Challenges in Data-to-Document Generation

Sam Wiseman and **Stuart M. Shieber** and **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA
{swiseman,shieber,srush}@seas.harvard.edu

Abstract

Recent neural models have shown significant progress on the problem of generating short descriptive texts conditioned on a small number of database records. In this work, we suggest a slightly more difficult data-to-text generation task, and investigate how effective current approaches are on this task. In particular, we introduce a new, large-scale corpus of data records paired with descriptive documents, propose a series of extractive evaluation methods for analyzing performance, and obtain baseline results using current neural generation methods. Experiments show that these models produce fluent text, but fail to convincingly approximate human-generated documents. Moreover, even templated baselines exceed the performance of these neural models on some metrics, though copy- and reconstruction-based extensions lead to noticeable improvements.

1 Introduction

Over the past several years, neural text generation systems have shown impressive performance on tasks such as machine translation and summarization. As neural systems begin to move toward generating longer outputs in response to longer and more complicated inputs, however, the generated texts begin to display reference errors, inter-sentence incoherence, and a lack of fidelity to the source material. The goal of this paper is to suggest a particular, long-form generation task in which these challenges may be fruitfully explored, to provide a publically available dataset for this task, to suggest some automatic evaluation metrics, and finally to establish how current, neural text generation methods perform on this task.

A classic problem in natural-language generation (NLG) (Kukich, 1983; McKeown, 1992; Reiter and Dale, 1997) involves taking structured data, such as a table, as input, and producing text that adequately and fluently describes this data as output. Unlike machine translation, which aims for a complete transduction of the sentence to be translated, this form of NLG is typically taken to require addressing (at least) two separate challenges: *what to say*, the selection of an appropriate subset of the input data to discuss, and *how to say it*, the surface realization of a generation (Reiter and Dale, 1997; Jurafsky and Martin, 2014). Traditionally, these two challenges have been modularized and handled separately by generation systems. However, neural generation systems, which are typically trained end-to-end as conditional language models (Mikolov et al., 2010; Sutskever et al., 2011, 2014), blur this distinction.

In this context, we believe the problem of generating multi-sentence summaries of tables or database records to be a reasonable next-problem for neural techniques to tackle as they begin to consider more difficult NLG tasks. In particular, we would like this generation task to have the following two properties: (1) it is relatively easy to obtain fairly clean summaries and their corresponding databases for dataset construction, and (2) the summaries should be primarily focused on conveying the information in the database. This latter property ensures that the task is somewhat congenial to a standard encoder-decoder approach, and, more importantly, that it is reasonable to *evaluate* generations in terms of their fidelity to the database.

One task that meets these criteria is that of generating summaries of sports games from associated box-score data, and there is indeed a

long history of NLG work that generates sports game summaries (Robin, 1994; Tanaka-Ishii et al., 1998; Barzilay and Lapata, 2005). To this end, we make the following contributions:

- We introduce a new large-scale corpus consisting of textual descriptions of basketball games paired with extensive statistical tables. This dataset is sufficiently large that fully data-driven approaches might be sufficient.
- We introduce a series of extractive evaluation models to automatically evaluate output generation performance, exploiting the fact that post-hoc information extraction is significantly easier than generation itself.
- We apply a series of state-of-the-art neural methods, as well as a simple templated generation system, to our data-to-document generation task in order to establish baselines and study their generations.

Our experiments indicate that neural systems are quite good at producing fluent outputs and generally score well on standard word-match metrics, but perform quite poorly at content selection and at capturing long-term structure. While the use of copy-based models and additional reconstruction terms in the training loss can lead to improvements in BLEU and in our proposed extractive evaluations, current models are still quite far from producing human-level output, and are significantly worse than templated systems in terms of content selection and realization. Overall, we believe this problem of data-to-document generation highlights important remaining challenges in neural generation systems, and the use of extractive evaluation reveals significant issues hidden by standard automatic metrics.

2 Data-to-Text Datasets

We consider the problem of generating descriptive text from database records. Following the notation in Liang et al. (2009), let $\boldsymbol{s} = \{r_j\}_{j=1}^{J}$ be a set of records, where for each $r \in \boldsymbol{s}$ we define $r.t \in \mathcal{T}$ to be the *type* of r, and we assume each r to be a binarized relation, where $r.e$ and $r.m$ are a record's entity and value, respectively. For example, a database recording statistics for a basketball game might have a record r such that $r.t$ = PTS, $r.e$ = RUSSELL WESTBROOK, and $r.m = 50$. In this case, $r.e$ gives the player in question, and $r.m$ gives the number of points the player scored. From these records, we are interested in generating descriptive text, $\hat{y}_{1:T} = \hat{y}_1, \ldots, \hat{y}_T$ of T words such that $\hat{y}_{1:T}$ is an adequate and fluent summary of $\boldsymbol{s}$. A dataset for training data-to-document systems typically consists of $(\boldsymbol{s}, y_{1:T})$ pairs, where $y_{1:T}$ is a document consisting of a gold (i.e., human generated) summary for database $\boldsymbol{s}$.

