State-of-the-art neural coreference resolution for chatbots

Thomas Wolf Follow
Jul 7, 2017 · 9 min read

> ***TL;DR, Links:*** *Online demo at https://huggingface.co/coref, Github repo for Neuralcoref: https://github.com/huggingface/neuralcoref*

At Hugging Face 🤗 we work on the most amazing and challenging subset of natural language: millennial language. Full of uncertainties 🤔, implicit references 👾, emojis 😭, jokes 😂 and constantly creating novel expressions…

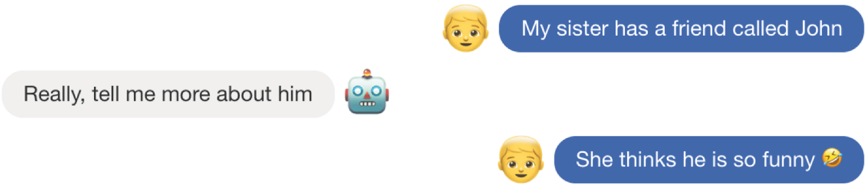
To navigate these stormy waters 🛳, we have developed a number of specific NLP tools based on the latest research in the field. One of these tools is a *coreference system* we use to keep track of short term references already at the front end of our AI 🤗 brains.

We couldn't find a tool we could easily integrate in our conversational agent so we decided to develop it internally and open-source it.

You can also try the coreference system for yourself on the demo we setup!

> *But, what is coreference?*

Let's have a look at **Bob** 👦 who is talking with his AI friend **Alice** 🤖



My sister has a friend called John

Really, tell me more about him

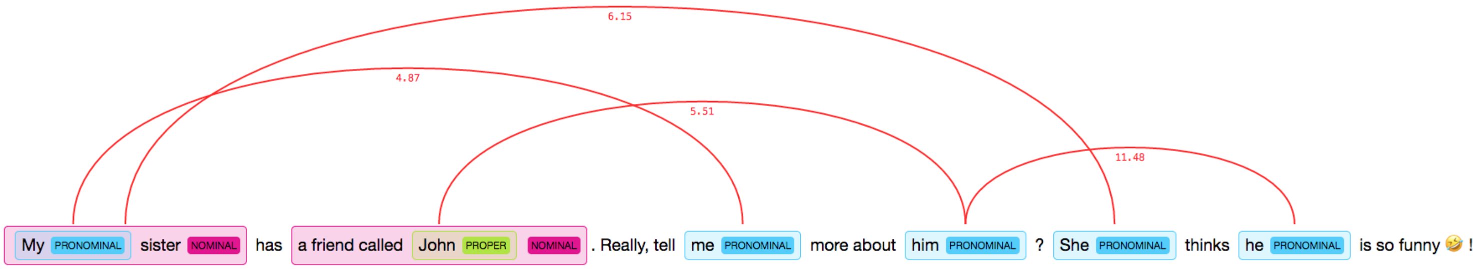
She thinks he is so funny 🤣

There are several implicit references in the last message from Bob 👦

- "**she**" refers to the same entity as "**My sister**": *Bob's sister*.

- “**he**” refers to the same entity as “**a friend called John**”: *Bob’s sister’s friend.*

The process of linking together mentions that relates to real world entities is called *coreference resolution* [1].



Hugging face coreference system in operation. Try it for yourself!

Humans naturally associate these references together—but for an AI 🤖 brain, it is much more difficult! And when we say it is hard, we mean it,

> *Coreference resolution is the basis of the Winograd Schema Challenge, a test of machine intelligence … build to defeat the AIs who’ve beaten the Turing Test!* 🔥

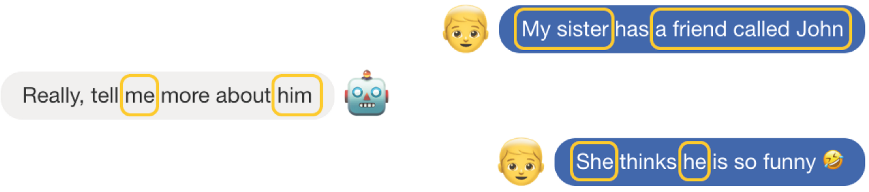
> *So how do we solve this problem without creating a strong-AI?*

. . .

