Pragmatically Informative Text Generation

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein
Why Might Generation Need Pragmatics?

**Input, i**
Name [Fitzbillies],
Type [Coffee Shop],
Food [English], Price [Cheap]

**Output, o**
Fitzbillies is a cheap coffee shop.

**Predicted Input, i**
Name [Fitzbillies],
Type [Coffee Shop],
Price [Cheap]
Why Might Generation Need Pragmatics?

Input, $i$
Name [Fitzbillies],
Type [Coffee Shop],
Food [English], Price [Cheap]

Output, $o$
Fitzbillies is a cheap English coffee shop.

Predicted Input, $i$
Name [Fitzbillies],
Type [Coffee Shop],
Food [English], Price [Cheap]
Generation as a Pragmatic Game

Input, $i$
Name [Fitzbillies],
Type [Coffee Shop],
Food [English], Price [Cheap]

Output, $o$
Fitzbillies is a cheap coffee shop.

Predicted Input, $i$
Name [Fitzbillies],
Type [Coffee Shop],
Price [Cheap]

Predicted Input, $i$
Name [Fitzbillies],
Type [Coffee Shop],
Food [English], Price [Cheap]

Listener ✔ ✗ ✗ ✗

Speaker

Predicted Input, $i$
Name [Fitzbillies],
Type [Coffee Shop],
Price [Cheap] ✔

Predicted Input, $i$
Name [Fitzbillies],
Type [Coffee Shop],
Price [Cheap] ✗
Generation as a Pragmatic Game

**Speaker**

Input, i
- Name [Fitzbillies],
- Type [Coffee Shop],
- Food [English], Price [Cheap]

Output, o
- Fitzbillies is a cheap English coffee shop.

**Listener**

Predicted Input, i
- X Name [Fitzbillies], Type [Coffee Shop], Price [Cheap]
- X Name [Fitzbillies], Price [Cheap]
- X Name [Fitzbillies], Type [Coffee Shop], Food [English], Price [Cheap]
- ✔ Name [Fitzbillies], Type [Coffee Shop], Price [Cheap]
- X Name [Fitzbillies], Type [Coffee Shop], Price [Cheap]
Generating Pragmatic Output Text

Input, $i^*$
- Name [Fitzbillies],
- Type [Coffee Shop],
- Food [English], Price [Cheap]
Fitzbillies is a cheap English coffee shop.
Fitzbillies is a cheap coffee shop.

Generating Pragmatic Output Text

Input, i*
Name [Fitzbillies],
Type [Coffee Shop],
Food [English], Price [Cheap]

Output, o
Fitzbillies is a cheap English coffee shop.

Searching:
Search over possible outputs o, using candidates from a standard seq-to-seq speaker model.
Fitzbillies is a cheap coffee shop.

Input, $i^*$
- Name [Fitzbillies]
- Type [Coffee Shop]
- Food [English]
- Price [Cheap]

Output, $o$
- Fitzbillies is a cheap English coffee shop.

Searching:
Search over possible outputs $o$, using candidates from a standard seq-to-seq speaker model

Predicted Input, $i$
- Name [Fitzbillies]
- Type [Coffee Shop]
- Food [English]
- Price [Cheap]
Generating Pragmatic Output Text

**Input, \( i^* \)**
- Name: Fitzbillies
- Type: Coffee Shop
- Food: English
- Price: Cheap

**Output, \( o \)**
Fitzbillies is a cheap English coffee shop.

**Searching:**
Search over possible outputs \( o \), using candidates from a standard seq-to-seq speaker model.

**Predicted Input, \( i \)**
- Name: Fitzbillies
- Type: Coffee Shop
- Food: English
- Price: Cheap

**Listener**

**P(\( i^* | o \))**

**Scoring:**
Choose an output with maximum listener probability, \( P(\( i^* | o \)) \).
How to Construct the Listener?

**Reconstructor-Based**
Train a separate Listener model to give a distribution over any possible inputs.

**Distractor-Based**
Construct a context-appropriate distractor input that Listener needs to distinguish the true input from.

$P(i | o)$
Past Work on Pragmatic Generation

**Convey All Relevant Info**

**Be Informative in Context**

*Motivates Reconstructor*  
*Motivates Distractor*
Reconstructor-Based Pragmatics

Listener

$P(i | o)$

$O_1$

$i_1$

All Possible Inputs
Reconstructor-Based Pragmatics

**Searching:**
Obtain candidate outputs by beam search in $P(o \mid i^*)$

Speaker
$P(o \mid i^*)$

Listener
$P(i \mid o)$

True input
$i^*$

- $O_1$
- $O_2$
- $O_3$

- $i_1$
- $i_1$
- $i_1$

All Possible Inputs
All Possible Inputs
All Possible Inputs
Reconstructor-Based Pragmatics

**Searching:**
Obtain candidate outputs by beam search in $P(o | i^*)$

**Scoring:**
Score and select $o$ using $P(i^* | o)$
Distractor-Based Pragmatics

When we use a Listener can only produce the true input and a distractor, we can define the Listener using the Speaker and Bayes’ rule:

**Searching:**
Obtain candidate outputs by beam search in $P(o \mid i^*)$

**Scoring:**
Choose output by $\text{argmax}_o P(i^* \mid o)$
Distractor-Based Pragmatics

\[ P(i^* | o) = \frac{P(o | i^*)}{\sum_{i' \epsilon \{i^*, \bar{i}\}} P(o | i')} \]

Possible Outputs
(search over these)

- \( O_1 \): Fitzbillies is a cheap coffee shop.
- \( O_2 \): Fitzbillies is a cheap English coffee shop.