Several benchmark datasets have been used in recent years for the text generation task, the most popular of these being WEATHERGOV (Liang et al., 2009) and ROBOCUP (Chen and Mooney, 2008). Recently, neural generation systems have show strong results on these datasets, with the system of Mei et al. (2016) achieving BLEU scores in the 60s and 70s on WEATHERGOV, and BLEU scores of almost 30 even on the smaller ROBOCUP dataset. These results are quite promising, and suggest that neural models are a good fit for text generation. However, the statistics of these datasets, shown in Table 1, indicate that these datasets use relatively simple language and record structure. Furthermore, there is reason to believe that WEATHERGOV is at least partially machine-generated (Reiter, 2017). More recently, Lebret et al. (2016) introduced the WIKIBIO dataset, which is at least an order of magnitude larger in terms of number of tokens and record types. However, as shown in Table 1, this dataset too only contains short (single-sentence) generations, and relatively few records per generation. As such, we believe that early success on these datasets is not yet sufficient for testing the desired linguistic capabilities of text generation at a document-scale.

With this challenge in mind, we introduce a new dataset for data-to-document text generation, available at https://github.com/harvardnlp/boxscore-data. The dataset is intended to be comparable to WEATHERGOV in terms of token count, but to have significantly longer target texts, a larger vocabulary space, and to require more difficult content selection.

The dataset consists of two sources of articles summarizing NBA basketball games, paired with their corresponding box- and line-score tables. The data statistics of these two sources, ROTOWIRE and SBNATION, are also shown in Table 1. The first dataset, ROTOWIRE, uses professionally written, medium length game summaries

TEAM	WIN	LOSS	PTS	FG_PCT	RB	AS ...
Heat	11	12	103	49	47	27
Hawks	7	15	95	43	33	20

PLAYER	AS	RB	PT	FG	FGA	CITY ...
Tyler Johnson	5	2	27	8	16	Miami
Dwight Howard	4	17	23	9	11	Atlanta
Paul Millsap	2	9	21	8	12	Atlanta
Goran Dragic	4	2	21	8	17	Miami
Wayne Ellington	2	3	19	7	15	Miami
Dennis Schroder	7	4	17	8	15	Atlanta
Rodney McGruder	5	5	11	3	8	Miami
Thabo Sefolosha	5	5	10	5	11	Atlanta
Kyle Korver	5	3	9	3	9	Atlanta
...						

The Atlanta Hawks defeated the Miami Heat , 103 - 95 , at Philips Arena on Wednesday . Atlanta was in desperate need of a win and they were able to take care of a shorthanded Miami team here . Defense was key for the Hawks , as they held the Heat to 42 percent shooting and forced them to commit 16 turnovers . Atlanta also dominated in the paint , winning the rebounding battle , 47 - 34 , and outscoring them in the paint 58 - 26.The Hawks shot 49 percent from the field and assisted on 27 of their 43 made baskets . This was a near wire - to - wire win for the Hawks , as Miami held just one lead in the first five minutes . Miami (7 - 15) are as beat - up as anyone right now and it 's taking a toll on the heavily used starters . Hassan Whiteside really struggled in this game , as he amassed eight points , 12 rebounds and one blocks on 4 - of - 12 shooting ...

Figure 1: An example data-record and document pair from the ROTOWIRE dataset. We show a subset of the game's records (there are 628 in total), and a selection from the gold document. The document mentions only a select subset of the records, but may express them in a complicated manner. In addition to capturing the writing style, a generation system should select similar record content, express it clearly, and order it appropriately.

	RC	WG	WB	RW	SBN
Vocab	409	394	400K	11.3K	68.6K
Tokens	11K	0.9M	19M	1.6M	8.8M
Examples	1.9K	22.1K	728K	4.9K	10.9K
Avg Len	5.7	28.7	26.1	337.1	805.4
Rec. Types	4	10	1.7K	39	39
Avg Records	2.2	191	19.7	628	628

Table 1: Vocabulary size, number of total tokens, number of distinct examples, average generation length, total number of record types, and average number of records per example for the ROBOCUP (RC), WEATHERGOV (WG), WIKIBIO (WB), ROTOWIRE (RW), and SBNATION (SBN) datasets.

targeted at fantasy basketball fans. The writing is colloquial, but relatively well structured, and targets an audience primarily interested in game statistics. The second dataset, SBNATION, uses fan-written summaries targeted at other fans. This dataset is significantly larger, but also much more challenging, as the language is very informal, and often tangential to the statistics themselves. We show some sample text from ROTOWIRE in Figure 1. Our primary focus will be on the ROTOWIRE data.

3 Evaluating Document Generation

We begin by discussing the evaluation of generated documents, since both the task we introduce and the evaluation methods we propose are motivated by some of the shortcomings of current approaches to evaluation. Text generation systems are typically evaluated using a combination of automatic measures, such as BLEU (Papineni et al., 2002), and human evaluation. While BLEU is perhaps a reasonably effective way of evaluating short-form text generation, we found it to be unsatisfactory for document generation. In particular, we note that it primarily rewards fluent text generation, rather than generations that capture the most important information in the database, or that report the information in a particularly coherent way. While human evaluation, on the other hand, is likely ultimately necessary for evaluating generations (Liu et al., 2016; Wu et al., 2016), it is much less convenient than using automatic metrics. Furthermore, we believe that current text generations are sufficiently bad in sufficiently obvious ways that automatic metrics can still be of use in evaluation, and we are not yet at the point of needing to rely solely on human evaluators.

3.1 Extractive Evaluation

To address this evaluation challenge, we begin with the intuition that assessing document quality is easier than document generation. In particular, it is much easier to automatically extract information from documents than to generate documents that accurately convey desired information. As such, simple, high-precision information extraction models can serve as the basis for assessing and better understanding the quality of automatic

generations. We emphasize that such an evaluation scheme is most appropriate when evaluating generations (such as basketball game summaries) that are primarily intended to summarize information. While many generation problems do not fall into this category, we believe this to be an interesting category, and one worth focusing on *because* it is amenable to this sort of evaluation.

