PoMo: Generating Entity-Specific Post-Modifiers in Context

Jun Seok Kang[1], **Robert L. Logan IV**[2], **Zewei Chu**[3], **Yang Chen**[3],
Dheeru Dua[2], **Kevin Gimpel**[4], **Sameer Singh**[2], **Niranjan Balasubramanian**[1]
[1]Stony Brook University, NY, USA
[2]University of California, Irvine, CA, USA
[3]University of Chicago, IL, USA
[4]Toyota Technological Institute at Chicago, IL, USA
{junkang,niranjan}@cs.stonybrook.edu
{rlogan,ddua,sameer}@uci.edu
{zeweichu,yangc1}@uchicago.edu
{kgimpel}@ttic.edu

Abstract

We introduce entity post-modifier generation as an instance of a collaborative writing task. Given a sentence about a target entity, the task is to automatically generate a post-modifier phrase that provides contextually relevant information about the entity. For example, for the sentence, "Barack Obama, _____, supported the #MeToo movement.", the phrase "a father of two girls" is a contextually relevant post-modifier. To this end, we build PoMo, a post-modifier dataset created automatically from news articles reflecting a journalistic need for incorporating entity information that is relevant to a particular news event. PoMo consists of more than 231K sentences with post-modifiers and associated facts extracted from Wikidata for around 57K unique entities. We use crowdsourcing to show that modeling contextual relevance is necessary for accurate post-modifier generation.

We adapt a number of existing generation approaches as baselines for this dataset. Our results show there is large room for improvement in terms of both identifying relevant facts to include (knowing which claims are relevant gives a $> 20\%$ improvement in BLEU score), and generating appropriate post-modifier text for the context (providing relevant claims is not sufficient for accurate generation). We conduct an error analysis that suggests promising directions for future research.

1 Introduction

The goal of machine-in-the-loop writing systems is to assist human writers by directly augmenting their text. Examples include systems that refine human text for grammar (Rao and Tetreault, 2018), collaborate on story plot generation systems (Clark et al., 2018; Yu and Riedl, 2012), or modify the content for style (Hu et al., 2017; Shen et al., 2017; Yang et al., 2018). In this paper, we introduce post-modifier generation as an instance of such an assistive writing task in the news domain. Journalists use post-modifiers to introduce background information about entities discussed in news articles. To write these post-modifiers journalists often need to look up relevant facts about entities. A post-modifier generation system can be seen as a collaborative assistant that automatically finds relevant facts and inserts a small text fragment that augments the text produced by the human writer.

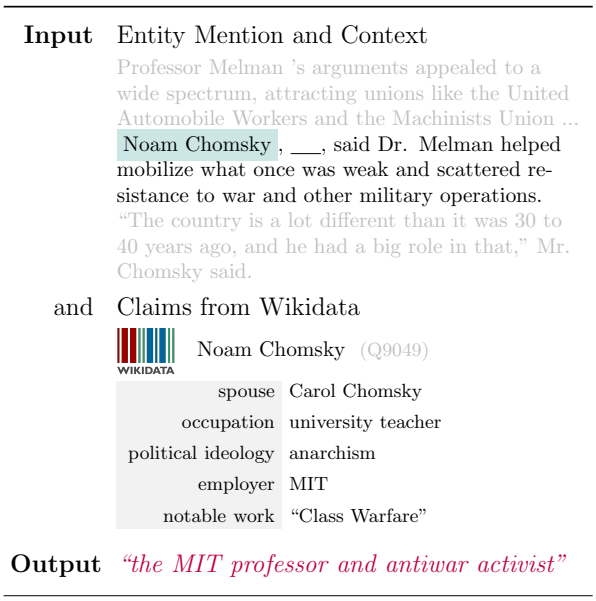


Figure 1: Post-Modifier Generation Task

Post-modifier generation is a *contextual* data-to-text generation problem, where the data is the set of known facts about the target entity, and the text to be generated is a post-modifier that is relevant to the rest of the information conveyed in the text. Figure 1 shows an example. Given a sentence about the anti-war resistance work of Noam Chomsky, the target entity, and a set of known facts about him, the task is to generate a post-modifier that introduces Chomsky as a professor and mentions

Proceedings of NAACL-HLT 2019, pages 826–838
Minneapolis, Minnesota, June 2 - June 7, 2019.

his background as an anti-war activist. An effective post-modifier generation system must: (i) select suitable facts about the entity given the text, and (ii) produce text that covers these facts in a way that fits in with the rest of the text.

We introduce PoMo, an automatically generated dataset for developing post-modifier generation systems.[1] PoMo is a collection of sentences that contain entity post-modifiers, along with a collection of facts about the entities obtained from Wikidata (Vrandečić and Krötzsch, 2014). We use a small number of dependency patterns to automatically identify and extract post-modifiers of entities in sentences. We then link the extracted entities with the entries in Wikidata. The resulting dataset has 231,057 instances covering 57,966 unique entities. Our analysis show that the post-modifiers often combine multiple facts and are specific to the sentential context.

