Paper 2: End-to-end neural coreference

tl;dr: Language rules suuuuuuuck

link: https://arxiv.org/pdf/1707.07045.pdf
implementation example: https://www.kaggle.com/keyit92/end2end-coref-resolution-by-attention-rnn
implementation from paper: https://github.com/kentonl/e2e-coref

What's the idea?
Consider all spans (span is basically an n-gram) in a document. Then, for each span calculate probability of it referring to previously considered spans (mention score). Finally, calculate probability of a spans being connected to each other (antecedent score) and sum up scores accordingly to get the probability. Mention may as well not be connected to anything that came before it, so we need to calculate that score as well.

Any example?
The paper has an example that has a nice graph, but I'll write it down here. Let's say we have a sentence:

„General Electric said the Postal Service contacted the company"

Mention score: If there was a „they" before the sentence, which of spans could refer to it? „General Electric"? Sure. „Electric said the"? Not so much. In other words: does a span make sense to be referred to?
Antecendent score: So we have highest-ranked mentions: „General Electric", „the Postal Service", „the company". Which of them are most likely to refer to each other? First two make much sense, but „the company" may refer to either - we may need to decide.
Coreference score: Sum of mention scores and antecedent score of a pair of mentions. We also consider coreference score of a mention and so-called dummy antecedent, which means no such mention exists or it doesn't exist before that mention.
Final decision: We put all coreference scores into a softmax function to get a probability distribution.

How do we measure it?
We use feed-forward neural networks to calculate scores, using vectors of span representation. In case of mention, we only need representation of a single span. In case of antecedent, we use representations of both vectors, their common parts (element-wise multiplication) and so-called feature vector containing data on distance between spans (considered as buckets rather than just values, e.g. 32-63 instead of 52) and speaker and genre information from metadata (e.g. their gender).

But how are spans represented?
First, we need to somehow represent words. We use pretrained vector representations for that (in paper, it's a concatenation of two types of embeddings - 350 dimensions in total). We also use one-dimensional convolution neural networks over 8 characters with convolutions of size 3, 4 and 5 with 50 filters each. We then use following formulas (f for found words, o for out-of-vocabulary words and c for characters, I guess) on an LSTM:

$$
\begin{aligned}
\boldsymbol{f}_{t,\delta} &= \sigma(\mathbf{W}_{\mathrm{f}}[\boldsymbol{x}_t, \boldsymbol{h}_{t+\delta,\delta}] + \boldsymbol{b}_{\mathrm{i}}) \\
\boldsymbol{o}_{t,\delta} &= \sigma(\mathbf{W}_{\mathrm{o}}[\boldsymbol{x}_t, \boldsymbol{h}_{t+\delta,\delta}] + \boldsymbol{b}_{\mathrm{o}}) \\
\widetilde{\boldsymbol{c}}_{t,\delta} &= \tanh(\mathbf{W}_{\mathrm{c}}[\boldsymbol{x}_t, \boldsymbol{h}_{t+\delta,\delta}] + \boldsymbol{b}_{\mathrm{c}}) \\
\boldsymbol{c}_{t,\delta} &= \boldsymbol{f}_{t,\delta} \circ \widetilde{\boldsymbol{c}}_{t,\delta} + (\mathbf{1} - \boldsymbol{f}_{t,\delta}) \circ \boldsymbol{c}_{t+\delta,\delta} \\
\boldsymbol{h}_{t,\delta} &= \boldsymbol{o}_{t,\delta} \circ \tanh(\boldsymbol{c}_{t,\delta}) \\
\boldsymbol{x}_t^* &= [\boldsymbol{h}_{t,1}, \boldsymbol{h}_{t,-1}]
\end{aligned}
$$

We can also try to find so-called syntactic heads, or words that categorise the entire span. We use an attention mechanism to find them using these formulas:

$$\alpha_t = \boldsymbol{w}_\alpha \cdot \text{FFNN}_\alpha(\boldsymbol{x}_t^*)$$
$$a_{i,t} = \frac{\exp(\alpha_t)}{\sum_{k=\text{START}(i)}^{\text{END}(i)} \exp(\alpha_k)}$$
$$\hat{\boldsymbol{x}}_i = \sum_{t=\text{START}(i)}^{\text{END}(i)} a_{i,t} \cdot \boldsymbol{x}_t$$

a-weights are taught really well, so we don't have to rely on language rules to get good results. Finally, we get the span representation:

$$\boldsymbol{g}_i = [\boldsymbol{x}^*_{\text{START}(i)}, \boldsymbol{x}^*_{\text{END}(i)}, \hat{\boldsymbol{x}}_i, \phi(i)]$$

Now, what is that phi? That's feature representation. That representation contains binary data on whether speaker is the same, the genre is the same and distance and mention width, each of them in one of nine buckets: [1,2,3,4,5-7, 8-15, 16-31, 32-63, 64+]. In total, that representation has 20 dimensions.

Does it work?

The paper claims it does - significantly better than other methods. But it has some problems with similarly placed mentions (e.g „(The flight attendants) have until 6:00 today to ratify labor concessions. (The pilots') union and ground crew did so yesterday. - here model believes attendants and pilots are the same thing) and real-life knowledge (e.g. „Also such location devices, (some ships) have smoke floats (they) can toss out so the man overboard will be able to use smoke signals as a way of trying to, let the rescuer locate (them)." - here model thinks „they" refers to „some ships" rather than to „the man overboard").

Room for improvement?

Mostly in feature representation: we can replace genre with gender (as that matters in our case), we can change buckets to more natural ones (research required into how distributed these things are).