Can Deep Learning Techniques Improve Entity Linking?

Julien Plu
julien.plu@eurecom.fr
@julienplu
Me, Myself and I

- Master in artificial intelligence from UM2 in 2012
- Research engineer at Orange for 2 years
- PhD Student at EURECOM since July 2014
- Lead the Semantic Web section at Developpez.com
- Co-author of the book: Web de données Méthodes et outils pour les données liées
- Areas of expertise: Semantic Web, Natural Language Processing and Machine Learning
En 1968, lorsque les Yardbirds se séparent, Jimmy Page est encore sous contrat et doit honorer des dates de concerts. Alors il recherche des musiciens pour former un nouveau groupe avec le manager des Yardbirds, Peter Grant, John Paul Jones apprend la nouvelle et contacte Jimmy Page avec qui il a déjà travaillé lors de différentes sessions studio. Jimmy Page, connaissant le professionnalisme de John Paul Jones, l'accepte tout de suite. Pour le chanteur, Jimmy Page, pense tout d'abord à... en lire plus
Example: Recognize and link entities in Tweets

Raphaël Troncy
@rtroncy

Tampa Bay Lightning vs Canadiens in Montreal tonight with @erikmannens hockey NHL

https://www.youtube.com/watch?v=Rmug-PUylZl
Example: Recognize and link entities in Tweets

Raphaël Troncy
@rtroncy

Tampa Bay Lightning vs Canadiens in Montreal tonight with @erikmannens #hockey #NHL

Voir la traduction

17:52 - 10 avr. 2016

https://www.youtube.com/watch?v=Rmug-PUylzI
Part-of-Speech Tagging on Tweets

Tampa NNP
Bay NNP
Lightning NNP
vs CC
Canadiens NNP
in IN
Montreal NNP
tonight NN
with IN
@erikmannens USR
#hockey HT
#NHL HT

(N)ER: What is NHL?

(N)EL: Which Montreal are we talking about?

https://gate.ac.uk/wiki/twitter-postagger.html
What is NHL? Type Ambiguity
# (Named) Entity Recognition

<table>
<thead>
<tr>
<th>Entity</th>
<th>Tag</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tampa</td>
<td>NNP</td>
<td>ORG</td>
</tr>
<tr>
<td>Bay</td>
<td>NNP</td>
<td>ORG</td>
</tr>
<tr>
<td>Lightning</td>
<td>NNP</td>
<td>ORG</td>
</tr>
<tr>
<td>vs</td>
<td>CC</td>
<td>O</td>
</tr>
<tr>
<td>Canadiens</td>
<td>NNP</td>
<td>ORG</td>
</tr>
<tr>
<td>in</td>
<td>IN</td>
<td>O</td>
</tr>
<tr>
<td>Montreal</td>
<td>NNP</td>
<td>LOC</td>
</tr>
<tr>
<td>tonight</td>
<td>NN</td>
<td>O</td>
</tr>
<tr>
<td>with</td>
<td>IN</td>
<td>O</td>
</tr>
<tr>
<td>@erikmannens</td>
<td>USR</td>
<td>PER</td>
</tr>
<tr>
<td>#hockey</td>
<td>HT</td>
<td>THG</td>
</tr>
<tr>
<td>#NHL</td>
<td>HT</td>
<td>ORG</td>
</tr>
</tbody>
</table>
What is Montreal? Name Ambiguity

Montréal, Ardèche

Montréal, Aude

Montréal, Gers

Montréal, Québec

Montreal, Wisconsin
Popular Knowledge Bases
Named Entity Linking

Tampa NNP ORG http://dbpedia.org/resource/Tampa_Bay_Lightning
Bay NNP ORG http://dbpedia.org/resource/Tampa_Bay_Lightning
Lightning NNP ORG http://dbpedia.org/resource/Tampa_Bay_Lightning
vs CC O
Canadiens NNP ORG http://dbpedia.org/resource/Canadiens
in IN O
Montreal NNP LOC http://dbpedia.org/resource/Montreal
tonight NN O
with IN O
@erikkannens USR PER NIL
#hockey HT THG http://dbpedia.org/resource/Hockey
#NHL HT ORG http://dbpedia.org/resource/National_Hockey_League
Test with Babelfy, TagMe, Spotlight, AIDA and ADEL