We conduct two sets of experiments that highlight the challenges in post-modifier generation. **(i) Claim Selection:** Given an input sentence, the first step in generating a post-modifier is to figure out which facts to use. We formulate this as a distantly-supervised ranking problem, where we train neural models that learn to identify relevant claims for a given sentence. These claim ranking models perform well when predicting the relevance of coarse-grained facts (e.g. occupation), but fare poorly when predicting finer-grained facts (e.g. place of birth). **(ii) Generation:** We adapt recent sequence-to-sequence generation models for this task. Results show that generation remains a challenge. Even though our automatic claim ranking does not improve generation, further experiments with oracle selected claims demonstrate that when relevant claims are known, the models can generate post-modifiers which humans deem comparable in quality to ones written by professional journalists.

In summary, the main contributions of this work are: 1) a data-to-text problem that introduces new challenges, 2) an automated dataset creation pipeline and a large resulting dataset, 3) a crowdsourcing study that verifies the contextual relevance of post-modifiers, and 4) a characterization of the difficulty of the task via performance analysis of numerous baselines.

[1] https://stonybrooknlp.github.io/PoMo/

	CNN	DM	NYT	**Total**
Train	6,557	11,323	202,735	**220,615**
Valid	162	267	4,771	**5,200**
Test	181	288	4,773	**5,242**
Total	6,900	11,878	212,279	**231,057**

Table 1: Dataset distribution by sources.

2 PoMo: Task and Dataset

Post-modifier generation can be formulated as a data-to-text generation problem. The input is text mentioning a target entity and a set of known facts about the entity. The output is a phrase that: (i) fits as a post-modifier of the target entity mentioned in the input text, and (ii) conveys a subset of facts relevant to the context of the input text.

Figure 1 shows an example for the target entity Noam Chomsky. The input includes a sentence mentioning Chomsky's work on mobilizing anti-war groups along with its surrounding context, and a listing of all facts about Chomsky that are available in Wikidata. Given these inputs, the task is to output a post-modifier phrase that conveys facts about Chomsky that fit within the sentence. In this example the post-modifier conveys both general background information about Chomsky (his occupation), and specific information relevant to the context of the sentence (being an anti-war activist).

This task can be seen as an instance of collaborative writing, where the journalist writes text about specific news events involving entities, and the generation system assists the journalist by inserting new text that augments the story. Given a large collection of news articles, we can automatically create training data for such systems by removing the pieces of text that we want the assistant to generate. This requires reliable ways to identify text to remove and sources of information that can be used to generate the text. Here we describe a pipeline for generating such a dataset for our task.

2.1 Dataset

We construct the PoMo dataset using three different news corpora: NYTimes (Sandhaus, 2008), CNN and DailyMail (Hermann et al., 2015). We use Wikidata to collect facts about entities.[2]

[2] Wikidata dump from https://www.wikidata.org/wiki/Wikidata:Database_download (Dump date: 2018/06/25)

2.1.1 Post-Modifier and Entity Identification

We use Stanford CoreNLP (Manning et al., 2014) to parse each sentence in the news articles and to identify named entities. We extract post-modifiers by finding noun phrases that share an *appos* relation[3] with any recognized named entity in the sentence. In this work, we only consider post-modifiers for *people*. In the future, we plan to expand `PoMo` to include more post-modifiers for other targets, such as organizations. We extract only one such pair from a given sentence to reduce the possible noise in the extraction process.

In our running example from Figure 1, `Noam Chomsky` is recognized as a person entity. The word "professor" is an appositive dependency of the word "Chomsky" and therefore, we extract the NP "the Massachusetts Institute of Technology professor and antiwar activist" which includes the word "professor" as a post-modifier for the target entity `Noam Chomsky`.

2.1.2 Entity Claim Matching

Wikidata provides information about entities in the form of key-value pairs that are called *claims*. To collect the facts about a target entity, we need to link the target to a specific entity in Wikidata. We first search through Wikidata labels and aliases to find candidates with the same name as the target. We sort the candidates based on the number of claims that have a significant word overlap with the extracted post-modifier. We link the entity to the highest ranked candidate whose claims cover at least 30% of the non stop words in the post-modifier. If such a candidate is found we record the claims that overlap with the post-modifier. If no such candidate is found then we discard the entity.

We evaluate this simple heuristic by comparing the results to using an off-the-shelf entity linking system AIDA-light (Nguyen et al., 2014) and show the results in Table 2. We find that AIDA-light agrees with our entity linking in 91.2% of the cases. AIDA-light is able to link 94.3% of the entities we found from NYTimes, but for CNN and DailyMail, it links only 87.0% and 86.34% of the entities, respectively. This decrease is likely due to the fact that AIDA-light was last updated in 2014 while the CNN/DailyMail datasets contain articles collected until the end of April 2015. On the other hand, NYTimes articles range from 1987 to 2007. Our heuristic seems to be reasonably reliable as it does not depend on anything else but the data sources: news articles and Wikidata.