Inputs

- **True Input, \( i^* \)**
  - Name [Fitzbillies],
  - Eat Type [Coffee Shop],
  - Food [English],
  - Price [Cheap]

- **Distractor, \( \bar{i} \)**
  - Name [Fitzbillies],
  - Eat Type [Coffee Shop],
  - Price [Cheap]

| Possible Outputs | \( P(o | i^*) \) |
|------------------|-----------------|
| \( O_1 \) | 0.4 |
| \( O_2 \) | 0.2 |
| ... | ... |
| \( O_1 \) | 0.8 |
| \( O_2 \) | 0.05 |
| ... | ... |
Distractor-Based Pragmatics

\[ P(i^*|o) = \frac{P(o|i^*)}{\sum_{i' \in \{i^*,i\}} P(o|i')} \]

Possible Outputs (search over these)

- \( O_1 \): Fitzbillies is a cheap coffee shop.
- \( O_2 \): Fitzbillies is a cheap \textbf{English} coffee shop.

Inputs

- True Input, \( i^* \)
  - Name [Fitzbillies], Eat Type [Coffee Shop], Food[English], Price[Cheap]
  - Probability: 0.33

- Distractor, \( \tilde{i} \)
  - Name [Fitzbillies], Eat Type [Coffee Shop], Price[Cheap]
  - Probability: 0.66

...
In practice: do the search and normalization incrementally, word-by-word. [Cohn-Gordon et al. 2018.]

$$P(i^*|o) = \frac{P(o|i^*)}{\sum_{i' \in \{i^*, \tilde{i}\}} P(o|i')}$$
Input:
Name[Fitzbillies],
EatType[Coffee Shop],
PriceRange[Cheap],
Area[Riverside],
Food[English]

Output:
Fitzbillies is a coffee shop that serves English food. It is located in riverside area.

[Puzikov and Gurevych, 2018]
Reconstructor:

$S^R$ (a multi-task classifier) maps each output to input.

- **Fitzbillies** is a coffee shop that serves English food. It is located in riverside area.

### Input:
- Name: Fitzbillies
- EatType: Coffee Shop
- PriceRange: Cheap
- Area: Riverside
- Food: English
Generation from Meaning Representations

- **BLEU**

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>State-of-the-Art Base Speaker</td>
<td>66.5</td>
</tr>
<tr>
<td>Pragmatic with Reconstructor</td>
<td>68.6</td>
</tr>
</tbody>
</table>

- [Puzikov and Gurevych, 2018]
Distractor: $S^D$ is based on the MR that masks out other attributes.

Eg: Near[Burger King]

Listener $P(i^* \mid o)$

Input:
Name[Fitzbillies],
EatType[Coffee Shop],
PriceRange[Cheap],
Area[Riverside],
Food[English]
Generation from Meaning Representations

State-of-the-Art Base Speaker

Pragmatic with Reconstructor

Pragmatic with Distractor

[Puzikov and Gurevych, 2018]

BLEU

66.5

68.6

67.8
Abstractive Summarization

Long Document:
It is the primary reason all four English teams - Liverpool, Chelsea, Arsenal and Manchester City - were eliminated from the Champions League before the quarter-final draw.

Extracted Sentences:

1. Manchester City became the latest team to be eliminated from Europe;
2. City were dumped out of the Champions League last 16 by Barcelona;
3. ...

[Chen and Bansal, 2018]
Abstractive Summarization

ROUGE-2

19.5
State-of-the-Art
[Celikyilmaz+ ‘18]

17.8
Base Speaker
[Chen & Bansal ‘18]

METEOR

21.0
State-of-the-Art
[Paulus+, ‘18]

20.4
Base Speaker
[Chen & Bansal, ‘18]
Abstractive Summarization

Long Document:

Extracted Sentences:

\[ I_{e1} \]
\[ I_{e2} \]
\[ I_{e3} \]

Seq-to-Seq Listener

Abstractive Output:

\[ O_{a1} \]
\[ O_{a2} \]
\[ O_{a3} \]

Reconstructor:

\( S^R \) (seq-to-seq model) maps abstractive outputs to extractive inputs.
Abstractive Summarization

ROUGE-2

- State-of-the-Art: 19.5
  - [Celikyilmaz+ ‘18]
- Base Speaker: 17.8
  - [Chen & Bansal ‘18]
- Pragmatic Rec.: 18.1

METEOR

- State-of-the-Art: 21.0
  - [Paulus+ ‘18]
- Base Speaker: 20.4
  - [Chen & Bansal ‘18]
- Pragmatic Rec.: 20.6
Abstractive Summarization

Long Document:

Extracted Sentences:

\[ I_{e1}, I_{e2}, I_{e3} \]

Extractor

Abstractive Output:

\[ O_{a1}, O_{a2}, O_{a3} \]

Listener’s Distractor

Listener’s Distractor

Distractor:

For a given extracted sentence, use the next extracted sentence as the distractor.
Conclusions

- Modeling generation as a speaker-listener game leads to more adequate and informative outputs
- Computational pragmatics produces improvements for general text generation tasks
Thanks!

Our code is publicly available at

https://github.com/sIncerass/prag_generation