Teaching Machines to Read and Comprehend

1 Citation

Hermann, Karl Moritz, et al. "Teaching machines to read and comprehend." Advances in Neural Information Processing Systems. 2015.

`http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.pdf`

2 Abstract

We want AI that can read a document and answer a Cloze question (e.g. "X's stock price drop tied to new defense spending bill" - find X given the document). We present a dataset for this and create a number of baseline models.

3 Introduction

There are few reading comprehension datasets out there, so most people don't really solve reading comprehension with supervised learning. Some people have generated synthetic data by simulating people, objects, and places interacting together and generating sentences based on these interactions. Models that do well on this synthetic data don't really scale to other data. We find that combining CNN and Daily Mail news articles with their summaries and using named entity recognition, we can generate a huge dataset of Cloze questions. We make a bunch of baseline models.

4 Supervised training data for reading comprehension

Reading comprehension models aim to compute $p(a|d, q)$ where d is a context document, q is a question, and a is an answer. We a create a dataset for this problem. We take 93K articles from CNN and 220K articles from Daily Mail. Each article comes with a bullet point summary. We do named entity recognition on the summaries and select one entity in a summary as "X" - this is the Cloze question. We then replace every entity with a random ID (e.g. Republican may be replaced with "ent12", "Democrat" with "ent93"). We do this because we want to force the models to need the document to answer the question. For example, a model may be able to answer: "Most Americans do not identify as Democrats or X" without reading the document, but it certainly will not be able to answer "Most ent243 do not identify as ent93 or X" without reading the document.

5 Models

We have two simple baselines. The maximum frequency baseline just picks the most frequent entity in the document. The exclusive frequency baseline just picks the most frequent entity in the document that is NOT in the query.

We have some traditional NLP baselines. Our frame-semantic parsing model runs frame-semantic parsing (i.e. produces (subject, verb, object) triples from the document and query) and tries to match the query triple to the most likely document triple. Checking for a match uses a bunch of simple rules, starting with exact match and progressively loosening (see table in paper). The second model is a word distance model. We align the Cloze X with each entity in the document and see which alignment minimizes the sum of distances (each distance is clipped to $m = 8$ if it is $> m$) between each query word and the corresponding word in the document.

We also create some neural network models to solve $p(a|d, q) \propto \exp(W(a)g(d, q))$ such that $a \in V$ (where $W(a)$ is the row of the weight matrix W corresponding to answer a and $g(d, q)$ is a document-query embedding). We consider three neural network models.

First, we use a Deep LSTM. We feed the document (one word at a time), a delimeter, and then the query (we also tried switching the order). The input is $x(t)$ and output is $y(t)$. We have skip connections from input to hidden layers and hidden layers to output. For time t and hidden layer k with input i, forget f, and output o gates and the query/document delimiter $||$, we get:

$$x'(t,k) = x(t)||y'(t,k-1) \tag{1}$$
$$i(t,k) = \sigma(W_{kxi}x'(t,k)W_{khi}h(t-1,k) + W_{kci}c(t-1,k) + b_{ki}) \tag{2}$$
$$f(t,k) = \sigma(W_{kxf}x(t)W_{khf}h(t-1,k) + W_{kcf}c(t-1,k) + b_{kf}) \tag{3}$$
$$c(t,k) = f(t,k)c(t-1,k) + i(t,k)\tanh(W_{kxc}x'(t,k) + W_{khc}h(t-1,k) + b_{kc}) \tag{4}$$
$$o(t,k) = \sigma(W_{kco}x'(t,k) + W_{kho}h(t-1,k) + W_{kco}c(t,k) + b_{ko}) \tag{5}$$
$$h(t,k) = o(t,k)\tanh(c(t,k)) \tag{6}$$
$$y'(t,k) = W_{ky}h(t,k) + b_{ky} \tag{7}$$
$$y(t) = y'(t,1)||...||y'(t,K) \tag{8}$$
$$g^{LSTM}(d,q) = y(|d| + |q|) \text{ (joint embedding)} \tag{9}$$

The fixed-size hidden vector in the Deep LSTM is limiting, so we also create another model we call the Attentive Reader. It has a bidirectional LSTM and attention mechanism. It has the following equations (the arrow accents indicate output from the forward or backward LSTM):

$$u = \overrightarrow{y}_q(|q|) + \overleftarrow{y}_q(1) \text{ (query embedding)} \tag{10}$$
$$y_d(t) = \overrightarrow{y}_d(t)||\overleftarrow{y}_d(t) \tag{11}$$
$$m(t) = \tanh(W_{ym}y_d(t) + W_{um}u) \tag{12}$$
$$s(t) \propto \exp(w_{ms}^T m(t)) \tag{13}$$
$$r = y_d s \text{ (document embedding)} \tag{14}$$
$$g^{AR}(d,q) = \tanh(W_{rg}r + W_{ug}u) \text{ (joint embedding)} \tag{15}$$

We can also create a model called the Inattentive Reader that accumulates information while it reads the query and attends to different pieces of the document:

$$y_q(i) = \overrightarrow{y}_q(i)||\overleftarrow{y}_q(i) \tag{16}$$
$$m(i,t) = \tanh(W_{dm}y_d(t) + W_{rm}r(i-1) + W_{qm}y_q(i)) \text{ for } 1 \le i \le |q| \tag{17}$$
$$s(i,t) \propto \exp(w_{ms}^T m(i,t)) \tag{18}$$
$$r(0) = \mathbf{r}_0 \tag{19}$$
$$r(i) = y_d^T s(i) + \tanh(W_{rr}r(i-1)) \text{ for } 1 \le i \le |q| \tag{20}$$
$$g^{IR}(d,q) = \tanh(W_{rg}r(|q|) + W_{qg}u) \tag{21}$$

6 Empirical Evaluation

The Attentive Reader and Impatient Readers are the best (63.8% accuracy for CNN and 68.0% for Daily Mail).

The Frame Semantic Parser does quite poorly because many sentences in the dataset do not follow the "subject-verb-object" form and because documents have have a lot of sentences.

The Word Distance model does surprisingly well, but we don't think it will generalize beyond Cloze questions.

The Deep LSTM does well, but the Attentive Reader and Impatient Reader do better. Looking at where they attend, they do a good job focusing on the relevant part of the document.

The maximum frequency and exclusive frequency models do very poorly.

7 Conclusion

Using named entity recognition and a dataset of news articles and summary, we create a dataset of Cloze questions. We present the Attentive Reader and Impatient Reader models, which do pretty well (between 60-70% accuracy) at this task.