To see how a simple information extraction system might work, consider the document in Figure 1. We may first extract candidate entity (player, team, and city) and value (number and certain string) pairs $r.e, r.m$ that appear in the text, and then predict the type $r.t$ (or none) of each candidate pair. For example, we might extract the entity-value pair ("Miami Heat", "95") from the first sentence in Figure 1, and then predict that the *type* of this pair is POINTS, giving us an extracted record r such that $(r.e, r.m, r.t) =$ (MIAMI HEAT, 95, POINTS). Indeed, many relation extraction systems reduce relation extraction to multi-class classification precisely in this way (Zhang, 2004; Zhou et al., 2008; Zeng et al., 2014; dos Santos et al., 2015).

More concretely, given a document $\hat{y}_{1:T}$, we consider all pairs of word-spans in each sentence that represent possible entities e and values m. We then model $p(r.t \mid e, m; \boldsymbol{\theta})$ for each pair, using $r.t = \epsilon$ to indicate unrelated pairs. We use architectures similar to those discussed in Collobert et al. (2011) and dos Santos et al. (2015) to parameterize this probability; full details are given in the Appendix.

Importantly, we note that the $(\boldsymbol{s}, y_{1:T})$ pairs typically used for training data-to-document systems are also sufficient for training the information extraction model presented above, since we can obtain (partial) supervision by simply checking whether a candidate record lexically matches a record in $\boldsymbol{s}$.[1] However, since there may be multiple records $r \in \boldsymbol{s}$ with the same e and m but with different types $r.t$, we will not always be able to determine the type of a given entity-value pair found in the text. We therefore train our classifier to minimize a latent-variable loss: for all document spans e and m, with observed types $t(e, m) = \{r.t : r \in \boldsymbol{s}, r.e = e, r.m = m\}$ (possibly $\{\epsilon\}$), we minimize

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{e,m} \log \sum_{t' \in t(e,m)} p(r.t = t' \mid e, m; \boldsymbol{\theta}).$$

We find that this simple system trained in this way is quite accurate at predicting relations. On the ROTOWIRE data it achieves over 90% accuracy on held-out data, and recalls approximately 60% of the relations licensed by the records.

3.2 Comparing Generations

With a sufficiently precise relation extraction system, we can begin to evaluate how well an automatic generation $\hat{y}_{1:T}$ has captured the information in a set of records $\boldsymbol{s}$. In particular, since the predictions of a precise information extraction system serve to align entity-mention pairs in the text with database records, this alignment can be used both to evaluate a generation's content selection ("what the generation says"), as well as content placement ("how the generation says it").

We consider in particular three induced metrics:

- Content Selection (CS): precision and recall of unique relations r extracted from $\hat{y}_{1:T}$ that are also extracted from $y_{1:T}$. This measures how well the generated document matches the gold document in terms of selecting which records to generate.
- Relation Generation (RG): precision and number of unique relations r extracted from $\hat{y}_{1:T}$ that also appear in $\boldsymbol{s}$. This measures how well the system is able to generate text containing factual (i.e., correct) records.
- Content Ordering (CO): normalized Damerau-Levenshtein Distance (Brill and Moore, 2000)[2] between the sequences of records extracted from $y_{1:T}$ and that extracted from $\hat{y}_{1:T}$. This measures how well the system orders the records it chooses to discuss.

We note that CS primarily targets the "what to say" aspect of evaluation, CO targets the "how to say it" aspect, and RG targets both.

We conclude this section by contrasting the automatic evaluation we have proposed with

[1] Alternative approaches explicitly align the document with the table for this task (Liang et al., 2009).

[2] DLD is a variant of Levenshtein distance that allows transpositions of elements; it is useful in comparing the ordering of sequences that may not be permutations of the same set (which is a requirement for measures like Kendall's Tau).

recently proposed *adversarial evaluation* approaches, which also advocate automatic metrics backed by classification (Bowman et al., 2016; Kannan and Vinyals, 2016; Li et al., 2017). Unlike adversarial evaluation, which uses a black-box classifier to determine the quality of a generation, our metrics are defined with respect to the predictions of an information extraction system. Accordingly, our metrics are quite interpretable, since by construction it is always possible to determine which fact (i.e., entity-value pair) in the generation is determined by the extractor to not match the database or the gold generation.

4 Neural Data-to-Document Models

In this section we briefly describe the neural generation methods we apply to the proposed task. As a base model we utilize the now standard attention-based encoder-decoder model (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015). We also experiment with several recent extensions to this model, including copy-based generation, and training with a source reconstruction term in the loss (in addition to the standard per-target-word loss).

Base Model For our base model, we map each record $r \in \boldsymbol{s}$ into a vector $\tilde{\boldsymbol{r}}$ by first embedding $r.t$ (e.g., POINTS), $r.e$ (e.g., RUSSELL WESTBROOK), and $r.m$ (e.g., 50), and then applying a 1-layer MLP (similar to Yang et al. (2016)).[3] Our source data-records are then represented as $\tilde{\boldsymbol{s}} = \{\tilde{\boldsymbol{r}}_j\}_{j=1}^{J}$. Given $\tilde{\boldsymbol{s}}$, we use an LSTM decoder with attention and input-feeding, in the style of Luong et al. (2015), to compute the probability of each target word, conditioned on the previous words and on $\boldsymbol{s}$. The model is trained end-to-end to minimize the negative log-likelihood of the words in the gold text $y_{1:T}$ given corresponding source material $\boldsymbol{s}$.