[3] An *appos*itional modifier of an NP is another NP immediately to the right that defines or modifies the NP.

	AIDA Succ.	Agreement
Overall	93.66	91.22
Train	93.65	91.16
Valid	94.06	91.13
Test	93.80	93.65
CNN	87.03	90.34
DM	86.34	85.66
NYT	94.29	91.53

Table 2: Percent agreement with AIDA-light's named entity disambiguation results.

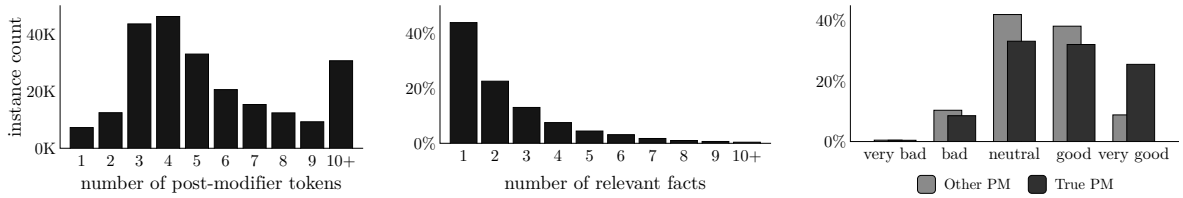
2.2 Analysis

Table 1 shows the distribution of the data sources over train, validation, and test sets. All splits maintain the relative distributions of the data sources to prevent stylistic mismatches from influencing generation. We also ensure that there is no entity overlap among the splits. Within the NYTimes data, we verify that the distribution over years between 1987 and 2007 is also similar over the sets.

Distribution of Post-Modifiers and Entities Figure 2a shows the distribution of post-modifier lengths in terms of token counts. Most post-modifiers are three to eight words long, and about 17.3% are even longer. Figure 2b shows an estimate of the number of relevant facts covered by the post-modifiers; this estimate uses the number of claims that overlap with the post-modifier via heuristic matching. More than half of the post-modifiers convey two or more facts. About 11.4% convey five or more facts. These results suggest that generating post-modifiers requires composing together multiple relevant facts.

Table 3 lists the most frequent types of facts used in the post-modifiers in our dataset. Most relate to generic biographical information such as the entity's occupation, organizations they belong to, place of birth, etc. Here again we see a range of types of information being conveyed which is likely to present a challenge for generation systems.

The dataset also covers a wide variety of entity types. We cluster the target entities by their occupation listed in Wikidata. We also use WordNet (Miller, 1995) to traverse the hypernyms of the words to find frequent ones. Then, we manually select the top ten occupation types. Any entity that



(a) Histogram of the token counts of the post-modifiers. Majority of the post-modifiers (171K instances, 74.14%) have 3 to 8 tokens. Average is 5.8 tokens.

(b) Number of relevant facts per instance in the dataset. More than a half of the post-modifiers are related to two or more facts.

(c) Histogram of the scores for post-modifiers, averaged over three annotations. The distribution of ratings for true and other post-modifiers.

Figure 2: PoMo Post-Modifier Statistics

Fact Type	Count
position held	151,959
occupation	82,781
educated at	53,067
member of political party	42,416
member of sports team	41,602
employer	36,412
award received	31,618
position played on team / speciality	23,987
country of citizenship	17,444
nominated for	15,139
place of birth	9,185
participant of	8,520
member of	7,565
languages spoken, written or signed	4,827
place of death	4,071

Table 3: Top 15 frequent fact types based on heuristic fact coverage identification.

does not belong to the top ten is assigned to a single *other* group. The resulting distribution is shown in Table 4.

Quality of Post-Modifiers We conduct a crowdsourcing study to understand how often the post-modifiers are specific to the particular context. For each (entity, context, post-modifier) triple in the validation set, we create multiple alternative post-modifiers by randomly choosing up to ten other post-modifiers that are found in some other sentences for the same entity. Crowd workers rate the quality of these post-modifiers. Figure 3 shows a screenshot of a task given to crowd workers. If the true post-modifier, the one that is actually used in the context, is rated the highest compared to the rest, then we assume the post-modifier is indeed specific to the context. On the other hand, if the crowd workers rate multiple other post-modifiers as good fits for the context, then the true post-modifier is not context specific. Figure 2c shows the distribution of ratings for true and other post-modifiers. The true post-modifiers tend to be rated *very good* or *good* more often than the other post-modifiers.

Occupation	Count	Percentage
athlete	13,560	23.39%
writer	9,177	15.83%
politician	8,518	14.69%
entertainer	6,488	11.19%
other	5,870	10.13%
scientist	4,487	7.74%
artist	4,175	7.20%
official	2,098	3.62%
lawyer	1,132	1.95%
educator	961	1.66%
capitalist	789	1.36%
scholar	711	1.23%

Table 4: Distribution of the inferred occupations of the target entities. Entities clustered by their occupation.