Coreference is a rather old NLP research topic [1]. It* has seen a revival of interest in the past two years as several research groups [2] applied cutting-edge deep-learning and reinforcement-learning techniques to it. It was published earlier this year that coreference resolution may be instrumental in improving the performances of NLP neural architectures like RNN and LSTM (see “Linguistic Knowledge as Memory for Recurrent Neural Networks” by B. Dhingra, Z. Yang, W. W. Cohen, and R. Salakhutdinov).

A typical coreference resolution algorithm goes like this:

- We extract a series of *mentions* –words that are potentially referring to real world entities–



My sister has a friend called John

Really, tell me more about him

She thinks he is so funny 🤣

In our previous dialogue, we can identify six mentions

- For each mentions and each pair of mentions, we compute a set of *features* (more on that soon).
- Then, we find the most likely antecedent for each mention (if there is one) based on this set of features. This last step is called *pairwise ranking* [3].

Traditionally the set of *features* was hand-engineered from linguistic features and it could be huge. Some high quality systems use 120+ features [4]!

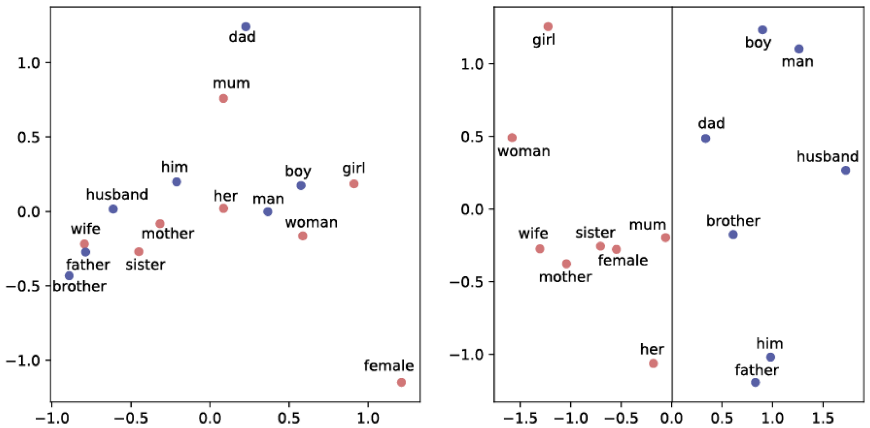
Here comes the nice thing about modern NLP techniques like word vectors and neural nets. They allow us to automatically learn a lot of these hand-engineered features and reduce the set of hand-designed features by an almost an order of magnitude while keeping a good or even better accuracy.

Let's see that in action on a simple example.

Her is a feminine pronoun and should have a higher probability to refer to *my sister* 👧 than to *my brother* 👦.