Copying There has been a surge of recent work involving augmenting encoder-decoder models to copy words directly from the source material on which they condition (Gu et al., 2016; Gülçehre et al., 2016; Merity et al., 2016; Jia and Liang, 2016; Yang et al., 2016). These models typically introduce an additional binary variable z_t into the per-timestep target word distribution, which indicates whether the target word $\hat{y}_t$ is copied from the source or generated:

$$p(\hat{y}_t \mid \hat{y}_{1:t-1}, \boldsymbol{s}) = \sum_{z \in \{0,1\}} p(\hat{y}_t, z_t = z \mid \hat{y}_{1:t-1}, \boldsymbol{s}).$$

In our case, we assume that target words are copied from the *value* portion of a record r; that is, a copy implies $\hat{y}_t = r.m$ for some r and t.

Joint Copy Model The models of Gu et al. (2016) and Yang et al. (2016) parameterize the *joint* distribution table over $\hat{y}_t$ and z_t directly:

$$p(\hat{y}_t, z_t \mid \hat{y}_{1:t-1}, \boldsymbol{s}) \propto \begin{cases} \text{copy}(\hat{y}_t, \hat{y}_{1:t-1}, \boldsymbol{s}) & z_t = 1, \hat{y}_t \in \boldsymbol{s} \\ 0 & z_t = 1, \hat{y}_t \notin \boldsymbol{s} \\ \text{gen}(\hat{y}_t, \hat{y}_{1:t-1}, \boldsymbol{s}) & z_t = 0, \end{cases}$$

where copy and gen are functions parameterized in terms of the decoder RNN's hidden state that assign scores to words, and where the notation $\hat{y}_t \in \boldsymbol{s}$ indicates that $\hat{y}_t$ is equal to $r.m$ for some $r \in \boldsymbol{s}$.

Conditional Copy Model Gülçehre et al. (2016), on the other hand, decompose the joint probability as:

$$p(\hat{y}_t, z_t \mid \hat{y}_{1:t-1}, \boldsymbol{s}) = \begin{cases} p_{\text{copy}}(\hat{y}_t \mid z_t, \hat{y}_{1:t-1}, \boldsymbol{s})\, p(z_t \mid \hat{y}_{1:t-1}, \boldsymbol{s}) & z_t{=}1 \\ p_{\text{gen}}(\hat{y}_t \mid z_t, \hat{y}_{1:t-1}, \boldsymbol{s})\, p(z_t \mid \hat{y}_{1:t-1}, \boldsymbol{s}) & z_t{=}0, \end{cases}$$

where an MLP is used to model $p(z_t \mid \hat{y}_{1:t-1}, \boldsymbol{s})$.

Models with copy-decoders may be trained to minimize the negative log marginal probability, marginalizing out the latent-variable z_t (Gu et al., 2016; Yang et al., 2016; Merity et al., 2016). However, if it is known which target words y_t are copied, it is possible to train with a loss that does not marginalize out the latent z_t. Gülçehre et al. (2016), for instance, assume that any target word y_t that also appears in the source is copied, and train to minimize the negative joint log-likelihood of the y_t and z_t.

In applying such a loss in our case, we again note that there may be multiple records r such that $r.m$ appears in $\hat{y}_{1:T}$. Accordingly, we slightly modify the p_{copy} portion of the loss of Gülçehre et al. (2016) to sum over all matched records. In particular, we model the probability of relations $r \in \boldsymbol{s}$ such that $r.m = y_t$ and $r.e$ is in the same sentence as $r.m$. Letting $r(y_t) =$

[3] We also include an additional feature for whether the player is on the home- or away-team.

$\{r \in \boldsymbol{s} : r.m = y_t, \text{same}-\text{sentence}(r.e, r.m)\}$, we have:

$$p_{\text{copy}}(y_t \mid z_t, y_{1:t-1}, \boldsymbol{s}) = \sum_{r \in r(y_t)} p(r \mid z_t, y_{1:t-1}, \boldsymbol{s}).$$

We note here that the key distinction for our purposes between the Joint Copy model and the Conditional Copy model is that the latter *conditions* on whether there is a copy or not, and so in p_{copy} the source records compete only with each other. In the Joint Copy model, however, the source records also compete with words that cannot be copied. As a result, training the Conditional Copy model with the supervised loss of Gülçehre et al. (2016) can be seen as training with a word-level reconstruction loss, where the decoder is trained to choose the record in $\boldsymbol{s}$ that gives rise to y_t.

Reconstruction Losses Reconstruction-based techniques can also be applied at the document- or sentence-level during training. One simple approach to this problem is to utilize the hidden states of the decoder to try to reconstruct the database. A fully differentiable approach using the decoder hidden states has recently been successfully applied to neural machine translation by Tu et al. (2017). Unlike copying, this method is applied only at training, and attempts to learn decoder hidden states with broader coverage of the input data.