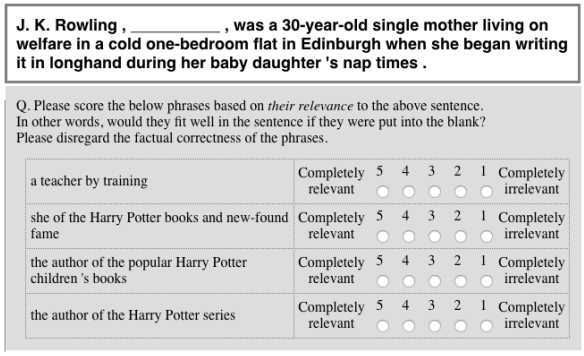


Figure 3: Screenshot of the crowdsourcing task. We asked crowd to rate the quality of post-modifiers.

This suggests that in many cases post-modifiers are specific to the context and cannot be simply replaced by other post-modifiers.

3 Relevant Claim Selection

One of the key challenges of generating post-modifiers is to identify the claims about an entity that are relevant to the given context. In this section, we explore methods for solving this task.

3.1 Methods

We consider three different models: a most-common claim baseline and two neural baselines.

Most-Common Claim This model employs a simple frequency heuristic: rank claims by the frequency of their types in the training post-modifiers (e.g. as in the order given in Table 3) and deem the top n claims in this ranking as relevant.

Neural Baselines We use two neural baselines with the following architecture. Word embeddings are used to represent words in the context (e.g. current and previous sentence) and claims. The sequences of embeddings are then fed through 2-layer LSTM's (Hochreiter and Schmidhuber, 1997) to obtain separate representations of the context and claims. These representations are subsequently concatenated together and fed through a fully-connected layer with sigmoid activation, producing a scalar value for each claim representing the probability that it is relevant. We use this model in two ways: as a classifier, and as a ranking model. When used as a classifier, any claim whose score exceeds a threshold τ is predicted to be relevant. When used as a ranking model, the top n highest-scoring claims are predicted to be relevant.

3.2 Experiments

We train our baselines on the PoMo dataset, using the claims detected during dataset collection as a (distant) source of supervision. Precision, recall, and F_1 score are used to evaluate model performance. Model hyperparameters are chosen using (coarse) grid search to maximize F_1 score on the validation set. The neural baselines use a vocabulary size of 50,000, 100-dimensional word embeddings, and 256 hidden units in the LSTM layers. Dropout (Srivastava et al., 2014) is applied between the LSTM layers with a 0.5 keep probability. The neural classifier uses threshold $\tau = 0.37$. We find the optimal value of n is 4 for the most-common claims model and 2 for the neural ranker.

Quantitative results are provided in Table 5. Both neural baselines perform considerably better than the most-common claims model. This indicates that the provided contexts and claim values contain useful information for claim selection that goes beyond the information captured by global statistics of the dataset alone. We additionally observe that the ranking-based approach outperforms the classification-based approach in terms of both precision and F_1 score, while having only slightly worse recall.

	Prec.	Recall	F_1
Most-Common Claim (n=4)	39.9	51.6	45.0
Neural Classifier (τ=0.37)	52.0	63.8	57.4
Neural Ranker (n=2)	66.5	62.7	64.5

Table 5: Baseline model performance on the claim selection task.

Fact Type	F_1
employer	76.95
position played on team / speciality	76.65
position held	63.10
occupation	50.02
member of political party	48.71
member of	45.60
member of sports team	38.53
award received	37.53
nominated for	30.87
educated at	29.56
participant of	29.04
country of citizenship	16.28
place of death	14.72
place of birth	6.80
languages spoken, written or signed	0.00

Table 6: F_1 score of neural ranker ($n = 2$) on top 15 fact types.

To better understand the cases where the neural models fail and succeed, we examine the distribution of F_1 scores over the top 15 fact types (see Table 6). Interestingly, when ranked by F_1 score we observe that fact types fall naturally into topically related groups:

1. position / occupation-related facts: *position played, position held, occupation*
2. membership-related facts: *member of political party, member of, member of sports team*
3. achievement-related facts: *award received, nominated for*
4. location-related facts: *country of citizenship, place of death, place of birth*

With the exception of *employer*, the overarching trend is that the model identifies the relevance of coarse-grained claims better than fine-grained claims (e.g occupations, political parties, and sports positions are much more likely to be shared between entities than birth and death places). This suggests that developing better methods for determining the relevance of fine-grained claims is a promising avenue for future research on this task.

4 Post-Modifier Generation

We move our focus to the main task of post-modifier generation.