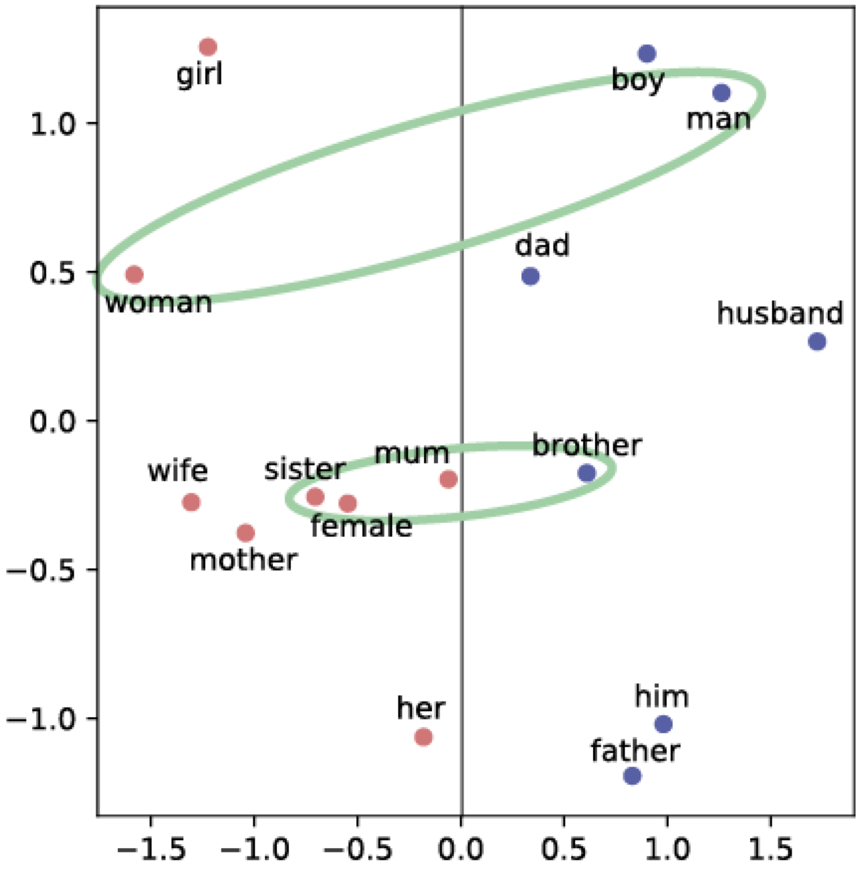
We could encode that by hand, listing the gender and other properties of every word in our vocabulary—but we can also assign a *vector* to every word the vocabulary and let our model learn vectors that are adapted for our task on a training dataset.

So we trained our word embeddings on a large coreference annotated dataset without supplying any information regarding word gender. And here is what we got.



Left: **Initial** word embeddings (PCA of pre-trained word2vec) — Right: **trained** word embeddings (PCA)

On the left, the original word2vec words vectors don't specifically care about gender association [5]. On the right side, after training, our word vectors shows feminine and masculine nouns nicely separated along the principal components of the vectors even though we didn't supply any information regarding word gender (*gender* has become a direction of main variance of our trained word vectors).



Obviously the quality of our trained embeddings depends a lot on the training dataset. Our embeddings are trained on the OntoNotes 5.0 dataset (the largest coreference annotated corpus).

But clearly this dataset has its flaws. In particular, as often in NLP datasets, it is build mainly from news and web articles hence with a more formal language than the usual chatbot user.

We can see on our PCA projection that pairs of words like *dad/mum or brother/sister* seem less clearly separated than a pair of more formal words like *woman/man* (along the first components of the PCA).

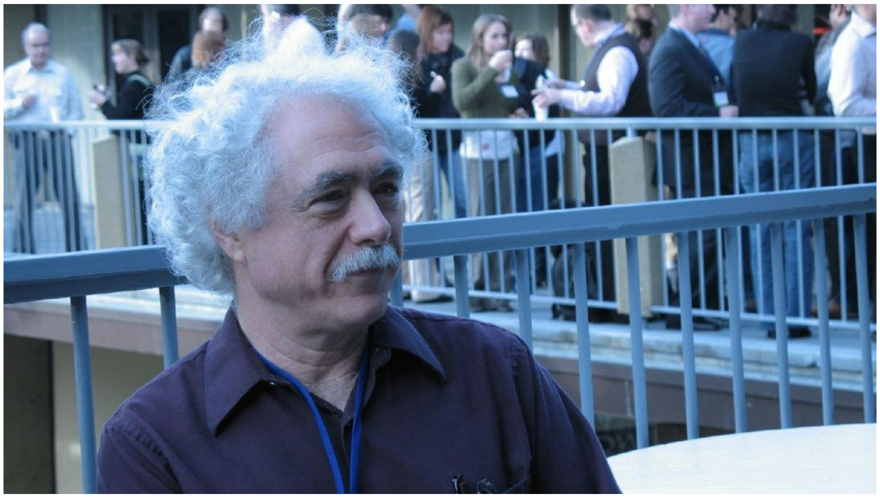
And, in fact, you can construct sentences for which our coreference system will work nicely on some pairs of mention but fail on another pair of mentions which are less formal. We give some hacks to circumvent that in a minute but the cleanest way is of course to gather a labeled data set that is more representative of your production data (like the over 10M messages already exchanged by our users with their Huggingface AIs).

