach for generating synthetic data using language models trained on non-parallel, unannotated data and rules de-

rived from a comparatively limited amount of annotated data. We demonstrate that an existing translation model can be trained with synthetic data to approach the levels of effectiveness of models trained on substantially more human annotated sentences, and we show that augmenting translation models is more effective than existing classifier-based approaches for correction alone given comparable amounts of annotated correction data.[1]

5.1 Introduction

The task of sentence correction, the transduction of a sentence that has zero or more errors into one with zero errors, was introduced in Chapter 4 and is one version of the more general task of language correction on which there has been steady progress in the last few years.

Recent work has adapted sequence-to-sequence and phrase-based translation models for the task (Junczys-Dowmunt and Grundkiewicz, 2016; Yuan and Briscoe, 2016; Xie et al., 2016; Ji et al., 2017; Schmaltz et al., 2017; Sakaguchi et al., 2017; inter alia). However, all of these models were trained on hundreds of thousands or even millions of annotated sentences.

We seek an approach that may be useful for creating correction systems for languages other than English, for which only small amounts (if any) of hand-annotated

[1]Note that unlike the other chapters of this dissertation, the material in this chapter is unpublished and is thus considered preliminary.

correction datasets are currently available. Furthermore, many of the professionally annotated datasets that are available, including in English, have licensing restrictions and cover only a limited number of domains. Thus, while translation models have demonstrated promising results on the task, the fully supervised case with millions of annotated sentences is not realistic for most settings.

We propose a data augmentation method for generating synthetic data (e.g., as in Table 5.1) for use with existing translation models when only given access to very limited amounts of annotated data.[2] We extend recent insights from the related task of word ordering (Hasler et al., 2017), casting synthetic data generation as the search for the highest scoring ordered sequence of copy, deletion, insertion, and replacement operations that transform a well-formed sentence into a version with zero or more errors. We do so via an adaptation of beam search, using a language model (LM) over well-formed sentences and a second, distinct LM over a non-parallel, disjoint corpus of sentences with errors. We combine these with a small number of simple rules derived from a limited number of annotated sentences.

In this work, we hold two recent, existing translation models constant, training with synthetic data and limited amounts of labeled data. We demonstrate that at scale, our method can be used to train a translation model that approaches a comparable model trained with additional human annotated data, and we show that data

[2]Generating synthetic data also has potential standalone utility for creating unique language exercises, such as for online testing, and perhaps as a means of bootstrapping a crowdsourcing system (e.g., as a means of testing would-be crowd annotators).

augmentation for translation models is more effective than previously proposed correction models that use limited annotations.

At scale, the approach can be used to train a sequence-to-sequence model that approaches a comparable model trained with access to additional human annotated data. For example, we show that a synthetic dataset of 2.8 million sentences derived from (and also including, in training) 3,688 professionally annotated sentences (of which 1,000 have errors) can be used to train a system, tuned on the standard CoNLL-2013 data (1,381 human annotated sentences), that approaches the effectiveness (within 1 M^2) of a model trained on the standard 57 thousand CoNLL-2014 annotated sentences combined with 200 thousand crowd-sourced annotated sentences. We also compare to a best-of-class classifier based approach, which can also correct from limited annotations, demonstrating superiority when given access to similar amounts of annotated data, while not relying on external linguistic annotations and tools, such as syntactic parsers and taggers.

5.2 Task: Sentence Correction and Sentence Corruption

The task of *sentence correction* is to convert a natural language sentence that may or may not have errors into a corrected version, and it was introduced in Chapter 4. We briefly review the basic setup here. We assume a vocabulary $\mathcal{V}$ of natural language word types. Given a sentence $\mathbf{s} = [s_1 \cdots s_I]$, where $s_i \in \mathcal{V}$ is the i-th token of the

O	Clinton emphasized that both the western allies and Russia agree on the goal of halting bloodshed in Kosovo and restoring the political autonomy that Kosovo long enjoyed from Belgrade .
R	Clinton emphasized that both the western allies ~~and~~ ant Russia agree on the goal of halting bloodshed in Kosovo and restoring the political autonomy that Kosovo long enjoyed from Belgrade ~~.~~ ;
S	Clinton emphasized that both the western allies and Russia agree on the goal of halting bloodshed ~~in~~ Kosovo and restoring the political autonomy that Kosovo have long enjoyed from Belgrade .
O	Historically , Thailand 's southern provinces have closer ties with northern Malaysia , just across the river , than with Thailand 's distant capital .
R	Historically , Thailand 's southern provinces have closer ties with northern Malaysia , just across the river , than with Thailand 's distant ~~capital~~ Capital .
S	Historically , Thailand 's southern provinces have closer ties with northern Malaysia , just across the river ~~,~~ than with Thailand 's distant capital .
O	Still , DeMauro said that when contractors began sawing the tree , she almost lost her nerve .
R	Still , DeMauro said that ~~when~~ When contractors began sawing the tree , she revolutionize almost lost her nerve .
S	Still , DeMauro said ~~that~~ when contractors began sawing the tree , she almost lost her nerve .

Table 5.1: First three random samples of synthetic error data drawn from the training set, generated via CORRUPTION+RANDOM (R) and CORRUPTION+SEARCH (S), including the original, correct sentences (O). Deletions are marked with a red strikethrough and insertions with a blue underline.

sentence of length I, we seek to predict the target sentence $\mathbf{t} = [t_1 \cdots t_J]$, where $t_j \in \mathcal{V}$ is the j-th token of the corrected sentence of length J. The target sentence $\mathbf{t}$ is the "corrected" version of $\mathbf{s}$ according to stylistic and grammatical conventions defined, for the purposes here, by the available annotated data.

In the standard correction setup, we are given pairs of $\mathbf{s}$ and $\mathbf{t}$ sentences from an aligned dataset for supervised training. At test time, we are only given access to a new, unseen sequence $\mathbf{s}$. We learn to predict sequence $\mathbf{t}$ (which can be a copy of $\mathbf{s}$ without modifications).

We define *sentence corruption* as the task of converting, with access to a limited amount of aligned data, a correct natural language sentence into a version with zero or more errors. We seek to predict the sentence $\mathbf{s}$ given the sentence $\mathbf{t}$. As with correction, there are multiple ways to corrupt a sentence. In practice, we seek corruptions that match the distribution of human errors in available annotated datasets.

In the corruption setup, we are given instances of $\mathbf{s}$ sentences from the dataset $\mathbf{C^{(s)}}$ and instances of $\mathbf{t}$ sentences from the dataset $\mathbf{D^{(t)}}$. $\mathbf{C^{(s)}}$ and $\mathbf{D^{(t)}}$ are disjoint. We are also given pairs of $\mathbf{s}$ and $\mathbf{t}$ sentences from the aligned dataset $\mathbf{A^{(s,t)}}$, where $|\mathbf{A^{(s,t)}}| \ll |\mathbf{C^{(s)}}|$ and $|\mathbf{A^{(s,t)}}| \ll |\mathbf{D^{(t)}}|$. At test time, we are only given access to new, unseen sequences $\mathbf{t}$, and we learn to predict the corresponding sequences $\mathbf{s}$.

5.3 Related work

Most recent work on generating errors has itself depended on large amounts of annotated data, using back-translation and extracting error patterns from annotated text (Rei et al., 2017), as well as generating confusion sets for particular types of errors from annotated data (Bowen et al., 2017; Cahill et al., 2013; Rozovskaya et al., 2012). We surmise (but leave to future investigation) that a combination of back-translation by reversing the approach of Schmaltz et al. (2017) or Junczys-Dowmunt and Grundkiewicz (2016) and the beam search approach here to introduce new errors may benefit an approach already given large amounts of annotated data; however, we are primarily interested in the case where there is a very limited amount of annotated data to begin with.

Prior to the availability of labeled data with recent shared tasks and the resultant focus on translation-based approaches, classifier-based approaches were a dominant paradigm for correction. The contemporaneous work of Rozovskaya et al. (2017) provides an overview of this line of work and extends the confusion set and classifier based approaches of generating hand-chosen subsets of artificial errors, using large amounts of native text and hand-tuned feature sets, to a version with reliance on only a limited amount of annotated data. Our classifier-based comparison is with the CoNLL-2014 model that appears in this most recent work.

In an earlier generation of work, a discriminative beam search approach was pro-

posed by Dahlmeier and Ng (2012a). That beam search approach was applied directly to correction, was discriminatively trained, and also differs significantly in the approach for scoring hypotheses presented here. Recent translation-based approaches are extensions of the work of Brockett et al. (2006). In this early work, synthetic data was created using a dictionary and hand-written regular expressions based on patterns observed in the Chinese Learner Error Corpus (CLEC). That work, a pilot study, only generated synthetic data for mass noun errors, whereas our approach generates corruptions for a much larger set of possible errors.

When large amounts of labeled data are available, as made available in recent years for English via the CoNLL-2014 shared task (Ng et al., 2014), translation-based approaches have been shown to be state-of-the-art on the task, as noted in previous chapters. Competitive results have been demonstrated by adapting both phrase-based systems (Junczys-Dowmunt and Grundkiewicz, 2016; Chollampatt et al., 2016b,a; Hoang et al., 2016; Napoles and Callison-Burch, 2017) and sequence-to-sequence systems (Yuan and Briscoe, 2016; Xie et al., 2016; Ji et al., 2017; Schmaltz et al., 2017; Sakaguchi et al., 2017) to the task, and an ensemble of methods can be helpful (Yannakoudakis et al., 2017; Chollampatt and Ng, 2017). In this chapter, we consider both a sequence-to-sequence model and a phrase-based model for data augmentation.

5.4 Methods

We propose an approach for sentence corruption that we then use to generate training data for correction models.

Here, we describe our approach for sentence corruption, and then the details of our particular implementation appear in the Experiments section.