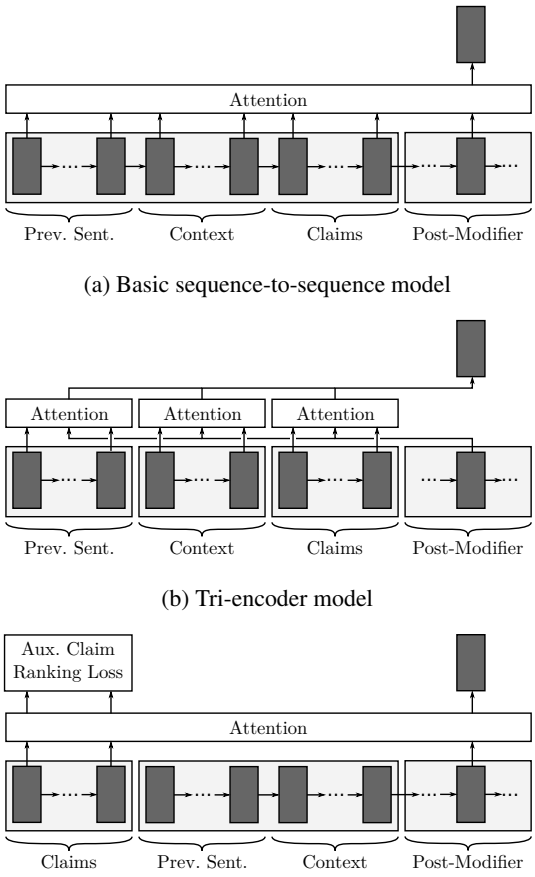
4.1 Methods

At its core, post-modifier generation involves producing a variable-length sequence output conditioned on two variable-length inputs: the words in the current and previous sentence (e.g. the context), and the collection of claims about the entity. Accordingly, the sequence-to-sequence (seq2seq) framework (Sutskever et al., 2014) is a natural fit for the task — we use it as the foundation for all of our baseline models. Since research has shown that attention (Bahdanau et al., 2015) and copy mechanisms (Gu et al., 2016) consistently improve seq2seq model performance, we use these in our baselines as well.

One choice that must be made when using this framework is how to combine the different inputs. The default approach we use is to concatenate the claim and context into a linear sequence of tokens during preprocessing (shown in Figure 4a). We also experiment with encoding the claims and each of the context sentences separately, then concatenating their vector representations before decoding. We refer to this as the *tri-encoder* approach (shown in Figure 4b).

As discussed earlier, selecting relevant claims is crucial to generating good post-modifiers. One way to incorporate claim selection is to use our baseline models from Section 3 to cut out irrelevant claims from the input before feeding them to the encoder (e.g. performing hard claim selection). This pipelined approach is not differentiable, and can suffer from cascading errors. An alternative way is to use the model's attention mechanism as a form of soft claim selection that attends only to the relevant claims. The drawback of this approach is that it does not make use of the available claim annotations, which are an important source of supervision.

Building on these observations, we propose an *end-to-end claim selection* model which incorporates an additional term to the loss function that encourages the claim-level attention probabilities to be higher for the identified relevant claims as shown in Figure 4c. The process for computing this loss term works as follows. We begin by summing together attention scores for tokens within claims to obtain a claim-level score. These scores



(a) Basic sequence-to-sequence model

(b) Tri-encoder model

(c) End-to-end claim selection model

Figure 4: PoMo Models for post-modifier generation. Grey boxes at the bottom represent individual encoder/decoder modules. (a) For baseline BiLSTM and transformer models all inputs are concatenated into one sequence. (b) The tri-encoder model has a separate encoder and attention for each type of input. The outputs of attention layers are concatenated together before generation. (c) The end-to-end claim selection model attends to only the claim embeddings and uses an auxiliary loss term to encourage high attention scores for relevant claims.

are then fed through a sigmoid activation function to obtain a soft claim selection probability. For each claim, we measure the binary cross entropy between the predicted selection probability and a binary variable indicating whether or not the claim was identified as relevant. The final loss term is the average of these binary cross entropies. Note that we do not use a copy mechanism in this model to avoid double-counting (since relevant claims were identified using word overlap).

4.2 Experiments

We experiment with two types of encoder/decoder modules: bidirectional LSTMs, and transform-

ers (Vaswani et al., 2017). We use a vocabulary of size 50K, truncate the maximum input sequence length to 500, and use a batch size of 32 in all experiments. To help models distinguish between claims and context we demarcate claim fields with special <claim>, <key>, and <value> tokens. We train all the models for 150k steps, and evaluate on the validation dataset every 10k steps. Evaluation is performed using the BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) translation metrics, and Precision, Recall and F_1 score of the predicted bag-of-words (omitting stopwords). The model with the highest F_1 score on the validation set is used during test time.