In adopting this reconstruction approach we segment the decoder hidden states $\boldsymbol{h}_t$ into $\lceil \frac{T}{B} \rceil$ contiguous blocks of size at most B. Denoting a single one of these hidden state blocks as $\boldsymbol{b}_i$, we attempt to predict each field value in some record $r \in \boldsymbol{s}$ from $\boldsymbol{b}_i$. We define $p(r.e, r.m \mid \boldsymbol{b}_i)$, the probability of the entity and value in record r given $\boldsymbol{b}_i$, to be $\text{softmax}(f(\boldsymbol{b}_i))$, where f is a parameterized function of $\boldsymbol{b}_i$, which in our experiments utilize a convolutional layer followed by an MLP; full details are given in the Appendix. We further extend this idea and predict K records in $\boldsymbol{s}$ from $\boldsymbol{b}_i$, rather than one. We can train with the following reconstruction loss for a particular $\boldsymbol{b}_i$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= -\sum_{k=1}^{K} \min_{r \in \boldsymbol{s}} \log p_k(r \mid \boldsymbol{b}_i; \boldsymbol{\theta}) \\ &= -\sum_{k=1}^{K} \min_{r \in \boldsymbol{s}} \sum_{x \in \{e,m,t\}} \log p_k(r.x \mid \boldsymbol{b}_i; \boldsymbol{\theta}),\end{aligned}$$

where p_k is the k'th predicted distribution over records, and where we have modeled each component of r independently. This loss attempts to make the *most* probable record in $\boldsymbol{s}$ given $\boldsymbol{b}_i$ more probable. We found that augmenting the above loss with a term that penalizes the total variation distance (TVD) between the p_k to be helpful.[4] Both $\mathcal{L}(\boldsymbol{\theta})$ and the TVD term are simply added to the standard negative log-likelihood objective at training time.

5 Experimental Methods

In this section we highlight a few important details of our models and methods; full details are in the Appendix. For our ROTOWIRE models, the record encoder produces $\tilde{\boldsymbol{r}}_j$ in $\mathbb{R}^{600}$, and we use a 2-layer LSTM decoder with hidden states of the same size as the $\tilde{\boldsymbol{r}}_j$, and dot-product attention and input-feeding in the style of Luong et al. (2015). Unlike past work, we use two identically structured attention layers, one to compute the standard generation probabilities (gen or p_{gen}), and one to produce the scores used in copy or p_{copy}.

We train the generation models using SGD and truncated BPTT (Elman, 1990; Mikolov et al., 2010), as in language modeling. That is, we split each $y_{1:T}$ into contiguous blocks of length 100, and backprop both the gradients with respect to the current block as well as with respect to the encoder parameters for each block.

Our extractive evaluator consists of an ensemble of 3 single-layer convolutional and 3 single-layer bidirectional LSTM models. The convolutional models concatenate convolutions with kernel widths 2, 3, and 5, and 200 feature maps in the style of (Kim, 2014). Both models are trained with SGD.

Templatized Generator In addition to neural baselines, we also use a problem-specific, template-based generator. The template-based generator first emits a sentence about the teams playing in the game, using a templatized sentence taken from the training set:

> The `<team1>` (`<wins1>-<losses1>`) defeated the `<team2>` (`<wins2>-<losses2>`) `<pts1>-<pts2>`.

[4] Penalizing the TVD between the p_k might be useful if, for instance, K is too large, and only a smaller number of records can be predicted from $\boldsymbol{b}_i$. We also experimented with *encouraging*, rather than penalizing the TVD between the p_k, which might make sense if we were worried about ensuring the p_k captured different records.

Then, 6 player-specific sentences of the following form are emitted (again adapting a simple sentence from the training set):

> <player> scored <pts> points (<fgm>-<fga> FG, <tpm>-<tpa> 3PT, <ftm>-<fta> FT) to go with <reb> rebounds.

The 6 highest-scoring players in the game are used to fill in the above template. Finally, a typical end sentence is emitted:

> The <team1>' next game will be at home against the Dallas Mavericks, while the <team2> will travel to play the Bulls.

Code implementing all models can be found at https://github.com/harvardnlp/data2text. Our encoder-decoder models are based on OpenNMT (Klein et al., 2017).

6 Results

We found that all models performed quite poorly on the SBNATION data, with the best model achieving a validation perplexity of 33.34 and a BLEU score of 1.78. This poor performance is presumably attributable to the noisy quality of the SBNATION data, and the fact that many documents in the dataset focus on information not in the box- and line-scores. Accordingly, we focus on ROTOWIRE in what follows.

The main results for the ROTOWIRE dataset are shown in Table 2, which shows the performance of the models in Section 4 in terms of the metrics defined in Section 3.2, as well as in terms of perplexity and BLEU.

6.1 Discussion

There are several interesting relationships in the development portion of Table 2. First we note that the Template model scores very poorly on BLEU, but does quite well on the extractive metrics, providing an upper-bound for how domain knowledge could help content selection and generation. All the neural models make significant improvements in terms of BLEU score, with the conditional copying with beam search performing the best, even though all the neural models achieve roughly the same perplexity.

The extractive metrics provide further insight into the behavior of the models. We first note that on the gold documents $y_{1:T}$, the extractive model reaches 92% precision. Using the Joint Copy model, generation only has a record generation (RG) precision of 47% indicating that relationships are often generated incorrectly. The best Conditional Copy system improves this value to 71%, a significant improvement and potentially the cause of the improved BLEU score, but still far below gold.

> The Utah Jazz (38 - 26) defeated the Houston Rockets (38 - 26) 117 - 91 on Wednesday at Energy Solutions Arena in Salt Lake City . The Jazz got out to a quick start in this one , out - scoring the Rockets 31 - 15 in the first quarter alone . Along with the quick start , the Rockets were the superior shooters in this game , going 54 percent from the field and 43 percent from the three - point line , while the Jazz went 38 percent from the floor and a meager 19 percent from deep . The Rockets were able to out - rebound the Rockets 49 - 49 , giving them just enough of an advantage to secure the victory in front of their home crowd . The Jazz were led by the duo of Derrick Favors and James Harden . Favors went 2 - for - 6 from the field and 0 - for - 1 from the three - point line to score a game - high of 15 points , while also adding four rebounds and four assists ....