Given input $\mathbf{t}$, we aim to generate $\mathbf{s}$ by introducing zero or more errors into $\mathbf{t}$. For example, in the first sentence of Table 5.1, the preposition "in" is deleted and the verb "have" is inserted. The transduction from $\mathbf{t}$ to $\mathbf{s}$ is described in terms of *edit operations*, which for the purposes here, are functions that take a single token and return zero or more tokens. Each token of $\mathbf{t}$ is associated with one edit operation, and the ordered, left-to-right application of the edit operations associated with the tokens of $\mathbf{t}$ produces $\mathbf{s}$. Our goal is to assign one edit operation to each token of $\mathbf{t}$. We define the output set $\mathcal{Y}(\mathbf{t})$ to consist of all considered sequences of edit operations over the J tokens of $\mathbf{t}$ that generate the corrupted sentence $\mathbf{s} = [y_1(t_1) \cdots y_J(t_J)]$.

We aim to find $\hat{\mathbf{s}} \in \mathcal{Y}(\mathbf{t})$, which is the ground-truth corrupted sentence, or a sentence similar to it. We generate the corrupted sentence by approximately optimizing a scoring function f over the set of sequences of edit operations that generate $\mathbf{s}$, $\mathbf{s}^* = \arg\max_{\mathbf{s} \in \mathcal{Y}(\mathbf{t})} f(\mathbf{t}, \mathbf{s})$.

The scoring function has access to a model over poorly written sentences, as in $\mathbf{C}^{(\mathbf{s})}$, and a model over well-formed sentences, as in $\mathbf{D}^{(\mathbf{t})}$. In our notation, $p^{(s)}(\mathbf{s})$ is

the probability of the sentence $\mathbf{s}$ under the former model, and $p^{(t)}(\mathbf{s})$ and $p^{(t)}(\mathbf{t})$ are the probabilities of sentences $\mathbf{s}$ and $\mathbf{t}$, respectively, under the latter model.

We define the scoring function as

$$f(\mathbf{t}, \mathbf{s}) = \begin{cases} p^{(s)}(\mathbf{s}), & \text{if } p^{(t)}(\mathbf{s}) < p^{(t)}(\mathbf{t}) \\ 0, & \text{otherwise} \end{cases}$$

Conceptually, we seek the most likely sentence according to a model of poorly written sentences that is less likely than the unchanged sentence according to a model of well-written sentences. To model the probabilities over the sentences, we use two distinct LMs for well-written sentences and poorly written sentences, as described further below, with implementation details in the Experiments section.

5.4.1 Search

We adapt beam search to our new task of searching for the highest scoring sequence of edit operations according to f. This is inspired by recent work that suggests LSTM LMs, with certain search heuristics, can approach the effectiveness of other sequence-based models for the related generation task of word ordering (Hasler et al., 2017). Our approach thus resembles the approach for LM-based word ordering via beam search appearing in Chapter 1.

We use the following four edit operations, making a distinction among operations in

order to control their individual frequencies and the tokens on which they are allowed to operate, as detailed in the Experiments section:

- COPY, which takes a single token and returns the same token;
- DELETION, which takes a single token and returns a null token;
- INSERTION, which takes a single token and returns a new token followed by a copy of the argument token;
- REPLACEMENT, which takes a single token and returns a new token.

Since INSERTION and REPLACEMENT perform single token insertions, which is a decision made to limit the search space, these operations cannot fully express all possible transformations from $\mathbf{t}$ to $\mathbf{s}$.

Beam search maintains $J+1$ beams, $B_0, \ldots, B_J$, each containing at most the top-K partial hypotheses of that length. A partial hypothesis is a 6-tuple $(\mathbf{y}, \boldsymbol{h}^{(s)}, \boldsymbol{h}^{(t)}, \pi^{(s)}, \pi^{(t)}, \chi)$, where at j, the current index into $\mathbf{t}$,

- $\mathbf{y}$ is a sequence of edit operations, $\mathbf{y} = [y_1 \cdots y_J]$, of length J, initially filled with COPY operations;
- $\boldsymbol{h}^{(s)}$ and $\boldsymbol{h}^{(t)}$ are the current LM states;
- $\pi^{(s)} = p^{(s)}([y_1(t_1) \cdots y_j(t_j)])$, the LM probability of the sentence prefix;
- $\pi^{(t)} = p^{(t)}([y_1(t_1) \cdots y_j(t_j)])$, analogous to the above; and
- χ is the score of the partial hypothesis.

In our notation for the LMs, $\boldsymbol{h} \in \mathcal{H}$ is the state of an LM, initialized with $\boldsymbol{h}_0$. The state of an LM is advanced when seeing a word $w_i \in \mathcal{V}$ via its update function, $\boldsymbol{h}_i = \delta(w_i, \boldsymbol{h}_{i-1})$. An LM can be queried for an estimate of the probability of the next word

Algorithm 2 Dual LM beam-search corruption (CORRUPTION+SEARCH)

procedure CORRUPTION+SEARCH($t_1 \dots t_J$, K, $\mathcal{Y}(\mathbf{t})$)
  $B_0 \leftarrow \langle([\text{COPY}_1 \dots \text{COPY}_J], \boldsymbol{h}_0^{(s)}, \boldsymbol{h}_0^{(t)}, 0, 0, f(\mathbf{t}, \mathbf{t}))\rangle$
  for $j = 0, \dots, J-1$ **do**
    $j' \leftarrow j + 1$
    for $k = 1, \dots, |B_j|$ **do**
      ▷ $y_{j'}^{prev}, \dots, y_J^{prev}$ are always COPY operations:
      $(\mathbf{y}^{prev}, \_, \_, \_, \_, \_) \leftarrow B_j^{(k)}$
      ▷ Loop through allowed edits:
      for $\mathbf{y}^{new} \in \mathcal{Y}(\mathbf{t})$, **where** $\mathbf{y}^{prev}$ and $\mathbf{y}^{new}$ are equal or only differ at index j' **do**
        $(\mathbf{y}', \boldsymbol{h}^{(s)'}, \boldsymbol{h}^{(t)'}, \pi^{(s)'}, \pi^{(t)'}, \chi') \leftarrow B_j^{(k)}$
        ▷ Update the edit operation at index j':
        $\mathbf{y}' \leftarrow [y'_1, \dots, y'_{j'-1}, y_{j'}^{new}, y'_{j'+1}, \dots, y'_J]$
        Update $\boldsymbol{h}^{(s)'}, \boldsymbol{h}^{(t)'}, \pi^{(s)'}, \pi^{(t)'}$ with ADVANCELMS
        $\chi' \leftarrow f(\mathbf{t}, \mathbf{y}'(\mathbf{t}))$
        if $\mathbf{t} = \mathbf{y}'(\mathbf{t})$ **then**
          $\chi' \leftarrow 1$
        if $\chi' > 0$ **then**
          $B_{j'} \leftarrow B_{j'} + (\mathbf{y}', \boldsymbol{h}^{(s)'}, \boldsymbol{h}^{(t)'}, \pi^{(s)'}, \pi^{(t)'}, \chi')$
          keep unique top-K of $B_{j'}$ by χ
  return B_J
procedure ADVANCELMS($w, \boldsymbol{h}^{(s)}, \boldsymbol{h}^{(t)}, \pi^{(s)}, \pi^{(t)}$)
  for $x \in \{s, t\}$ **do**
    $\pi^{(x)} \leftarrow \pi^{(x)} \cdot q^{(x)}(w, \boldsymbol{h}^{(x)})$
    $\boldsymbol{h}^{(x)} \leftarrow \delta^{(x)}(w, \boldsymbol{h}^{(x)})$
  return $\boldsymbol{h}^{(s)}, \boldsymbol{h}^{(t)}, \pi^{(s)}, \pi^{(t)}$

$q(w_i, \boldsymbol{h}_{i-1}) \approx p(w_i \mid w_1, \ldots, w_{i-1})$. $\mathcal{H}$, δ, and q are unique to each of the two LMs. Superscripts distinguish the identity of the LMs: $\mathcal{H}^{(s)}$, $\delta^{(s)}$, and $q^{(s)}$ are for the LM over the poorly written sentences, and $\mathcal{H}^{(t)}$, $\delta^{(t)}$, and $q^{(t)}$ are for the LM over the well-formed sentences.

The full beam search is given in Algorithm 2, which we refer to as CORRUPTION+SEARCH. Note that we perform hypothesis recombination, preventing duplicate hypotheses from occupying multiple beam positions. One hypothesis representing the unchanged sentence, $\mathbf{t} = \mathbf{y}(\mathbf{t})$, always occupies one position in the beam, and in practice, that is the only hypothesis with $f(\mathbf{t}, \mathbf{s}) = 0$ retained on the beam.

We use a future cost heuristic that consists of the probability of the remaining, unchanged tokens in $\mathbf{t}$, $[t_{j+1} \cdots t_J]$, under the applicable LMs, at index j. (Recall that $\mathbf{y}$ is initialized as a sequence of COPY operations of length J.) When calculating $f(\mathbf{t}, \mathbf{s})$, we assume that the remaining tokens of $\mathbf{s}$ are copies of the uncovered tokens of $\mathbf{t}$. This involves running the LSTM LMs forward to the end of the sequences at each beam step (and then discarding the final states), but we find that it is feasible (albeit still slow) in practice by batching the hypotheses and cacheing the prefix states $(\boldsymbol{h}^{(s)}, \boldsymbol{h}^{(t)})$ and prefix scores $(\pi^{(s)}, \pi^{(t)})$ up to the current index, j. The score $p^{(t)}(\mathbf{t})$, used in calculating f, is run once per sentence and cached.

5.5 Experiments

5.5.1 Data

The standard CoNLL-2014 shared task data, the focus of most previous sentence correction work, consists of the National University of Singapore (NUS) Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013), a training set of 56,560 professionally annotated sentences after preprocessing. This data set is often combined with the publicly available Lang-8 data (Tajiri et al., 2012; Mizumoto et al., 2012), which consists of about 1.4 million crowd-source sentences after filtering. In our experiments, the professionally annotated sentences are drawn from the NUCLE dataset and the crowd-sourced annotated sentences are drawn from the Lang-8 data. Final testing of all models is on the CoNLL-2014 test set (without alternatives).