For the bidirectional LSTM, we use 2 hidden layers with 512 hidden units, 500-dimensional word embeddings, and apply dropout between layers with a keep probability of 0.7. Models are trained using stochastic gradient descent with a learning rate of 1.0. For the transformer model, we use 4 attention heads, 4 layers of transformer blocks with 64 hidden units for the encoder and the decoder, a penultimate hidden layer with 256 units, and 64-dimensional word embeddings. Transformer models are trained using Adam (Kingma and Ba, 2015) with an initial learning rate of 2.0, and a label smoothing (Szegedy et al., 2016) factor of 0.1 when calculating loss.

We perform a variety of experiments, the results of which are displayed in Table 7. In this table, *Transformer* and *BiLSTM* refer to models trained using the default approach to combining context and claims, while *Tri-encoder* refers to a BiLSTM model trained using the approach described in 4.1 (we do not train a transformer version since its performance is lackluster). Here are detailed descriptions of the experiments performed in each section:

- **All Claims**: Results for vanilla seq2seq models.
- **Oracle**: Hard claim selection is performed using the oracle relevant claims.
- **Neural Ranker** ($n = 10$): Hard claim selection is performed using the top-10 claims returned by the neural ranker baseline.
- **End-to-End Claim Selection**: Results for the end-to-end claim selection model.

In order to understand the relative contribution of the different inputs, we also include results for the BiLSTM model trained using either only the claims, or only the context sentences. In Figure 5 and 6, we show the performances by post-modifier and sentence lengths to examine the impact of the such variables.

Discussion of Quantitative Results Our results contain a few key findings. The first is that knowing the relevant claims is critical to obtaining state-of-the-art performance; even knowing only oracle claims is sufficient to perform better than all of the other baselines, although there is a still a large improvement when context is additionally provided. However, model-based approaches for claim selection do not seem to help: hard claim selection using the neural ranker performs just as well as the vanilla models, and our proposed approach for end-to-end claim selection has a negative impact. This motivates the need for more effective methods of claim selection. The decreasing performances of the BiLSTM seq2seq models by the increasing target post-modifier and sentence lengths show the difficulty of generating long texts and handling long input data. Finally, we observe that the transformer-based seq2seq models are not particularly well-suited to this task. In all cases their performance is inferior to the BiLSTM-based approaches. Large-scale, pre-trained transformer-based language models, such as GPT-2 (Radford et al., 2019) and BERT (Devlin et al., 2018), might be an interesting addition to the baselines, by framing the task as filling in the blanks for post-modifiers. However, when restricted to approaches that only use our dataset for training, we expect those based on language models to struggle due to the separation of entities among train, validation, and test.

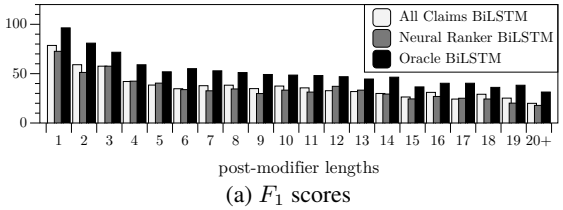
Qualitative Analysis A cursory examination of model predictions (see Table 8 for examples) provides insight into why post-modifier generation is a challenging task. One issue that consistently appears is temporal inconsistency between the target and generated post-modifiers. That is, the model may make an error since it is unaware of the time period that the article is written in (and also may not be aware of the periods of time for which a claim are true). For example, in the first instance in Table 8 the Oracle model predicts an almost correct post-modifier but misses the fact that Kenneth Clarke is a *former* Chancellor of the Exchequer. Another apparent issue is that models tend to generate shorter post-modifiers than humans. As is indicated in Figure 2a the post-modifiers in the dataset on average contain 5.8 tokens, whereas generated post-modifiers have only 3.8. Lastly, we

	Prec.	Rec.	F_1	BLEU	MET.
All Claims					
Transformer	41.9	22.2	29.0	7.0	12.1
Tri-Encoder	53.9	32.4	40.5	17.0	17.6
BiLSTM	51.1	34.7	41.4	19.4	18.8
Oracle					
Transformer	69.4	38.6	49.6	15.7	20.0
Tri-Encoder	68.8	47.3	56.1	24.0	24.5
BiLSTM	66.4	48.8	56.2	25.1	25.3
Neural Ranker ($n = 10$)					
Transformer	41.5	22.4	29.1	6.9	12.1
Tri-Encoder	53.5	34.1	41.6	17.6	18.3
BiLSTM	49.0	34.2	40.3	18.5	18.5
End-to-End Claim Selection					
BiLSTM	47.5	27.9	35.2	13.7	15.3
Context Only					
BiLSTM	13.3	8.5	10.3	3.4	6.2
Claims Only					
BiLSTM	47.3	28.5	35.6	13.5	15.0
Oracle Claims Only					
BiLSTM	63.8	44.7	52.5	21.3	22.7

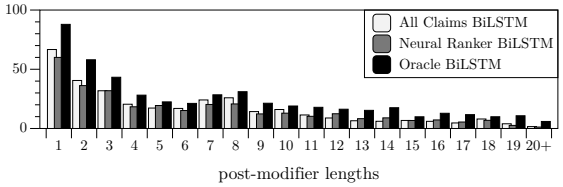
Table 7: Post modifier generation model performances with seq2seq models. Precision, recall and F_1 scores are computed ignoring stopwords.

observe that our quantitative evaluation metrics can be too strict. Take for example the second instance in Table 8. Here the content of the target and generated post-modifiers is almost exactly the same, however our metrics would give very low scores due to low overlap.