- [http://babelfy.org/](http://babelfy.org/)
- [https://tagme.d4science.org/tagme/](https://tagme.d4science.org/tagme/)
- [https://dbpedia-spotlight.github.io/demo/](https://dbpedia-spotlight.github.io/demo/)
- [https://gate.d5.mpi-inf.mpg.de/webaida/](https://gate.d5.mpi-inf.mpg.de/webaida/)
Different Approaches

E2E approaches:
A dictionary of mentions and links is built from a referent KB. A text is split in n-grams that are used to look up candidate links from the dictionary. A selection function is used to pick up the best match.

Linguistic-based approaches:
A text is parsed by a NER classifier. Entity mentions are used to look up resources in a referent KB. A ranking function is used to select the best match.

ADEL is a combination of both to make a hybrid approach.
ADEL from 30,000 feet

ADEL

Entity Extraction

Entity Linking

Index
**Entity Extraction: Extractors Module**

- **POS Tagger:**
  - bidirectional CMM (left to right and right to left)

- **NER Combiner:**
  - Use a combination of CRF with Gibbs sampling (Monte Carlo as graph inference method) models. A simple CRF model could be:

  X set of features for the current word: word capitalized, previous word is “de”, next word is a NNP, ... Suppose \( P(\text{PER} \mid X, \text{PER}, O, \text{LOC}) = P(\text{PER} \mid X, \text{neighbors(\text{PER})}) \) then \( X \) with \( \text{PER} \) is a CRF

![Diagram of Extractors Module](image-url)
CRF Models Combination in details

- Apply multiple CRF models over the same piece of text
- Merge the results into one single output

Algorithm 1: Combining multiple CRF models

Result: Annotated tokens
Input : (Txt, M) with Txt the text to be annotated and M a list of CRF models
Output: A = List({token, label}) a list of tuples {token, label}

begin
  finalTuples ← EmptyList();
  foreach model in M do
    /* tmpTuples contains the tuples {token, label} got from model */
    tmpTuples ← apply model over Txt;
    foreach {token, label} in tmpTuples do
      if token not in finalTuples then
        add {token, label} in finalTuples;
      end
    end
  end
end
Entity Extraction: Overlap Resolution

- Detect overlaps among boundaries of entities coming from the extractors

- Different heuristics can be applied:
  - Merge: (“United States” and “States of America” => “United States of America”) default behavior
  - Simple Substring: (“Florence” and “Florence May Harding” => ”Florence” and “May Harding”)
  - Smart Substring: (“Giants of New York” and “New York” => “Giants” and “New York”)
Index: Indexing

- Use DBpedia and Wikipedia as knowledge bases
- Integrate external data such as PageRank scores from Hasso Plattner Institute
- Backend system with Elasticsearch and Couchbase
- Turn DBpedia and Wikipedia into a CSV-based generic format
**Entity Linking: Linking tasks**

- **Generate candidate links for all extracted mentions:**
  - If any, they go to the linking method
  - If not, they are linked to NIL via NIL Clustering module

- **Linking method:**
  - Filter out candidates that have different types than the one given by NER
  - ADEL linear formula:
    \[
    r(l) = (a \cdot L(m, \text{title}) + b \cdot \max(L(m, R)) + c \cdot \max(L(m, D))). PR(l)
    \]
  - \(r(l)\): the score of the candidate \(l\)
  - \(L\): the Levenshtein distance
  - \(m\): the extracted mention
  - \(\text{title}\): the title of the candidate \(l\)
  - \(R\): the set of redirect pages associated to the candidate \(l\)
  - \(D\): the set of disambiguation pages associated to the candidate \(l\)
  - \(PR\): Pagerank associated to the candidate \(l\)
  - \(a, b, c\) are weights following the properties: \(a > b > c\) and \(a + b + c = 1\)
## Results