For the LMs of Corruption+Search, we use the source side of the NUCLE corpus combined with the source side of the Lang-8 data for the model over poorly written sentences, $\mathbf{D}^{(\mathbf{s})}$. For the model over well-written sentences, $\mathbf{D}^{(\mathbf{t})}$, we use a sample of 1,470,992 sentences from the New York Times (NYT) section of the Annotated Gigaword corpus (Napoles et al., 2012). For generating synthetic data, we set aside a final disjoint set of 1,414,432 sentences from the NYT section, of which we use samples of various sizes. The aligned dataset of $\mathbf{A}^{(\mathbf{s},\mathbf{t})}$ is a sample of 3,688 sentences from NUCLE, which amounts to 1,000 sentences with annotated errors.

Num. synthetic sent.	Num. human annotated sent. (professional)	Num. human annotated sent. (crowdsourced)	Total sent.	Sent. with errors	Num. unique *corrected* **t** sent.	Num. unique tokens (source | target)	$\|\mathcal{V}\|$	M^2 dev	M^2 test
0	57k	0	57k	37%	21k	33k | 34k	30k	18.38	16.06
0	57k	200k	257k	60%	153k	81k | 89k	50k	24.64	25.12
0	57k	250k	307k	61%	185k	89k | 98k	50k	26.32	28.83
2,768k	57k	0	2,825k	53%	577k	437k | 438k	50k	26.27	27.55
2,768k	57k	0	2,825k	53%	577k	437k | 438k	100k	28.02	30.47
2,768k	4k	0	2,772k	53%	557k	428k | 428k	50k	20.73	24.34
2,768k	4k	0	2,772k	53%	557k	428k | 428k	100k	21.26	28.17
3,129k	4k	0	3,133k	59%	692k	428k | 428k	50k	21.78	24.65
3,129k	4k	0	3,133k	59%	692k	428k | 428k	100k	22.40	27.69
0	57k	1,414k	1,471k	65%	946k	195k | 226k	50k	33.14*	41.37*

Table 5.2: M^2 scores on the CoNLL-2014 dev and test set and training data used for the sequence-to-sequence model. $|\mathcal{V}|$ is the size of the vocabulary used to train the sequence-to-sequence model, independently for source and target. The sixth column is the number of unique target sequences (after removing diff tags) in which the target differs from the source. *The final line is that of the WORD+BI model of Schmaltz et al. (2017). *All models are given access to an additional 1,381 human annotated sentences (i.e., the CoNLL-2013 set) for tuning.*

5.5.2 CORRUPTION

LMs

We use standard long short-term memory network (LSTM) LMs (Graves, 2013; Zaremba et al., 2014). Our two LSTM models are identical except for the data used to train and tune them, and they are based on that of Zaremba et al. (2014). We use a hidden state size of 1,200, unrolling the network for 79 steps. Both LMs use the same vocabulary of 50,000 tokens derived from the $\mathbf{C}^{(\mathbf{s})}$ data.

Corruption set

We restrict the sequences of edit operations under consideration in $\mathcal{Y}(\mathbf{t})$ using a series of rules. These rules, as described further below, reduce the search space and also encourage corruptions closer to those observed in the small parallel data of $\mathbf{A}^{(\mathbf{s},\mathbf{t})}$, which is used for tuning. In general, we aim for rules that are simple, based on broad (largely frequency and edit-distance-based) patterns observed in the parallel $\mathbf{A}^{(\mathbf{s},\mathbf{t})}$ data. Since some of the rules are stochastic, Corruption+Search can be viewed as a hybrid between standard beam search and a sequential Monte Carlo method, with parameters set by the $\mathbf{A}^{(\mathbf{s},\mathbf{t})}$ data. Our proposed approach is semi-supervised as these rules are predicated on at least a small amount of annotated data.

These rules are as follows:

- The maximum number of deletion, insertion, and replacment operations in $\mathcal{Y}(\mathbf{t})$ under consideration for any sentence are set by drawing from a distribution over frequencies of these operations in the $\mathbf{A}^{(\mathbf{s},\mathbf{t})}$ data.
- A token is considered for deletion by drawing from a distribution with a mean based on the frequency in the $\mathbf{C}^{(\mathbf{s})}$ data. We bucket the frequencies, with the mean for the 100 most frequent words at 1 (i.e., always considered), the next 100 at 0.1, the next 1000 at 0.05, and the remaining at 0.01. The following additional restrictions are made as a broad brush to avoid major semantic changes rarely occurring in the $\mathbf{A}^{(\mathbf{s},\mathbf{t})}$ data: The means for tokens with digits and/or one or more uppercase letters, as well as for the tokens "not", "n't", and "no", are set to 0. Insertions are considered at indices not considered for deletion (i.e., insertions are more likely to be considered before less frequent words). The actual token to be inserted is based on the same probability cutoffs as for deletion (i.e., more frequent words are more likely to be inserted, and insertions of tokens

with digits, uppercase letters, and “not”, “n’t”, and “no” are not allowed).

- Replacements in which a token with a digit is deleted are not allowed. Replacements that have a character Levenshtein edit distance of 1 are otherwise allowed. Replacements that have a Levenshtein distance of 2 or 3 are allowed, provided that the first 4 characters match and the token to be deleted does not contain an uppercase letter.

Otherwise unbridled generation using the above generated some replacement changes yielding obvious, major semantic changes (for example, changing “she” to “he”, which constitutes a Levenshtein distance of 1). As such, we perform a final filtering of replacement changes, removing non-capitalization changes in which the character-level diff between the token to be deleted and the token to be inserted consists of a final deletion, insertion, or replacement that occurs in the $\mathbf{A}^{(\mathbf{s},\mathbf{t})}$ data.

Search

We use a beam size of 5. Since the unchanged sentence is always retained on the beam, this corresponds to generating up to 4 distinct $\mathbf{s}$ sentences. Given the relatively slow generation time, we include all final hypotheses as synthetic examples. We also include the corresponding unchanged $\mathbf{t}$ sentences as training examples in the synthetic datasets.

Ablation

We aim to determine whether using the probabilities from the LMs is necessary, or whether it is sufficient to just sample the errors using the rules of the corruption set, as described above. To do so, we run Algorithm 2 without the LMs, with decisions otherwise determined by the LMs (i.e., by $f(\mathbf{t}, \mathbf{s})$) made at random. We refer to this ablation as Corruption+Random.

Performance

For generating synthetic data via Corruption+Search, we used two GPUs in practice, placing each language model on a separate 12 GB card. The LM states are reset after each sentence, so decoding is embarrassingly parallel. For this work, we corrupted about 10,000-20,000 sequences ($\mathbf{t}$ sentences) per day per GPU pair. The relatively slow generation time is the result of calculating the full sentence probabilities of $f(\mathbf{t}, \mathbf{s})$. We found that using the full sentence probabilities was helpful, and improving the efficiency of the approach is the focus of future work.

Evaluation

Our primary means of evaluating the quality of the synthetic data is via the effectiveness of the downstream correction model. We use the primary correction evaluation metric appearing in previous work for the CoNLL-2014 task. This MaxMatch (M^2)

metric (Dahlmeier and Ng, 2012b) is an $F_{0.5}$ score over the phrase-level edits between the generated output and references.

5.5.3 Correction

We generate synthetic data for training two translation models appearing in the recent literature. Our sequence-to-sequence model is that introduced in Chapter 4, a bidirectional model modeling the differences between source and target via diff tags on the target side. We perform tuning on the revised CoNLL-2013 test set. Our SMT model and tuning approach are identical to that of Junczys-Dowmunt and Grundkiewicz (2016), a phrase-based model with task specific features, using the publicly available code and Common Crawl LM.

An alternative approach for correction when little or no annotated data is available is to build individual classifiers for particular types of errors. Rozovskaya and Roth (2016) is a recent best-of-class[3] approach that uses Naive Bayes and Averaged Perceptron classifiers, a separate spell checker, and rules. Syntactic parsers and taggers are used, and the Web1T corpus is used for native data. To investigate the limited annotation scenario, we compare against the approach when only given access to the NUCLE data for annotations (additional gains are possible by adding the Lang-8

[3]To put it in perspective, as recently as ACL 2016, the classifier approach of Rozovskaya and Roth (2016), using only the NUCLE data for annotations, would have been the state-of-the-art on the task, including against approaches using significantly more annotations (cf. Table 19 in Rozovskaya et al. (2017)).

	Precision	Recall	$F_{0.5}$
Corruption+Random	29.84	14.04	24.36
Corruption+Search	39.32	13.20	28.17

Table 5.3: M^2 scores on the CoNLL-2014 test set with the sequence-to-sequence model. 2,768k synthetic training sentences are combined with 4k human annotated sentences, $|\mathcal{V}| = 100$k.

data via model combination, as with an SMT system). Since the classifier approach makes use of a large amount of external data, the primary comparison is with the SMT model with a large language model. We follow past work (Junczys-Dowmunt and Grundkiewicz, 2016; Rozovskaya and Roth, 2016) in treating the Web1T corpus and the Common Crawl dataset as comparable web-scale monolingual sources.

5.6 Results

We empirically analyze the effectiveness of the downstream correction models using the generated synthetic data, and we also provide a qualitative overview with examples of the generated data.

First, we analyze the contribution of the LMs by comparing Corruption+Random and Corruption+Search. Table 5.3 shows a comparison of results with the sequence-to-sequence model when generating synthetic data via these two approaches. Recall that Corruption+Random does not use the LMs but otherwise uses the same rules to restrict the allowed edit operations. We see that while recall is actually slightly stronger with the baseline, there is a precipitous loss in precision, presumably due

to generating changes that are in the tails of the distribution, contributing to an M^2 score of nearly 4 points lower than when using the LMs.