Human Evaluation We additionally evaluate the generated post-modifiers by performing a human evaluation using Amazon Mechanical Turk. We randomly select 500 instances from test set and show crowdworkers the sentence context, along with the true post-modifier and a generated one. For each instance, workers are asked to select the better phrase, or indicate that the two phrases are of equal quality. For the Oracle BiLSTM model, the true post-modifiers are preferred 46% of the time, while generated post-modifiers are preferred 43.2% of the time. For the Neural Ranker ($n = 10$) BiLSTM model, true post-modifiers are favored much more (57.60%) than the generated ones (20%). Consistent with our quantitative results, we see that claim selection is a crucial factor in this task. We also observe a few trends in the results. People tend to prefer generated post-modifiers over the ones written by professional journalists when they are shorter and to use more general terms without elaborating too much about the entity. In contrast, longer and more detailed human written post-modifiers are preferred when they are especially relevant to the rest of the sentence.

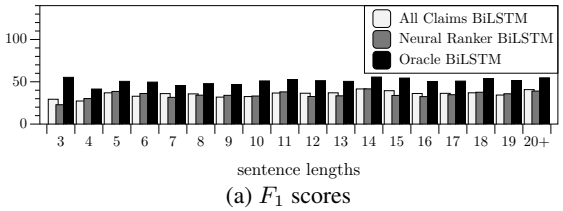


(a) F_1 scores

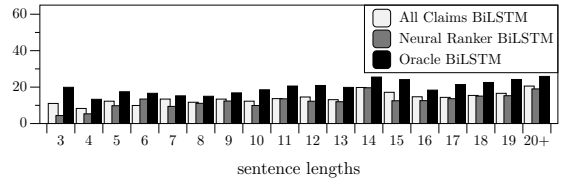


(b) BLEU scores

Figure 5: Performances by target post-modifier lengths of BiLSTM model. Post-modifiers with 20 or more tokens are put into one group, 20+.



(a) F_1 scores



(b) BLEU scores

Figure 6: Performances by input sentence lengths of BiLSTM model. Sentences with 20 or more tokens are put into one group, 20+.

5 Related Work

There is a large body of previous work on claim selection (Kukich, 1983; Duboue and McKeown, 2003; Reiter and Dale, 1997; Tanaka-Ishii et al., 1998; Barzilay and Lapata, 2005) and language generation from structured data (Reiter et al., 2005; Goldberg et al., 1994). Initially, hand-crafted grammars were employed for language generation, which later evolved to statistical machine translation style models (Wong and Mooney, 2007) or PCFG based models (Belz, 2008). More recently, the focus has shifted to learning both fact selection and language generation jointly (Liang et al., 2009; Angeli et al., 2010; Kim and Mooney, 2010; Lu and Ng, 2011; Konstas and Lapata, 2013).

Input	Sky News reported Thursday night that Kenneth Clarke , _____, had not yet decided whether to support Mr. Howard 's candidacy , raising the possibility the party could face a divisive battle for leadership .
Claims	+ (position held: *Chancellor of the Exchequer*) (position held: *Secretary of State for the Home Department*)
Target	a former chancellor of the exchequer
All Claims	the Home Secretary
Oracle	the Chancellor of the Exchequer
Input	" A lot of people think it 's something we just started , but we actually opened the season with our first drive using it against Indianapolis , " said Howard Ballard , _____.
Claims	+ (member of sports team: *Buffalo Bills*) + (position played on team / speciality: *offensive tackle*) (mass: *325 pound*) (height: *78 inch*)
Target	Buffalo 's robust , 6-foot-6-inch , 325-pound right tackle
All Claims & Oracle	the Bills ' offensive tackle

Table 8: **Challenging PoMo instances**. Two examples along with outputs of the best All Claims and Oracle models are displayed. Claims deemed relevant during dataset curation are prefaced with a +. In the first example, knowing the relevant claims helps the Oracle model produce an output that closely matches the Target, however lack of temporal information causes the model to miss the word *former*. In the second example, the All Claims and Oracle models produce the same post-modifier. Although it is similar to the Target in meaning, it receives a low score using our evaluation metrics. Futhermore, our data curation method fails to identify relevant claims.