Figure 2: Example document generated by the Conditional Copy system with a beam of size 5. Text that accurately reflects a record in the associated box- or line-score is highlighted in blue, and erroneous text is highlighted in red.

Notably, content selection (CS) and content ordering (CO) seem to have no correlation at all with BLEU. There is some improvement with CS for the conditional model or reconstruction loss, but not much change as we move to beam search. CO actually gets worse as beam search is utilized, possibly a side effect of generating more records (RG#). The fact that these scores are much worse than the simple templated model indicates that further research is needed into better copying alone for content selection and better long term content ordering models.

Test results are consistent with development results, indicating that the Conditional Copy model is most effective at BLEU, RG, and CS, and that reconstruction is quite helpful for improving the joint model.

6.2 Human Evaluation

We also undertook two human evaluation studies, using Amazon Mechanical Turk. The first study attempted to determine whether generations considered to be more precise by our metrics were also considered more precise by human raters. To accomplish this, raters were presented with a particular NBA game's box score and line score, as well as with (randomly selected) sentences from summaries generated by our different models for

		Development						
		RG		CS		CO	PPL	BLEU
Beam	Model	P%	#	P%	R%	DLD%		
	Gold	91.77	12.84	100	100	100	1.00	100
	Template	99.35	49.7	18.28	65.52	12.2	N/A	6.87
B=1	Joint Copy	47.55	7.53	20.53	22.49	8.28	7.46	10.41
	Joint Copy + Rec	57.81	8.31	23.65	23.30	9.02	7.25	10.00
	Joint Copy + Rec + TVD	60.69	8.95	23.63	24.10	8.84	7.22	12.78
	Conditional Copy	68.94	9.09	25.15	22.94	9.00	7.44	13.31
B=5	Joint Copy	47.00	10.67	16.52	26.08	7.28	7.46	10.23
	Joint Copy + Rec	62.11	10.90	21.36	26.26	9.07	7.25	10.85
	Joint Copy + Rec + TVD	57.51	11.41	18.28	25.27	8.05	7.22	12.04
	Conditional Copy	71.07	12.61	21.90	27.27	8.70	7.44	14.46
					Test			
	Template	99.30	49.61	18.50	64.70	8.04	N/A	6.78
	Joint Copy + Rec (B=5)	61.23	11.02	21.56	26.45	9.06	7.47	10.88
	Joint Copy + Rec + TVD (B=1)	60.27	9.18	23.11	23.69	8.48	7.42	12.96
	Conditional Copy (B=5)	71.82	12.82	22.17	27.16	8.68	7.67	14.49

Table 2: Performance of induced metrics on gold and system outputs of RotoWire development and test data. Columns indicate Record Generation (RG) precision and count, Content Selection (CS) precision and recall, Count Ordering (CO) in normalized Damerau-Levenshtein distance, perplexity, and BLEU. These first three metrics are described in Section 3.2. Models compare Joint and Conditional Copy also with addition Reconstruction loss and Total Variation Distance extensions (described in Section 4).

those games. Raters were then asked to count how many facts in each sentence were supported by records in the box or line scores, and how many were contradicted. We randomly selected 20 distinct games to present to raters, and a total of 20 generated sentences per game were evaluated by raters. The left two columns of Table 3 contain the average numbers of supporting and contradicting facts per sentence as determined by the raters, for each model. We see that these results are generally in line with the RG and CS metrics, with the Conditional Copy model having the highest number of supporting facts, and the reconstruction terms significantly improving the Joint Copy models.

Using a Tukey HSD post-hoc analysis of an ANOVA with the number of contradicting facts as the dependent variable and the generating model and rater id as independent variables, we found significant ($p < 0.01$) pairwise differences in contradictory facts between the gold generations and all models except "Copy+Rec+TVD," as well as a significant difference between "Copy+Rec+TVD" and "Copy". We similarly found a significant pairwise difference between "Copy+Rec+TVD" and "Copy" for number of supporting facts.

Our second study attempted to determine whether generated summaries differed in terms of how natural their *ordering* of records (as captured, for instance, by the DLD metric) is. To test this, we presented raters with random summaries generated by our models and asked them to rate the naturalness of the ordering of facts in the summaries on a 1-7 Likert scale. 30 random summaries were used in this experiment, each rated 3 times by distinct raters. The average Likert ratings are shown in the rightmost column of Table 3. While it is encouraging that the gold summaries received a higher average score than the generated summaries (and that the reconstruction term again improved the Joint Copy model), a Tukey HSD analysis similar to the one presented above revealed no significant pairwise differences.

	# Supp.	# Cont.	Order Rat.
Gold	2.04	0.70	5.19
Joint Copy	1.65	2.31	3.90
Joint Copy + Rec	2.33	1.83	4.43
Joint Copy + Rec +TVD	2.43	1.16	4.18
Conditional Copy	3.05	1.48	4.03

Table 3: Average rater judgment of number of box score fields supporting (left column) or contradicting (middle column) a generated sentence, and average rater Likert rating for the naturalness of a summary's ordering (right column). All generations use B=1.