Table 5.1 contains sentences corresponding to the first three indices, among sentences with errors, randomly drawn from the training set. This small, random sample reflects certain behavior seen in reviewing additional samples. In the first sentence, we see that CORRUPTION+RANDOM replaces "and" with "ant". This is an allowed REPLACEMENT, and the ablated approach choses randomly among this and other variants, such as "And" and "an", among others. This replacement occurs 7,170 times in the baseline generated data; however, it never occurs in the data generated with the LMs, nor does it ever occur in $\mathbf{A}^{(\mathbf{s},\mathbf{t})}$. Another example of the baseline generating a change that is possibly an unlikely human error is the insertion of "revolutionize" in the third example, which highlights the challenge of inserting without the LM context. In addition to generating changes in the tails of the distribution of human errors, we have seen examples of both CORRUPTION+RANDOM and CORRUPTION+SEARCH generating sentences that are different but not necessarily grammatically incorrect, as shown in the final sentence when CORRUPTION+SEARCH deletes "that". Reliably identifying both problem cases (unlikely changes and changes that are unnecessary) is challenging, but may improve effectiveness and is the subject of future work.

Table 5.2 shows results for the sequence-to-sequence model when varying the amount of synthetic data (via CORRUPTION+SEARCH) and professionally and crowd-sourced annotated data. The final line contains the M^2 score reported in the work of Schmaltz

et al. (2017), which is significantly higher than all of the results using the synthetic data, but it uses significantly more labeled data. At the other end, as one might expect, only training a sequence-to-sequence model with the 57k lines of the NUCLE data results in very poor scores, as shown in the first line.

We see that training the sequence-to-sequence model with 2,768k synthetic sentences and only 4k annotated sentences is competitive with the same model trained on NUCLE and about 200k crowd-source annotated sentences. We see a noticeable jump in effectiveness when doubling the vocabulary size to 100k, made possible by the large vocabulary of the NYT section of Gigaword. In practice, a system would benefit from the larger vocabulary, but we also include the 50k vocabulary results for comparison purposes owing to the smaller vocabulary sizes of the crowd-sourced data. Effectiveness leveled-off by 3,129k synthetic sentences, suggesting the limits of the edit operations of the current implementation.

Table 5.4 compares with established correction approaches that make use of large amounts of monolingual data. As with the sequence-to-sequence results, the M^2 score of a strong SMT system with over a million human annotations is greater than that of a system with much smaller amounts of annotated data combined with synthetic data. However, our approach makes progress in the limited annotation setting. We see that the classifier approach of Rozovskaya and Roth (2016) alone exceeds that of the SMT model only given the NUCLE annotations and a LM. However, when adding 2,772k synthetic sentences (derived from 4k annotations), the M^2 score im-

Model	Labeled Data	Additional Data	M^2
RR16	NUCLE (57k)	Web1T, linguistic tools	43.11
SMT	NUCLE (57k)	CommonCrawl	42.14
SMT	NUCLE (57k)	CommonCrawl, 2,772k synthetic	44.58
SMT	NUCLE (57k), Lang-8 ($>$ 1 million)	CommonCrawl	49.49*

Table 5.4: M^2 scores on the CoNLL-2014 test set when using large amounts of monolingual data. NUCLE and Lang-8 are annotated correction datasets of 57k and over 1 million sentences, respectively. RR16 is the model of Rozovskaya and Roth (2016). *The final line is that of Junczys-Dowmunt and Grundkiewicz (2016).

proves by 2.4 points over the SMT system without synthetic data and 1.5 points over the classifier approach alone, establishing a new baseline for the limited annotation setting, without making use of additional human annotations from linguistic tools. Additionally, this result demonstrates that generating the synthetic data provides a benefit even in the context of a web-scale LM.

5.7 Conclusion: Toward Translation-based Correction with Minimal Supervision

We have proposed a new framework for generating synthetic data for the task of sentence correction with limited labeled data. The approach is relatively straightforward to implement and does not depend on external linguistic tools and yet experi-

ments demonstrate that with a strong phrase-based model, using the synthetic data is more effective than alternative proposed approaches that can perform correction from limited annotations, given similar amounts of data. We also show that relatively small amounts of annotated data can be used to generate results similar to those of a sequence-to-sequence model trained on considerably more human annotated data, which in addition to establishing baselines for future work, is an important consideration for the cost-benefit analysis of annotating data for building a real system.

We consider the approach proposed in this chapter to be preliminary, as effectiveness still falls well short of the case when several hundreds of thousands of human annotated sentences are available, ceteris paribus. However, this work makes progress on this more practical scenario when relatively modest amounts of labeled data are available. This work will serve as a strong point of comparison for the adaptation of alternative, contemporaneous approaches for performing translation from limited amounts of parallel data (Artetxe et al., 2017; Lample et al., 2017) to correction, and in future work, we also aim to extend this approach to additional languages.

6

Conclusion

In this dissertation, we have reconsidered core tasks and themes in NLP, providing new insights to long-standing questions on syntax and new modeling approaches for core grammatical error applications.

In Chapter 1, we provided a counter-example to existing work that has suggested that hand-annotated syntactic trees are necessary for modeling ordering constraints.

Contrary to most previous work, we presented evidence from carefully controlled experiments to suggest that signal from such trees, at least at the scale of the Penn Treebank for English, does not exceed that of the ordering information available from the surface-level lexical items alone, with current modeling approaches.

In Chapter 2, we analyzed the output of a surface-level model for ordering at scale. Given adequate computing resources, the surface-level approach for word ordering is reasonably effective, and our analysis suggested that further gains might be possible by simply further increasing the amount of training data. However, there is a dichotomy with such models between the potential for future improvements with additional data and the brittleness in the presence of unseen n-grams.

Based on the results of Chapters 1 and 2, we revisited approaches for grammatical error correction with strong surface-level conditional language models. In Chapter 3, we presented a state-of-the-art approach for grammatical error identification that was externally validated in a shared task competition. Such sequence-to-sequence models have subsequently become standard baselines for grammatical error tasks. In Chapter 4, we presented approaches for adapting these models specifically to the task of correction. The approaches presented in Chapters 3 and 4 require large amounts of labeled data to be effective, so in Chapter 5, we proposed a preliminary approach for correction that requires minimal supervision.

6.1 Future Work

The insights from the diagnostic task of word ordering in Chapter 1 have been useful for improving the effectiveness of models for real-world natural language generation tasks. As future work, two variants of the task, at opposite levels of granularity, may yield additional insights for natural language tasks in the near term. The first variant is to increase the sequence length to the paragraph level. Approaches to ordering bags of words in paragraphs may yield insights for incorporating additional context into models for other generation tasks and will likely require improvements in the efficiency of search, relative to the approach proposed in Chapter 1. The second alternative is to evaluate at the sentence level, as before, but to order bags of characters instead of words. This variant will also likely require novel search approaches and/or alternative modeling approaches, and it generalizes the task to all languages, including those that do not separate words with spaces (to the extent of not requiring language-specific pre-processing for tokenization).

For both of these variants, as well as the basic word ordering task, it would be of interest to analyze the robustness of various models in the presence of noise, including both adversarially placed artificial noise and natural occurring human errors (as are present in the correction datasets).

Grammar correction is not a solved task, and future work will proceed along three lines. First, there has been relatively limited work on correction for languages other

than English. A focus on correction in other languages will necessitate further development of semi-supervised approaches, as in Chapter 5, or unsupervised approaches, and/or the creation of additional labeled datasets, as existing correction datasets for languages other than English are small.

Second, it is not currently known exactly how sensitive existing grammar correction models are to perturbations in inputs (including adversarial examples). This is particularly relevant for correction models, as there is a possibility that variants of models along the lines of those proposed here will begin to appear in real-world deployments. This is a very challenging problem, and following the insights from the work in this dissertation, it would likely be most productive to first carefully examine these issues in the context of the simpler word ordering task.

Finally, as strong models for error identification and correction emerge, it is worth conducting user studies to determine which approach is most effective for pedagogical purposes. As a first step, it would be useful to know, conditioned on the level of proficiency, at what level of granularity grammatical feedback is most effective. For example over time, are language learners less likely to repeat the same mistake if they are given the full corrections at the word-level or if they are simply given a binary indicator of whether or not an error occurred at the sentence-level? Other variants are also possible, such as providing indicators at the word-level or multiple possible corrections. Additionally, the effects of the aforementioned cases may interact with the errors made by the automated approaches, themselves, and this will need to be

accounted for in the analysis. A meaningful study along these lines is challenging to conduct in practice, as it would ideally be conducted with language learners with careful controlling of proficiency levels, demographic characteristics, and other factors in a realistic learning environment.

Against this backdrop, Oppenheimer held a meeting at about 6 p.m. He was terribly worried at the possibility of sabotage on top of all these other troubles, and decided that someone who understood the details should stay with the gadget until it was locked up for firing. We agreed, and as the youngest, I had priority. Soon a violent thunder and lightning storm enveloped the site, but at 9 p.m. (about 9 p.m.) I climbed the 100-foot tower to the top, where I babysat the live bomb. It was a deeply philosophical experience...
...to contemplate the monster beside me and what it was about to do to the world.

Donald F. Hornig (NPR, 2005)

A

An Abbreviated History of the Field

Natural language processing (NLP) started with a bang. The early computational devices used to calculate artillery and bomb parameters[1] during World War II were not long thereafter turned to the task of translation.[2]

[1]The novel *Black Rain* (translated from the original Japanese) by Masuji Ibuse, provides an important perspective (Ibuse and Bester, 2012).

[2]This is not intended as a comprehensive survey of the field. Rather, we aim to highlight for the reader key works related to the technical contributions of this dissertation, the core ideas of which harken back to the earliest days of computing. (Here and elsewhere, we shame-

Encouraged in part by the cryptological advances during the war, Warren Weaver wrote a well-known memo, originally written in 1949, proposing the use of electronic computers for the task of non-literary translation (Weaver, 1949/1955). The memo, which is of a high-level, qualitative nature, is optimistic in the exploitation of the statistical regularity of language, treating the problem of translation as one of decoding:

> [I]t is very tempting to say that a book written in Chinese is simply a book written in English which was coded into the "Chinese code." If we have useful methods for solving almost any cryptographic problem, may it not be that with proper interpretation we already have useful methods for translation?

Suggestions by Weaver, such as the significance of context windows around words, Claude Shannon's statistical work on communication, McCulloch-Pitts neurons (McCulloch and Pitts, 1943), and inter-lingual models were subsequently explored by various research groups.