. . .

So is that all? If we can learn every linguistic features relevant to our words, then how can coreference resolution be more difficult than the Turing Test?

Well, in the Winograd Schema Challenge, your AI has to solve questions like:

> *The trophy would not fit in the brown suitcase because* ***it*** *was too big. What was too big? the trophy or the suitcase?*

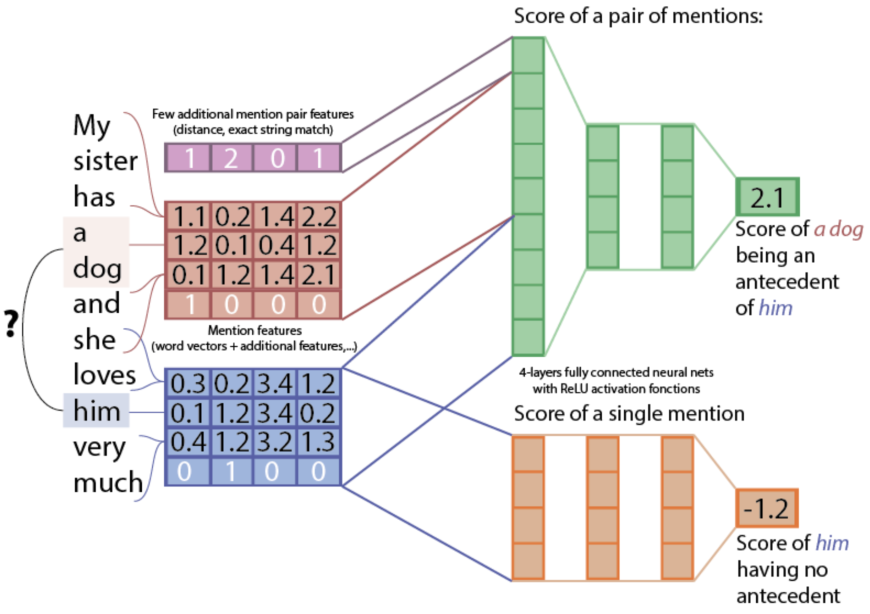


Terry Winograd (Flickr/Lisa Padilla)

Our carefully gender-tuned-word embeddings will not be enough to help us solve these questions because we actually have to *take the context of our mentions into account* and maybe even *external common knowledge*!

In our model, we add some simple context information by averaging word vectors surrounding each mention but there are many way you can add some context information [2]. The good thing is, because we're building an entertaining AI, we don't need 100% accuracy for it to work for users. And a high accuracy can be very hard to reach: the best team competing for the Winograd Schema Challenge last year reached only 58% of success!

Our model then goes very roughly as follow: we take word embeddings for several words inside and around each mention, average them if needed, add some simple integer features (length of the mention, speaker information, location of the mentions…) to obtain a features representation for each mention and it surroundings. Then we plug these representations into two neural nets. A first neural net gives us a score for each pair of a mention and a possible antecedent while a second neural net gives us a score for a mention having no antecedent (sometimes a mention is the first reference to an entity in a text). We can then simply compare all these scores together and take the highest score to determine whether a mention has an antecedent and which one it should be.


Score of a pair of mentions:
My
sister
has
a
dog
and
she
loves
him
very
much
?
Few additional mention pair features
(distance, exact string match)
1 2 0 1
1.1 0.2 1.4 2.2
1.2 0.1 0.4 1.2
0.1 1.2 1.4 2.1
1 0 0 0
Mention features
(word vectors + additional features,...)
0.3 0.2 3.4 1.2
0.1 1.2 3.4 0.2
0.4 1.2 3.2 1.3
0 1 0 0
2.1
Score of a dog being an antecedent of him
4-layers fully connected neural nets
with ReLU activation fonctions
Score of a single mention
-1.2
Score of him having no antecedent

Rough sketch of the coreference scoring model.