6.3 Qualitative Example

Figure 2 shows a document generated by the Conditional Copy model, using a beam of size 5. This particular generation evidently has several nice

properties: it nicely learns the colloquial style of the text, correctly using idioms such as "19 percent from deep." It is also partially accurate in its use of the records; we highlight in blue when it generates text that is licensed by a record in the associated box- and line-scores.

At the same time, the generation also contains major logical errors. First, there are basic copying mistakes, such as flipping the teams' win/loss records. The system also makes obvious semantic errors; for instance, it generates the phrase "the Rockets were able to out-rebound the Rockets." Finally, we see the model hallucinates factual statements, such as "in front of their home crowd," which is presumably likely according to the language model, but ultimately incorrect (and not supported by anything in the box- or line-scores). In practice, our proposed extractive evaluation will pick up on many errors in this passage. For instance, "four assists" is an RG error, repeating the Rockets' rebounds could manifest in a lower CO score, and incorrectly indicating the win/loss records is a CS error.

7 Related Work

In this section we note additional related work not noted throughout. Natural language generation has been studied for decades (Kukich, 1983; McKeown, 1992; Reiter and Dale, 1997), and generating summaries of sports games has been a topic of interest for almost as long (Robin, 1994; Tanaka-Ishii et al., 1998; Barzilay and Lapata, 2005).

Historically, research has focused on both content selection ("what to say") (Kukich, 1983; McKeown, 1992; Reiter and Dale, 1997; Duboue and McKeown, 2003; Barzilay and Lapata, 2005), and surface realization ("how to say it") (Goldberg et al., 1994; Reiter et al., 2005) with earlier work using (hand-built) grammars, and later work using SMT-like approaches (Wong and Mooney, 2007) or generating from PCFGs (Belz, 2008) or other formalisms (Soricut and Marcu, 2006; White et al., 2007). In the late 2000s and early 2010s, a number of systems were proposed that did both (Liang et al., 2009; Angeli et al., 2010; Kim and Mooney, 2010; Lu and Ng, 2011; Konstas and Lapata, 2013).

Within the world of neural text generation, some recent work has focused on conditioning language models on tables (Yang et al., 2016), and generating short biographies from Wikipedia Tables (Lebret et al., 2016; Chisholm et al., 2017). Mei et al. (2016) use a neural encoder-decoder approach on standard record-based generation datasets, obtaining impressive results, and motivating the need for more challenging NLG problems.

8 Conclusion and Future Work

This work explores the challenges facing neural data-to-document generation by introducing a new dataset, and proposing various metrics for automatically evaluating content selection, generation, and ordering. We see that recent ideas in copying and reconstruction lead to improvements on this task, but that there is a significant gap even between these neural models and templated systems. We hope to motivate researchers to focus further on generation problems that are relevant both to content selection and surface realization, but may not be reflected clearly in the model's perplexity.

Future work on this task might include approaches that process or attend to the source records in a more sophisticated way, generation models that attempt to incorporate semantic or reference-related constraints, and approaches to conditioning on facts or records that are not as explicit in the box- and line-scores.

Acknowledgments

We gratefully acknowledge the support of a Google Research Award.

References

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 *ICLR*.

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(04):431–455.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *CoNLL*, pages 10–21.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics.

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135. ACM.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. *CoRR*, abs/1702.06235.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Pablo A Duboue and Kathleen R McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *EMNLP*, pages 121–128. Association for Computational Linguistics.

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

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

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9:1735–1780.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *ACL Volume 1: Long Papers*.

Dan Jurafsky and James H Martin. 2014. *Speech and language processing*, volume 3. Pearson London.

Anjuli Kannan and Oriol Vinyals. 2016. Adversarial evaluation of dialogue models. In *NIPS 2016 Workshop on Adversarial Training*.

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.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *CoRR*, abs/1701.02810.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *J. Artif. Intell. Res.(JAIR)*, 48:305–346.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *ACL*, pages 145–150.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*, pages 1203–1213.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *CoRR*, abs/1701.06547.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL*, pages 91–99. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*, pages 2122–2132.

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.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 1412–1421.

Kathleen McKeown. 1992. *Text generation - using discourse strategies and focus constraints to generate natural language text*. Studies in natural language processing. Cambridge University Press.

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 *NAACL HLT*, pages 720–730.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

T. Mikolov, M. Karafit, L. Burget, J. Cernock, and S. Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

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.

Ehud Reiter. 2017. You need to understand your corpora! the weathergov example. https://ehudreiter.com/2017/05/09/weathergov/.

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.

Jacques Robin. 1994. *Revision-based generation of Natural Language Summaries providing historical Background*. Ph.D. thesis, Citeseer.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *ACL*, pages 626–634.

Radu Soricut and Daniel Marcu. 2006. Stochastic language generation using widl-expressions and its application in machine translation and summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1105–1112. Association for Computational Linguistics.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Kumiko Tanaka-Ishii, Kôiti Hasida, and Itsuki Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In *COLING-ACL*, pages 1282–1288.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *AAAI*, pages 3097–3103.

Michael White, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with ccg. In *Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+ MT)*, pages 267–276.

Yuk Wah Wong and Raymond J Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *HLT-NAACL*, pages 172–179.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. *CoRR*, abs/1611.01628.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Zhu Zhang. 2004. Weakly-supervised relation classification for information extraction. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 581–588. ACM.

GuoDong Zhou, JunHui Li, LongHua Qian, and Qiaoming Zhu. 2008. Semi-supervised learning for relation extraction. In *Third International Joint Conference on Natural Language Processing*, page 32.