Weaver's proposal was initially met with some skepticism by Norbert Wiener[3], but it was viewed with considerable interest by a number of research groups. Among others, the memo caught the attention of Howard Aiken, an early constructor of compu-

lessly conflate the phrases "computational linguistics" and "natural language processing", treating them as synonymous for all practical purposes.)

[3]Norbert Wiener, from Columbia, Missouri, was a contemporary of Weaver, having been born in the same year, 1894. Wiener was an early pioneer in areas that would later fall under the field of computer science. At the time, Wiener used the term "cybernetics". Partially because of this, John McCarthy and others favored the term "artificial intelligence" (AI), initially representing the strain of work that emerged from the 1956 Dartmouth Workshop (McCarthy, 8 June 2011). On this point, McCarthy reflected in 1988, "I, and I think most other A.I. researchers avoided our seniors, rather than either following them or opposing them...[The proposals of] Von Neumann, Wiener and McCulloch have not so far been fruitful...Of course, the older proposals have not been refuted, and maybe new people will find them fruitful" (McCarthy, 1988).

tational devices and the architect of the computer science effort at Harvard. Aiken reportedly gave the memo to a young Anthony Oettinger, a Nuremberg-born[4] emigre and graduate of high-school in New York via France, who knew some Russian, instructing him to "See what you can do" (Oettinger, 2006).

Out of this work grew early attempts to first build automated approaches for dictionary look-up (Oettinger, 1954). Direct derivatives of this line of work were insufficient for translation—and it was recognized before embarking on such work that word-for-word translation would not be the final answer; however, the idea of dictionary look-up by electro-mechanical/digital means was a first step in the larger engineering endeavor. Oettinger remarked, as such, in his dissertation in Applied Mathematics in 1954:

> There is, at present, considerable interest in the development of "translating machines." If by a "translating machine" one is to understand "a machine which will present a well-written English version of Russian input text," there is little doubt but that the great difficulties standing in the way of designers will make such machines unavailable for some time to come. It should be clear, however, that an automatic dictionary is one component essential to a translating machine (Oettinger, 1954).

It is, in fact, of interest that any progress at all was made at this point. To get a sense of the challenges, note that in 1954[5], the "modern magnetic drums" that Oet-

[4] "...I was born in Germany in 1929, and more particularly, in the city of Nuremberg, which was the home of the early Nazi party. Some of my earliest memories were of storm trooper torch light parades. But in an odd way, this probably saved my life, because my parents had the wit to notice what was going on," Oettinger recollected years later (Oettinger, 2006).

[5] 1954 was also of note in that it was the year Alan Turing died. Reflecting the milieu of the time, Oettinger had met Turing during a year in England in 1952. There, Oettinger

tinger and Aiken used for storage could only hold about five thousand Russian words with their English equivalents (Oettinger, 1954).

Other groups were also working on building dictionary-based approaches using early computational systems. Victor H. Yngve had taken a side interest in automated translation while still a graduate student in physics at the University of Chicago. On a visit to Bell Labs, Claude Shannon alerted him to the first conference on machine translation in 1952, which was organized by Yehoshua Bar-Hillel at MIT (Yngve, 1997; Hutchins, 2012). After moving to MIT, he would lead a group of scientists and linguists in the new field of machine translation. Early work by Yngve also focused on NLP more broadly, beyond machine translation. This included, among others, an investigation of the relation between natural language and error correcting codes (Yngve, 1954a) and the statistical analysis of distances between words (Yngve, 1956a).

This was also the era of I.J. Good's further analysis of a proposal by Turing[6] for estimating population frequencies (Good, 1953), which is an important early work on distribution smoothing, which is central to much of modern NLP.[7] Later in the decade, Frank Rosenblatt introduced the perceptron (Rosenblatt, 1958), which would prove to be a key building block of subsequent learning approaches in modern NLP.

worked with Maurice Wilkes, head of the University Mathematical Laboratory in Cambridge University. During his time in England, he worked on the problem of learning—in the computational sense (i.e., on work that would later fall under the banner of AI) (Oettinger, 2000).

[6]Good was among the mathematicians working with Turing at Bletchley Park during WWII (van der Vat, 2009).

[7]Smoothing techniques for n-gram language modeling used in contemporary work, such as modified Kneser-Ney smoothing (Kneser and Ney, 1995; Chen and Goodman, 1996) can be seen as successors to the work of Good (1953).

By 1954, the field had sufficiently coalesced to start a new journal around machine translation, initially called *Mechanical Translation*, with the tagline "Devoted to the translation of languages with the aid of machines", edited by William N. Locke and Yngve, both then at MIT (Yngve, 1954b). It was a small, but distinguished group through the 1950's and 1960's. The first meeting of the "Association for Machine Translation and Computational Linguistics" was held in Denver, Colorado from August 25-26, 1963[8] (MT, 1964).

However, the difficulty in making progress toward a practical machine translation system given available computational resources soon became evident. Reflecting the dashed hopes of researchers in the first push for machine translation, the journal *Mechanical Translation and Computational Linguistics* (as then named) ceased publication after 17 years, in 1970. Relatedly, two years after the ALPAC report of 1966, the Association for Machine Translation and Computational Linguistics became simply the Association for Computational Linguistics (Yngve, 1970).

Despite the marked decline in work on machine translation starting in the mid-1960's, the broader field of NLP nonetheless continued.[9] The full set of core, modern applications of NLP were articulated by researchers no latter than the 1960's. The December 1968 edition of the *Communications of the ACM* contains a commis-

[8]Of note (to add historical context) is that the meeting, as well as future scientific and humanistic progress, was made not impossible as a result of tail-risk-recognizing decision-makers prevailing in D.C., in Moscow, and in and around Cuba in October of the previous year.

[9]One could say, in a sense, that machine translation preceded NLP as a field.

sioned article by Susumu Kuno, also of Harvard, and Oettinger containing curricular recommendations for a Ph.D. program in computational linguistics (Kuno and Oettinger, 1968). The article makes reference to research topics and course topics encompassing core areas such as translation, post-editing, retrieval, writing aids, and question-answering, as well as broader subject areas such as the computational humanities/social sciences and applications in bio-medicine. Perhaps owing in part to the early focus on translation, multilingual processing was part of the research agenda from an early stage. The role of NLP in human-computer interaction was also recognized by this time, and research, for example, to explore effective means of inputting Chinese characters was well under way (Hayashi et al., 1968). In other words, the main NLP research agenda for the subsequent 50 years, up to the present, was set by December 1968, a side-note addition to an already peculiarly eventful year in U.S. and world[10] history, technologically and socially.

Future work would depend heavily on advances in computing power. Key innovations in subsequent years in conditional and unconditional language modeling are discussed in the core chapters of this dissertation, as applicable to the topics and tasks considered; however, to a degree probably under-appreciated in the modern literature, these advances had at their origin the explosive developments of the 1940's, 1950's,

[10]At this point, China was deep in the throes of the Cultural Revolution, with scientists and other intellectuals in the cross-hairs. The dramatic turn-around of the country's scientific output 50 years later (including in AI and NLP) would have been difficult to predict at that point by outsiders. For additional context, Spence (1999) provides a historical survey of modern Chinese history, starting from the Ming dynasty.

and 1960's.

At 5:29 all the needles dropped and the red lights went out and there was a brilliant light, which has been mentioned, swirling around outside. Well, that was the end of the war for me, and in fact, the war did end three weeks later. My own reaction then was to be terribly tired. I had hardly slept for 48 hours. My second thought was: We've really opened a can of worms, haven't we? And for many years, I worked on that proposition: Where do we go from here?

Donald F. Hornig (NPR, 2005)

B

Toward Robust, Open Research Oversight: Notes on AI Safety

Well before it reaches its longer-term ambitions, the field of artificial intelligence may end the way it started—with a bang.

At this point, computational systems have already been a part of critical infras-

tructure and administrative systems for several decades, deeply embedded into the modern functioning of everything from utilities to transportation to the military to communication to finance to healthcare and beyond. As noted in Appendix A, the expansion of computational systems into modern society was well underway in the years immediately following WWII. However, a confluence of factors in more recent decades warrant particular concern (but certainly not panic, at least at this stage) as to the deployment of more recent AI models and approaches, as well as to the more distant societal implications.

In part because of the application of computational systems throughout society in the last several decades, we live in an increasingly centralized and interdependent world. Financial disruptions, for example, among areas of the world that would have been relatively disconnected in past centuries (or even decades), now very rapidly propagate throughout the globe. Furthermore, countries and non-state actors with radically different (and in some cases, mutually incompatible) interests have potential access to technological force multipliers that can quickly change the global strategic balance in unexpected ways. Above all, humans do not have a particularly good track-record of anticipating tail risks in the complex systems of the modern world (e.g., as evidenced by the subprime mortgage crisis in the late 2000's or the Fukushima Daiichi nuclear disaster of 2011). In short, fragility has been centralized, and it is a vulnerability we generally do not (even know how to begin to) understand.

In this context, it is worth carefully considering the implications of the wide-spread

deployment of AI systems—before and during development, well-before actual deployment. As in many other aspects of society, centralized, top-down regulation is unlikely to have the robustness necessary to cover most use cases, and regulation of that nature may end up having the unintended (and in some cases, devastating) side effects of hiding risk and encouraging rent-seeking. This is particularly true for AI systems, the development of which may continue to be rapid, with new and unanticipated applications. In contrast to top-down approaches, a small number of carefully considered bottom-up, decentralized approaches may be effective in practice.[1]

In this position brief, which is based on Schmaltz (2018), we propose that the research community consider encouraging researchers to include two riders, a "Lay Summary" and an "AI Safety Disclosure", as part of future NLP papers published in ACL forums that present user-facing systems. The goal is to encourage researchers—via a relatively non-intrusive mechanism—to consider the societal implications of technologies carrying (un)known and/or (un)knowable long-term risks, to highlight failure cases, and to provide a mechanism by which the general public (and scientists in other disciplines) can more readily engage in the discussion in an informed manner.