The neural model is trained on a non-probabilistic slack-rescaled max-margin objective. It means that the system computes a score for each pair of mentions and a score for each individual mention but these scores are not probabilities, just scores in arbitrary unit (the highest the better).

This scoring system is an adaptation of the very nice work published last year by Kevin Clark and Christopher Manning (see “Deep Reinforcement Learning for Mention-Ranking Coreference Models” by Kevin Clark and Christopher D. Manning, EMNLP 2016, Improving Coreference Resolution by Learning Entity-Level Distributed Representations by Kevin Clark and Christopher D. Manning, ACL 2016, and the references therein). The full details and even more are given in theses publications which you should definitely read if you are interested in this model.

The scoring model of Clark and Manning is open-source and a full implementation (with mention detection and features computation) has been integrated in Stanford's CoreNLP.

We initially used this implementation in our system. However, we found that several important features were missing for our needs:

1. Easy integration in our current NLP processing pipeline. CoreNLP is an extensive tool but it is also a large monolithic java bloc that is hard to integrate in a high throughput distributed system like ours.
2. Capability to evolve with our users language and take into account user-specific informations. New word vectors have to be dynamically learned to use the fact that "Kendall Jenner" is a woman and a model even though she is not mentioned in our training dataset.
3. Taking care of the speakers in a conversation. The quality of the coreference resolution depend for a large part of the speaker information associated to each mention.

Here is how we solved that:

1. Our current pipeline is based on a set of deep-learning python tools and the high speed parsing is done by spaCy. We are big fans of spaCy ultra-fast parser and of the work of Matthew and Ines at Explosion.ai. The coreference system with mentions detection, features extraction and neural net computation is thus implemented on top of spaCy and Numpy (in the future we could easily switch to Thinc when Thinc's API is stabilized).
2. We make use of recent work on word embeddings to compute embeddings for unknown words on the fly from definitions or information that you can provide (it's very simple in fact: you can compute a word embedding for "Kendall Jenner" simply by averaging the vectors for "woman" and "model" for example).
3. We input and take care of the various speakers in the conversation when computing the features and resolving the coreferences.

You can try the coreference system for yourself!

You can also fork the code and use it in your projects. Hope you liked it and let us know how you use it 🚀

. . .

*. ^ Coreference pun!

1. ^ Good introductions of the subject can be found in the great NLP class of Chris Manning and in "*Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition*" by Daniel Jurafsky & James H. Martin, 2nd edition 2007 (Chapter 21 in particular). Linguistic is a fascinatingly huge domain so you may also have heard of anaphors and cataphors. Coreferences, anaphors and cataphors describes different relationships that sometimes overlap but not always. In particular *coreferences* are mentions that all relates to *the same object of the outside world*.

2. ^ See in particular: "Learning Global Features for Coreference Resolution" by Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber, NAACL 2016, "Deep Reinforcement Learning for Mention-Ranking Coreference Models" by Kevin Clark and Christopher D. Manning, EMNLP 2016, and "Latent Structures for Coreference Resolution" by Sebastian Martschat and Michael Strube, ACL 2015.

3. ^ There are other ways you can link entities (entity-mention models, , for e.g. by constructing cluster of mentions and considering global features (coreference is a clustering operation) but we won't talk about them here. You should read the references above of the great NLP class of Chris Manning, or refer to "*Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition*" by Daniel Jurafsky & James H. Martin, for example, if you want to know more about that.

4. ^ See Modeling the lifespan of discourse entities with application to coreference resolution by Marneffe et al. (2015) and Search space pruning: A simple solution for better coreference resolvers by Nafise Sadat Moosavi and Michael Strube (2016).

5. ^ well they do somehow but they also have to encode many other semantic/syntaxique features of the words—many of these being more important than gender— to predict the surrounding words (the objective in word2vec training)