Modern approaches employ neural networks to solve this problem end-to-end. Mei et al. (2016) utilize an encoder-decoder framework to map weather conditions to a weather forecast. Ahn et al. (2016) and Yang et al. (2017) introduce a new class of language models which are capable of entity coreference and copying facts from an external knowledge base. Building upon these models, Wiseman et al. (2017) introduce an auxiliary reconstruction loss which use the hidden states of the decoder to recover the facts used to generate the text. Liu et al. (2018) introduce a hierarchical attention model for fact selection, with the higher level focusing on which records in the table to select and the lower level focusing on which cells in a particular row to pay attention to.

In order to train complex neural models, the quest for larger datasets has become paramount. Lebret et al. (2016) introduce the WikiBio dataset containing Wikipedia articles of famous people and the corresponding infobox tables. One drawback of this dataset is that it is easily solved using template-based models. To address this issue, Wiseman et al. (2017) introduce the ROTOWire dataset, which contains summaries of basketball games that are very long and syntactically diverse. A comprehensive list of datasets is provided in Appendix B.

6 Conclusions and Future Work

Inspired by recent work on collaborative writing and data-to-text generation, we introduce post-modifier generation, a task that bridges the gap between these two fields. The task is to generate a factual description of an entity which fits within the context of a human written sentence. In order to promote research on this task we present PoMo, a large dataset of automatically extracted post-modifiers from news articles, aligned to the Wikidata knowledge graph. We study the performance of numerous strong baseline models on this dataset, with a particular focus on the specific sub-task of claim selection. Our results demonstrate that when relevant claims are known, sequence-to-sequence models are capable of generating post-modifiers which humans deem comparable in quality to ones written by professional journalists. However, according to both quantitative metrics and human judgment, performance is much lower when models must determine for themselves which claims are relevant. These experiments suggest plausible pathways to achieving human-level performance on this task that are both challenging and interesting problems for future research.

Acknowledgments

We would like to thank the Toyota Technological Institute at Chicago for hosting the Workshop on Collaborative and Knowledge-Backed Language Generation which initiated the efforts for this project. The authors would also like to thank David Yarowsky, Jason Eisner, Kevin Duh, Kyle Gorman, and Philipp Koehn for feedback on early ideas for post-modifier generation.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*.

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338. Association for Computational Linguistics.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A Smith. 2018. Creative writing with a machine in the loop: Case studies on slogans and stories. In *23rd International Conference on Intelligent User Interfaces*, pages 329–340. ACM.

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*.

Pablo A Duboue and Kathleen R McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 121–128. Association for Computational Linguistics.

Eli Goldberg, Norbert Driedger, and Richard I Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. In *Neural Computation*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org.

Joohyun Kim and Raymond J Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 145–150. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1611–1622. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, San Diego, California. Association for Computational Linguistics.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. 2014. Aida-light: High-throughput named-entity disambiguation. *LDOW*, 1184.

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*, pages 311–318. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Sudha Rao and Joel R. Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *NAACL-HLT*.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1-2):137–169.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6830–6841.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826.

Kumiko Tanaka-Ishii, Kôiti Hasida, and Itsuki Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 1282–1288. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Yuk Wah Wong and Raymond Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 172–179.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1850–1859, Copenhagen, Denmark. Association for Computational Linguistics.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7287–7298. Curran Associates, Inc.

Hong Yu and Mark O. Riedl. 2012. A sequential recommendation approach for interactive personalized story generation. In *AAMAS*.

A Additional Claim Selection Materials

Table 9 lists the evaluation results of most-common claim baseline for n, the number of claims to predict, from 1 to 5. We obtain highest F1 with $n = 4$.

n	Precision	Recall	F1
1	57.7	31.8	41.0
2	49.6	38.3	43.2
3	43.4	45.2	44.3
4	39.9	51.6	45.0
5	36.0	56.8	44.0

Table 9: Evaluation metrics for most-common claim baseline for different values of n.

Neural baseline shows improved performance compared to most-common claim baseline, showing its best performance when $n = 2$.

n	Precision	Recall	F1
1	75.2	42.4	54.3
2	66.5	62.7	64.5
3	56.0	69.6	62.1
4	48.7	76.2	59.4

Table 10: Evaluation metrics for neural baseline for different values of n.

B Existing Data-to-Text Datasets

Table 11 provides a comprehensive list of data-to-text datasets. PoMo presents a different set of challenges from these datasets. While the target text is shorter and less diverse, the task adds an additional challenge of figuring out which claims to use, a task which our evaluation shows is quite challenging.

Dataset	**Size**	**Domain of structured data to language**
WEATHER.GOV	29.5k	Weather conditions to forecast report
ALLRECIPES	31k	Table of ingredients to recipes
ROBOCUP	1.5k	Game statistics to summaries
ROTOWIRE	4.9k	Basketball statistics to game summaries
WIKIBIO	728k	Infobox to Wikipedia biography articles
SBNations	10.9K	Game statistic to fan written summaries
WikiFacts	40k	Freebase /film/actor facts to Wiki description of actor

Table 11: A comparative analysis of various datasets.