This simple proposal requires minimal additional up-front costs for researchers; the lay summary, at least, has significant precedence in the medical literature and other areas of science; and the proposal is aimed to supplement, rather than replace, exist-

[1]The key principles for a stable computational public policy are perhaps summarized with the following cliches: local before global, decentralized before centralized.

ing approaches for encouraging researchers to consider the ethical implications of their work, such as those of the Collaborative Institutional Training Initiative (CITI) Program and institutional review boards (IRBs).

B.1 Introduction

Recent research advances in natural language processing have the potential to translate into real-world products and applications. As with the broader field of AI, more generally, there is not a broad consensus on whether the long-term social impact of such advances will be positive or negative—and to whom any future negative impacts will be most acutely dealt. However, there is perhaps consensus that it is useful for researchers to at least consider the potential societal impacts of their work. The concern is not entirely speculative, as user-facing applications of NLP today in areas such as education, for example, have the potential to have large proportions of users who are minors and/or members of at-risk groups, with the output of such systems used in high-stakes educational assessment.

To encourage NLP researchers to consider the societal impacts of their work and to involve the general public in the discussion, we propose that the community consider encouraging authors—on a voluntary basis field-tested in a workshop setting—to include two riders for papers describing user-facing systems or methods. One, a "Lay Summary", which has precedence in journals in other scientific fields, is a short

summary aimed at a non-specialist audience designed to reduce misinformation and engage the public. The second, an "AI Safety Disclosure", is a brief overview of potential failure scenarios of which real-world implementations, downstream applications, and future research should be aware.

We surmise that the utility of these riders will be particularly high for NLP papers for which the proposed approaches or methods are aimed at eventually building user-facing systems (e.g., for machine translation, grammar correction, or summarization), but for which the actual research did not directly involve human subjects and thus (rightly so), fall outside the purview of traditional mechanisms such as institutional review boards.

B.2 Proposal

We propose that NLP articles presenting user-facing systems or methods include two riders, a "Lay Summary" and an "AI Safety Disclosure", as explained further below. By user-facing system or method, we refer to tasks in which the end consumer of the output is a human for performing a real-world task. This would include papers on tasks such as machine translation and summarization, even if the research itself did not involve human subjects. It would exclude papers of a more theoretic nature, or for which the end goal is not user-facing output. For example, this criteria might reasonably exclude a paper introducing a new approach for dependency parsing (McDon-

ald et al., 2005) or an empirical comparison of language models (Chen and Goodman, 1996). It would, however, include papers using dependency parsing or language models as part of a downstream task, such as machine translation. As with other aspects of this proposal, we leave it to the discretion of authors as to whether their paper meets this criteria, and the community may desire to restrict or expand the determination of which papers should include these riders (see Section B.4).

B.2.1 Lay Summary

The idea of including a summary of an article that is accessible to a general audience is a well-established concept, implemented in existing journals in a variety of scientific fields. Such a summary can assist science journalists and inform discussions in public forums. To a lesser extent, such summaries can also be useful for researchers in other branches of science and engineering.

The journal *Autism Research*, for example, requires a lay summary of "2-3 sentences (60-80 words; 300-500 characters including spaces) included at the end of the Abstract that summarizes the impact/importance/relevance/key findings of the study"[2]. In a similar vein, the *Proceedings of the National Academy of Sciences of the United States of America (PNAS)* requires authors to provide a "120-word-maximum statement about the significance of their research paper written at a level understandable

[2] `http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1939-3806/homepage/ForAuthors.html#_Lay_Summary` (accessed March 2018)

to an undergraduate-educated scientist outside their field of specialty. The primary goal of the Significance Statement is to explain the relevance of the work in broad context to a broad readership."[3] Shailes (2017) collected a list of 50 journals across the sciences that provide such summaries[4].

To the best of our knowledge, none of the major NLP conference proceedings or journals currently provide lay summaries, or the equivalent. In implementing this mechanism for the first time in this field, we suspect some experimentation will be needed to set guidelines and best practices, and initially, we recommend not being overly prescriptivist as to the form of the lay summaries (in terms of length, format, content, etc.).

B.2.2 AI Safety Disclosure

The goal of this second rider is to provide a common mechanism for applicable papers to highlight possible failure cases, even if just in broad terms—and even if in a relatively succinct format. Such error scenarios are not always obvious to downstream implementers, and the insight of the original researchers on the behavior of a system can, we surmise, often yield useful general guidelines for future work to consider. A description of failure cases can include an empirical analysis of inputs that generate

[3] `http://blog.pnas.org/iforc.pdf` (accessed March 2018)

[4] The list is available at `https://elifesciences.org/inside-elife/5ebd9a3f/plain-language-summaries-journals-and-other-organizations-that-produce-plain-language-summaries` (accessed March 2018)

incorrect or otherwise unreliable or uncertain outputs, but will often be of a more general, qualitative nature, highlighting potential biases in the output and future work needed to ensure reliable effectiveness in a real-world deployment.

Recognition of error cases can ground researchers in the current state of approaches and provide insights for future research. "It's in the errors that systems make that it's most evident that they have not cleared Turing's hurdle; they are not 'thinking' or 'intelligent' in the same sense in which people are" (Grosz, 2012). At the same time, analyzing errors in a systematic, representative fashion is non-trivial, and the next step of providing interpretable insights is perhaps harder still, and the subject of a burgeoning literature (Doshi-Velez and Kim, 2017). Simply asking researchers to highlight challenges in interpreting their models and problem cases in real-world deployments does not, of course, directly in itself yield innovations in error analysis or model interpretability. However, it does, we believe, encourage researchers to pay additional attention to these issues, and importantly, yields useful guides for downstream work.

Unlike lay summaries, the idea of an AI safety disclosure does not have an exact parallel in other fields nor existing mechanisms in the computer science publishing regime. It is in the spirit of existing guidelines for the treatment of human subjects in research, such as the Collaborative Institutional Training Initiative (CITI) Program, and the basic ethical principles of the Belmont Report (National Commission, 1979); however, importantly, it would also involve cases that would not typically be subject to review by an institutional review board (IRB). Existing IRB procedures are already

well-suited for their target use-cases, and the AI safety disclosure is by no means intended to replace such mechanisms. On the contrary, we recommend that the AI safety disclosure be introduced as a voluntary endeavor with initially relatively informal guidelines, allowing the community to establish best-practices in a bottom-up fashion. There is a danger in institutionalizing ethics, making the system vulnerable to being gamed and engendering inflexibility with unanticipated cases, so the voluntary aspect is a central component of the proposal. In this sense, it is a much lighter-weight alternative to (and largely orthogonal to) the creation of ethics review boards for non-university organizations (Leidner and Plachouras, 2017) and is not intended to involve any particular additional legal obligations.

Perhaps more so than lay summaries, this second type of rider is likely going to need several iterations of experimentation before the community converges on standard guidelines. Given the heterogeneity of papers in NLP, it may well turn out that a single format is not suitable for all types of applicable papers in the field. In Section B.4, we propose that ACL workshops can serve as useful testing grounds toward this end.

B.3 Example

For reference, we provide an example of a Lay Summary and an AI Safety Disclosure for the sentence correction work from previous chapters (in particular Chapter 4).

Note that by the criteria proposed in Section B.2, the word ordering work of Chapter 1 would not be expected to include such riders, since word ordering is a diagnostic task and the paper does not present or propose a user-facing system or method.

Other things being equal, it is possible that grammar correction may turn out to be a relatively benign technology. This may well be true for many NLP technologies, as well. However, it is important to get into this frame of mind of thinking about risks, and as alluded to before, seemingly obvious things to check for or expect may not necessarily be obvious to downstream implementers.

B.3.1 Lay Summary

This paper presents an approach for automatic grammar correction. The model for correction is based on models shown in previous work to be useful for the related task of automatic translation between languages, such as from Chinese to English. These types of models are referred to as a sequence-to-sequence models and are a type of neural network. The paper demonstrates ways of adapting these translation models for use in automatically correcting the grammar of English sentences. Effectiveness is improved over some previously proposed approaches, but the models are still noticeably worse than humans at the task.

B.3.2 AI Safety Disclosure

Effectiveness at the demonstrated levels likely falls short of what is needed for a production system, but ensembles of models (including the intersection of language models) may increase effectiveness. However, since a non-zero proportion of the end users of such a system would likely be minors, it is worth mentioning some general principles to keep in mind when building such a system. In particular, a system built in the manner proposed here would not be particularly robust against biases already present in the aligned parallel data. Flipping of gendered pronouns may occur, and phrases offensive to at-risk populations could be generated. While not explored in the current work, an additional, final classifier may be helpful in filtering such changes.

Learners might be sensitive to errors generated by such a system, learning to emulate the mistakes made in the output of the system. Without additional outside feedback and instruction, humans might learn to make the same false-positive and false-negative errors that the system makes. There is also a larger question of whether the existence of strong automatic correction systems will have the unintended effect of being detrimental to language learning, as students may become over-reliant on such tools. This, too, needs to be investigated further.

B.4 Implementation

In order to minimize disruption of existing peer-review practices and establish best-practices, we recommend that the use of the two riders first be tested in a workshop setting. Additionally, we recommend that the riders not be included as part of papers during peer-review and remain voluntary.

We suspect that adoption of this proposal will be closely correlated with both the real and perceived amount of additional time required on the part of researchers. This can be partially alleviated by providing a series of examples using existing, published papers; however, as with other aspects, we want to emphasize that a goal is to not be overly prescriptivist and to allow the community to establish practices in a bottom-up, decentralized fashion.

Perhaps the most significant administrative effort will need to be placed in deciding how to make these riders accessible to the public. There are, for example, a variety of approaches in how existing science journals present lay summaries (Shailes, 2017), and we defer to conference and journal administrators on how best to present these riders.

B.5 Challenges

The proposal here is a modest departure from what already exists in other fields and is proposed as a voluntary endeavor. However, as with any policy proposal, there will be both anticipated and unanticipated downsides, and we briefly consider the possibil-

ities here.