Appendix

A. Additional Dataset Details

The ROTOWIRE data covers NBA games played between 1/1/2014 and 3/29/2017; some games have multiple summaries. The summaries have been randomly split into training, validation, and test sets consisting of 3398, 727, and 728 summaries, respectively.

The SBNATION data covers NBA games played between 11/3/2006 and 3/26/2017; some games have multiple summaries. The summaries have been randomly split into training, validation, and test sets consisting of 7633, 1635, and 1635 summaries, respectively.

All numbers in the box- and line-scores (but not the summaries) are converted to integers; fractional numbers corresponding to percents are multiplied by 100 to obtain integers in $[0, 100]$. We show the *types* of records in the data in Table 4.

B. Generation Model Details

Encoder For the ROTOWIRE data, a relation r is encoded into $\tilde{\boldsymbol{r}}$ by embedding each of $r.e$, $r.t$, $r.m$ and a "home-or-away" indicator feature in $\mathbb{R}^{600}$, and applying a 1-layer MLP (with ReLU nonlinearity) to map the concatenation of these vectors back into $\mathbb{R}^{600}$. To initialize the decoder LSTMs, we first mean-pool over the $\tilde{\boldsymbol{r}}_j$ by entity (giving one vector per entity), and then linearly transform the concatenation of these pooled entity-representations so that they can initialize the cells and hidden states of a 2-layer LSTM with states also in $\mathbb{R}^{600}$. The SBNATION setup is identical, except all vectors are in $\mathbb{R}^{700}$.

Decoder As mentioned in the body of the paper, we compute two different attention distributions (i.e., using different parameters) at each decoding step. For the Joint Copy model, one attention distribution is not normalized, and is normalized along with all the output-word probabilities.

Within the Conditional Copy model we compute $p(z_t|\hat{y}_{1:t-1}, \boldsymbol{s})$ by mean-pooling the $\tilde{\boldsymbol{r}}_j$, concatenating them with the current (topmost) hidden state of the LSTM, and then feeding this concatenation via a 1-layer ReLU MLP with hidden dimension 600, and with a Sigmoid output layer.

For the reconstruction-loss, we feed blocks (of size at most 100) of the decoder's LSTM hidden states through a (Kim, 2014)-style convolutional model. We use kernels of width 3 and 5, 200 filters, a ReLU nonlinearity, and max-over-time pooling. To create the p_k, these now 400-dimensional features are then mapped via an MLP with a ReLU nonlinearity into 3 separate 200 dimensional vectors corresponding to the predicted relation's entity, value, and type, respectively. These 200 dimensional vectors are then fed through (separate) linear decoders and softmax layers in order to obtain distributions over entities, values, and types. We use $K = 3$ distinct p_k.

Models are trained with SGD, a learning rate of 1 (which is divided by 2 every time validation perplexity fails to decrease), and a batch size of 16. We use dropout (at a rate of 0.5) between LSTM layers and before the linear decoder.

C. Information Extraction Details

Data To form an information extraction dataset, we first sentence-tokenize the gold summary documents $y_{1:T}$ using NLTK (Bird, 2006). We then determine which word-spans $y_{i:j}$ could represent entities (by matching against players, teams, or cities in the database), and which word-spans $y_{k:l}$ could represent numbers (using the open source `text2num` library[5] to convert (strings of) number-words into numbers).[6] We then consider each $y_{i:j}, y_{k:l}$ pair in the same sentence, and if there is a record r in the database such that $r.e = y_{i:j}$ and $r.m = \text{text2num}(y_{k:l})$ we annotate the $y_{i:j}, y_{k:l}$ pair with the label $r.t$; otherwise, we give it a label of ϵ.

Model We predict relations by ensembling 3 convolutional models and 3 bidirectional LSTM (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005) models. Each model consumes the words in the sentence, which are embedded in $\mathbb{R}^{200}$, as well as the distances of each word in the sentence from both the entity-word-span and the number-word-spans (as described above), which are each embedded in $\mathbb{R}^{100}$. These vectors are concatenated (into a vector in $\mathbb{R}^{500}$) and fed into either a convolutional model or a bidirectional LSTM model.

The convolutional model uses 600 total filters, with 200 filters for kernels of width 2, 3, and 5, respectively, a ReLU nonlinearity, and max-pooling. These features are then mapped via a 1-

[5] https://github.com/exogen/text2num

[6] We ignore certain particularly misleading number-words, such as "three-point," where we should not expect a corresponding value of 3 among the records.

Player Types	POSN	MIN	PTS	FGM	FGA	FG-PCT	FG3M	FG3A	FG3-PCT
	FTM	FTA	FT-PCT	OREB	DREB	REB	AST	TOV	STL
	BLK	PF	FULL-NAME	NAME1	NAME2	CITY			
Team Types	PTS-QTR1	PTS-QTR2	PTS-QTR3	PTS-QTR4	PTS	FG-PCT	FG3-PCT	FT-PCT	REB
	AST	TOV	WINS	LOSSES	CITY	NAME			

Table 4: Possible Record Types

layer (ReLU) MLP into $\mathbb{R}^{500}$, which predicts one of the 39 relation types (or ϵ) with a linear decoder layer and softmax.

The bidirectional LSTM model uses a single layer with 500 units in each direction, which are concatenated. The hidden states are max-pooled, and then mapped via a 1-layer (ReLU) MLP into $\mathbb{R}^{700}$, which predicts one of the 39 relation types (or ϵ) with a linear decoder layer and softmax.