In terms of lay summaries, it is not a forgone conclusion that all researchers will be able to provide a summary that is understandable by a general audience. Of note, the current *PNAS* guidelines follow an earlier experiment with longer one- to two-page summaries, which "proved a burden for authors and editors. Some authors hit the mark precisely, but more frequently, the summary did not convey the salient features of the paper for a nonexpert" (Verma, 2012). Writing a summary for a general audience is non-trivial but learnable (Dubé and Lapane, 2014), and to the extent that computational tools can assist authors, the NLP community is in a unique position to develop such tools. While not a goal of this proposal, it is possible that a focus on such lay summaries could spark the development of tools that would be of use to authors in other areas of science, as well.

With the AI safety disclosure, we may find that in practice, the disclosures for some common tasks will be very similar across papers. It is possible that including this rider may become a mechanical exercise, with a small set of points reproduced across papers. It is possible that in such a scenario, the riders would be simply ignored in most cases by readers and authors, alike. One way to avoid this outcome would be to create an evolving challenge set of inputs/scenarios for common tasks on which previous approaches fail. The disclosures could then include results on these common sets, as well as announce additions to the challenge sets.

Researchers may be reluctant to acknowledge the potential downsides of their re-

search. In some cases, a conflict of interest may prevent fully disclosing negative impacts. One approach to displaying the riders would be to do so with a forum that allows feedback from fellow researchers, perhaps in the style of the public reviews of openreview.net. However, invariably, there will be unevenness in the quality of the riders provided by authors (and/or in subsequent feedback), and the community will have to decide whether the benefits of having such riders outweigh such inconsistencies.

As noted above, these riders are not intended to carry any additional, particular legal weight (beyond that already present in the current research and publishing regime) in preventing a downstream application from implementing a system in contravention of concerns raised in a given "AI Safety Disclosure". However, we surmise that this type of bottom-up, public, decentralized approach can often be quite effective in influencing community norms.

B.6 Related Work

There is an emergent literature on AI safety and research ethics. Hovy and Spruit (2016) sparked recent research on the ethical significance of NLP research, with a focus on the impact of NLP on social justice. The contemporaneous work of Gebru et al. (2018) proposes a common mechanism for specifying potential biases within, and other characteristics of, datasets and trained models. The resulting "datasheet"

is in the spirit of, and compatible with, our proposal, and in future work, we plan to explore combining these approaches. Grosz (2018) notes that "ethics must be taken into account from the start of system design", and the proposal here might be one small step in encouraging researchers to consider broader ethical implications as they develop their research.

There is a related, older literature addressing the limitations and potential unintended societal risks of complex, high-impact computational systems, more generally, of which the analysis of command and control systems is an illustrative example (Borning, 1987). A common theme of such work, as in the more recent work on biases in training data, is that data and technology reflect the social and political zeitgeist in which they are constructed. Technological solutions that ignore such coupling—even if well-intentioned—risk exacerbating existing tensions and creating new tensions.

There are a growing number of calls from scientists and journal editors for the need for lay summaries (Rodgers, 2017; Kuehne and Olden, 2015). Similarly, there is growing recognition for the need to both inform the general public about the state and possible future of AI, and to receive feedback from the public as stakeholders. Many of the realistic, near-term downsides of the current progress of AI, more generally, are likely to disproportionally impact those that are not AI researchers: commercial drivers, manufacturing workers, those in conflict zones, and those living under authoritarian governments, among others. Efforts to engage the public and/or broader cross-disciplinary collaborations include multi-disciplinary conferences, such as the recent

AAAI/ACM conference on Artificial Intelligence, Ethics, and Society; public outreach efforts by organizations such as the Future of Life Institute[5]; and efforts to summarize progress in AI for a wider audience, such as the AI Index[6] (Shoham, 2017).

B.7 Conclusion: The Utility of Lay Summaries and AI Safety Disclosures

We recommend that future NLP papers presenting user-facing systems or methods include a short summary accessible to a general public and a brief overview of possible failure scenarios (even if speculative) of which future implementations and work should be aware. This proposal is a modest departure from what already exists in other scientific fields and involves a relatively lightweight change to existing publishing procedures in NLP. Experimentation of such an approach in an ACL workshop setting will be useful for gaining feedback from the research community and the public, and we recommend such an incremental, evaluative approach before applying it to full conferences and journals.

[5] `https://futureoflife.org/`
[6] `https://aiindex.org/`

References

M. Artetxe, G. Labaka, E. Agirre, and K. Cho. 2017. Unsupervised Neural Machine Translation. *ArXiv e-prints*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473.

Alan Borning. 1987. Computer System Reliability and Nuclear War. *Commun. ACM*, 30(2):112–131.

Fraser Bowen, Jon Dehdari, and Josef Van Genabith. 2017. The Effect of Error Rate in Artificially Generated Data for Automatic Preposition and Determiner Correction. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 68–76, Copenhagen, Denmark. Association for Computational Linguistics.

Chris Brew. 1992. Letting the Cat out of the Bag: Generation for Shake-and-Bake MT. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 610–616, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL Errors Using Phrasal SMT Techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256, Sydney, Australia. Association for Computational Linguistics.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A Statistical Approach to Machine Translation. *Computational linguistics*, 16(2):79–85.

Aoife Cahill, Nitin Madnani, Joel Tetreault, and Diane Napolitano. 2013. Robust Systems for Preposition Error Correction Using Wikipedia Revisions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, pages 507–517, Atlanta, Georgia. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. *arXiv e-prints*, page arXiv:1312.3005.

Stanley F. Chen and Joshua Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 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)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Shamil Chollampatt, Duc Tam Hoang, and Hwee Tou Ng. 2016a. Adapting Grammatical Error Correction Based on the Native Language of Writers with Neural Network Joint Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1901–1911, Austin, Texas. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the Dots: Towards Human-Level Grammatical Error Correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 327–333, Copenhagen, Denmark. Association for Computational Linguistics.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016b. Neural Network Translation Models for Grammatical Error Correction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2768–2774. AAAI Press.

Lonnie Chrisman. 1991. Learning Recursive Distributed Representations for Holistic Computation. *Connection Science*, 3(4):345–366.

Daniel Dahlmeier and Hwee Tou Ng. 2012a. A Beam-search Decoder for Grammatical Error Correction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 568–578, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012b. Better Evaluation for Grammatical Error Correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 568–572, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A Report on the Preposition and Determiner Error Correction Shared Task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert Dale and Adam Kilgarriff. 2011. Helping Our Own: The HOO 2011 Pilot Shared Task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, ENLG '11, pages 242–249, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vidas Daudaravicius. 2016. Automated Evaluation of Scientific Writing Data Set (Version 1.2) [Data file]. VTeX.

Vidas Daudaravicius, Rafael E. Banchs, Elena Volodina, and Courtney Napoles. 2016a. A Report on the Automatic Evaluation of Scientific Writing Shared Task. In *Proceedings of the Eleventh Workshop on Innovative Use of NLP for Building Educational Applications*, San Diego, CA, USA. Association for Computational Linguistics.

Vidas Daudaravicius, Rafael E. Banchs, Elena Volodina, and Courtney Napoles. 2016b. A Report on the Automatic Evaluation of Scientific Writing Shared Task.

In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 53–62, San Diego, CA. Association for Computational Linguistics.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941, International Convention Centre, Sydney, Australia. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

F. Doshi-Velez and B. Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. *ArXiv e-prints*.

Catherine E. Dubé and Kate L. Lapane. 2014. Lay Abstracts and Summaries: Writing Advice for Scientists. *Journal of Cancer Education*, 29(3):577–579.

Nadir Durrani, Alexander Fraser, Helmut Schmid, Hieu Hoang, and Philipp Koehn. 2013. Can Markov Models Over Minimal Translation Units Help Phrase-Based SMT? In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 399–405, Sofia, Bulgaria. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding Structure in Time. *Cognitive Science*, 14(2):179 – 211.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24, Baltimore, Maryland. Association for Computational Linguistics.

Mikel L. Forcada and Ramón P. Ñeco. 1997. Recursive Hetero-associative Memories for Translation. In *Proceedings of the International Work-Conference on Artificial*

and Natural Neural Networks: Biological and Artificial Computation: From Neuroscience to Technology, IWANN '97, pages 453–462, Berlin, Heidelberg. Springer-Verlag.

T. Gebru, J. Morgenstern, B. Vecchione, J. Wortman Vaughan, H. Wallach, H. Daumeé, III, and K. Crawford. 2018. Datasheets for Datasets. *ArXiv e-prints*.

Adrià de Gispert, Marcus Tomalin, and Bill Byrne. 2014. Word Ordering with Phrase-Based Grammars. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 259–268, Gothenburg, Sweden. Association for Computational Linguistics.

I. J. Good. 1953. The Population Frequencies of Species and the Estimation of Population Parameters. *Biometrika*, 40(3-4):237–264.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2014. Randomized Significance Tests in Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 266–274, Baltimore, Maryland, USA. Association for Computational Linguistics.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2016. Efficient softmax approximation for GPUs. *arXiv e-prints*, page arXiv:1609.04309.

Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR*, abs/1308.0850.

Barbara J. Grosz. 2012. What Question Would Turing Pose Today? *AI Magazine*, 33(4):73–81.

Barbara J. Grosz. 2018. Smart Enough to Talk With Us? Foundations and Challenges for Dialogue Capable AI Systems. *Computational Linguistics*, 44(1):1–15.

Eva Hasler, Felix Stahlberg, Marcus Tomalin, Adrià de Gispert, and William Byrne. 2017. A Comparison of Neural Models for Word Ordering. In *Proceedings of the International Natural Language Generation Conference*.

Hideyuki Hayashi, Sheila Duncan, and Susumu Kuno. 1968. Computational Linguistics: Graphical Input/Output of Nonstandard Characters. *Commun. ACM*, 11(9):613–618.

Duc Tam Hoang, Shamil Chollampatt, and Hwee Tou Ng. 2016. Exploiting N-best Hypotheses to Improve an SMT Approach to Grammatical Error Correction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2803–2809. AAAI Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Matic Horvat and William Byrne. 2014. A Graph-Based Approach to String Regeneration. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 85–95, Gothenburg, Sweden. Association for Computational Linguistics.

Dirk Hovy and Shannon L. Spruit. 2016. The Social Impact of Natural Language Processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598, Berlin, Germany. Association for Computational Linguistics.

W. John Hutchins. 2012. Victor H. Yngve. *Computational Linguistics*, 38(3):461–467.

Masuji Ibuse and John Bester. 2012. *Black Rain*, 1st us ed. edition. Kodansha USA, New York.

J. Ji, Q. Wang, K. Toutanova, Y. Gong, S. Truong, and J. Gao. 2017. A Nested Attention Neural Hybrid Model for Grammatical Error Correction. *ArXiv e-prints*.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A Nested Attention Neural Hybrid Model for Grammatical Error Correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 753–762, Vancouver, Canada. Association for Computational Linguistics.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the Limits of Language Modeling. *arXiv preprint arXiv:1602.02410*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction. In

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1546–1556, Austin, Texas. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-Aware Neural Language Models. In *Proceedings of AAAI*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Reinhard Kneser and Hermann Ney. 1995. Improved Backing-off for M-Gram Language Modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Philipp Koehn. 2010. *Statistical Machine Translation*, 1st edition. Cambridge University Press, New York, NY, USA.

Ikuo Kudo, Hideya Koshino, Moonkyung Chung, and Tsuyosi Morimoto. 1988. Schema Method: A Framework for Correcting Grammatically Ill-formed Input. In *Coling Budapest 1988 Volume 1: International Conference on Computational Linguistics*.

Lauren M. Kuehne and Julian D. Olden. 2015. Opinion: Lay Summaries Needed to Enhance Science Communication. *Proceedings of the National Academy of Sciences*, 112(12):3585–3586.

Susumu Kuno and Anthony G. Oettinger. 1968. Computational Linguistics in a Ph.D. Computer Science Program. *Commun. ACM*, 11(12):831–836.

G. Lample, L. Denoyer, and M. Ranzato. 2017. Unsupervised Machine Translation Using Monolingual Corpora Only. *ArXiv e-prints*.

Jochen L. Leidner and Vassilis Plachouras. 2017. Ethical by Design: Ethics Best Practices for Natural Language Processing. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 30–40, Valencia, Spain. Association for Computational Linguistics.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The Winograd Schema Challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'12, pages 552–561. AAAI Press.

Jiangming Liu and Yue Zhang. 2015. An Empirical Comparison Between N-gram and Syntactic Language Models for Word Ordering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 369–378, Lisbon, Portugal. Association for Computational Linguistics.

Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-Based Syntactic Linearization. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 113–122, Denver, Colorado. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

John McCarthy. 1988. The Logic and Philosophy of Artificial Intelligence. *Kyoto Prize Lecture.*

John McCarthy. 8 June 2011. An Oral History Conducted in 2011 by Peter Asaro with Selma Šabanovic. *Indiana University, Bloomington Indiana, for Indiana University and the IEEE.*

Warren S. McCulloch and Walter Pitts. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective Dependency Parsing Using Spanning Tree Algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2012. The Effect of Learner Corpus Size in Grammatical Error Correction of ESL Writings. In *Proceedings of COLING 2012: Posters*, pages 863–872, Mumbai, India. The COLING 2012 Organizing Committee.

MT. 1964. Association for Machine Translation and Computational Linguistics. *Mechanical Translation*, 8(1).

Courtney Napoles and Chris Callison-Burch. 2017. Systematically Adapting Machine Translation for Grammatical Error Correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 345–356, Copenhagen, Denmark. Association for Computational Linguistics.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 95–100, Stroudsburg, PA, USA. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground Truth for Grammatical Error Correction Metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016. GLEU Without Tuning. *eprint arXiv:1605.02592 [cs.CL].*

The National Commission. 1979. The Belmont Report: Ethical Principles and Guidelines for the Protection of Human Subjects of Research. *United States, National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research.*

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Diane Nicholls. 2003. The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 conference*, pages 572–581.

NPR. 2005. Trinity Scientist Recalls Building the A-Bomb. *All Things Considered*, 16 July 2005. Radio.

Oettinger. 2006. Anthony Oettinger Interview: January 10-11, 2006. In *ACM Oral History Interviews*, New York, NY, USA. ACM. Interviewer-Akera, Atsushi and Interviewee-Oettinger, Anthony.

Anthony Oettinger. 2000. Machine Translation at Harvard. In *Early Years in Machine Translation: Memoirs and Biographies of Pioneers, Hutchins, William John [Ed], Amsterdam: John Benjamins, 2000, pp 73-86.*

Anthony G Oettinger. 1954. A Study for the Design of an Automatic Dictionary. Harvard University Dissertation (Applied Mathematics). Chapter 1.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the*

40th Annual Meeting on Association for Computational Linguistics, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Y. Albert Park and Roger Levy. 2011. Automated Whole Sentence Grammar Correction Using a Noisy Channel Model. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 934–944, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jordan B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. Technical report, OpenAI.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara Parser: A Fast and Accurate Dependency Parser. *CoRR*, abs/1503.06733.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial Error Generation with Machine Translation and Syntactic Patterns. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 287–292, Copenhagen, Denmark. Association for Computational Linguistics.

Peter Rodgers. 2017. Plain-language Summaries of Research: Writing for different readers. *eLife*, 6:e25408.

F. Rosenblatt. 1958. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6):386–408.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. 2014. The Illinois-Columbia System in the CoNLL-2014 Shared Task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 34–42, Baltimore, Maryland. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2016. Grammatical Error Correction: Machine Translation and Classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2205–2215, Berlin, Germany. Association for Computational Linguistics.

Alla Rozovskaya, Dan Roth, and Mark Sammons. 2017. Adapting to Learner Errors with Minimal Supervision. *Computational Linguistics*, 0(ja):1–53.

Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. The UI System in the HOO 2012 Shared Task on Error Correction. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 272–280, Montréal, Canada. Association for Computational Linguistics.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical Error Correction with Neural Reinforcement Learning. *CoRR*, abs/1707.00299.

Allen Schmaltz. 2018. On the Utility of Lay Summaries and AI Safety Disclosures: Toward Robust, Open Research Oversight. In *Proceedings of the Second ACL Workshop on Ethics in Natural Language Processing*, pages 1–6. Association for Computational Linguistics.

Allen Schmaltz, Yoon Kim, Alexander Rush, and Stuart Shieber. 2017. Adapting Sequence Models for Sentence Correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2807–2813, Copenhagen, Denmark. Association for Computational Linguistics.

Allen Schmaltz, Yoon Kim, Alexander M. Rush, and Stuart Shieber. 2016a. Sentence-Level Grammatical Error Identification as Sequence-to-Sequence Correction. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 242–251, San Diego, CA. Association for Computational Linguistics.

Allen Schmaltz, Alexander M. Rush, and Stuart Shieber. 2016b. Word Ordering Without Syntax. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2319–2324, Austin, Texas. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics.

Sarah Shailes. 2017. Plain-language Summaries of Research: Something for everyone. *eLife*, 6:e25411.

Yoav Shoham. 2017. Towards the AI Index. *AI Magazine*, 38(4):71–77.

Jonathan D Spence. 1999. *The Search for Modern China*, 2nd ed. edition. W.W. Norton, New York.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training Very Deep Networks. *CoRR*, abs/1507.06228.

Andreas Stolcke. 2002. SRILM – An Extensible Language Modeling Toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*, volume 2, pages 901–904, Denver.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and Aspect Error Correction for ESL Learners Using Global Context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202, Jeju Island, Korea. Association for Computational Linguistics.

Dan van der Vat. 2009. Jack Good. *The Guardian*, 28 April.

Inder M. Verma. 2012. PNAS Plus: Refining a Successful Experiment. *Proceedings of the National Academy of Sciences of the United States of America*, 109(34):13469–13469.

Warren Weaver. 1949/1955. Translation. In William N. Locke and A. Donald Boothe, editors, *Machine Translation of Languages*, pages 15–23. MIT Press, Cambridge, MA. Reprinted from a memorandum written by Weaver in 1949.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. 2016. Neural Language Correction with Character-Based Attention. *CoRR*, abs/1603.09727.

Helen Yannakoudakis, Marek Rei, Øistein E. Andersen, and Zheng Yuan. 2017. Neural Sequence-Labelling Models for Grammatical Error Correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2785–2796, Copenhagen, Denmark. Association for Computational Linguistics.

V. Yngve. 1956a. Gap Analysis and Syntax. *Information Theory, IRE Transactions on*, 2(3):106–112.

V. H. Yngve. 1954a. Language as an Error Correcting Code. *Quarterly Progress Report of the Research Laboratory of Electronics, M.I.T.*

V. H. Yngve. 1954b. Mechanical Translation. *Quarterly Progress Report of the Research Laboratory of Electronics, M.I.T.*

Victor H. Yngve. 1956b. The translation of languages by machine [with discussion]. In *Information theory: papers read at a symposium on 'Information theory' held at the Royal Institution, London, September 12th to 16th, 1955; edited by Colin Cherry*, pages 195–203. London: Butterworths Scientific Publications.

Victor H. Yngve. 1970. Final Report on the Journal MT. *Mechanical Translation and Computational Linguistics.*

Victor H. Yngve. 1997. Something Old and Something New. *Proceedings of the MT Summit VI. Machine Translation: Past, Present, Future*, pages 27–28.

Zheng Yuan and Ted Briscoe. 2016. Grammatical Error Correction using Neural Machine Translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent Neural Network Regularization. *CoRR*, abs/1409.2329.

Ye Zhang and Byron Wallace. 2015. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *CoRR*, abs/1510.03820.

Yue Zhang, Graeme Blackwood, and Stephen Clark. 2012. Syntax-based Word Ordering Incorporating a Large-scale Language Model. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 736–746, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2011. Syntax-based Grammaticality Improvement Using CCG and Guided Search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1147–1157, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2015. Discriminative Syntax-based Word Ordering for Text Generation. *Comput. Linguist.*, 41(3):503–538.