- **ADEL over OKE2015**

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>extraction</td>
<td>85.1</td>
<td>89.7</td>
<td>87.3</td>
</tr>
<tr>
<td>recognition</td>
<td>75.3</td>
<td>59</td>
<td>66.2</td>
</tr>
<tr>
<td>linking</td>
<td>85.4</td>
<td>42.7</td>
<td>57</td>
</tr>
</tbody>
</table>

- **ADEL over OKE2016**

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>extraction</td>
<td>81.5</td>
<td>72.4</td>
<td>76.6</td>
</tr>
<tr>
<td>recognition</td>
<td>74.8</td>
<td>66.5</td>
<td>70.4</td>
</tr>
<tr>
<td>linking</td>
<td>52.8</td>
<td>45.8</td>
<td>49.1</td>
</tr>
</tbody>
</table>

- **ADEL over NEEL2016**

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>extraction</td>
<td>80.6</td>
<td>91.0</td>
<td>85.5</td>
</tr>
<tr>
<td>recognition</td>
<td>57.5</td>
<td>64.9</td>
<td>61.0</td>
</tr>
<tr>
<td>linking</td>
<td>49.9</td>
<td>58.3</td>
<td>53.8</td>
</tr>
</tbody>
</table>
Issues with current methods

- **Supervised methods**
  - Efficient but needs a training set for every dataset
  - Not robust enough if the type of text or entities change
  - Mostly associated with an E2E approach
  - Inappropriate to detect NIL entities

- **Unsupervised methods**
  - Difficult to compute the relatedness among the candidates of each entity
  - Graph-based of linear formula are sometimes long to compute
  - Difficult to manipulate emerging entities in case of graph-based approach
Deep Learning for Textual Content

https://youtu.be/mp6UsuRteNw?t=1h17m50s

http://y2u.be/cJIIlew6I28

Hi, I’m Cortana.
From Machine Learning to Deep Learning: Logistic Classifier

- Logistic Classifier $\Rightarrow$ Linear Classifier

\[ W \cdot X + b = Y \rightarrow \text{Softmax} \rightarrow S(Y) \rightarrow \text{Cross-Entropy} \rightarrow D(S(Y),L) \rightarrow L \]

- Multinomial Logistic Classification

- Cross-Entropy
  Evaluate how much the model deviates from the GS

- Weights, Input data, Bias, Scores (logits), Probabilities, 1-Hot Labels

- Trained
From Machine Learning to Deep Learning: Deep Neural Network

- **Neural Network => Non Linear Classifier**

\[
X \times W + b \rightarrow Y \rightarrow S(Y) \rightarrow L
\]

**Hidden Layers**

**ReLU: Rectified Linear Units (Activation Function)**
Why Understanding Language is Difficult?

- **Human language has great variability**
  - Similar concepts are expressed in different ways, (e.g. kitty vs cat)

- **Human language has great ambiguity**
  - Similar expressions mean different concepts, (e.g. New York vs New York Times)

- **The meaning of text is usually vague and latent**
  - No clear supervision signal to learn from

- **Learning semantic meaning of texts is a key challenge in NLP**
Word Embeddings

- Find a way to represent and measure how much two different words have same/similar meaning
- Need a huge amount of labelled data then better using an unsupervised approach
The purrs
This hunts mice

Context gives a good idea that words are similar
Goal is to predict words context in order to treat cat-like words similarly

Cat-like behaving the same way
Word Embeddings

- Map words to small vectors (embeddings)
- Embeddings are close to each other in the words space when they have similar meaning
**Word2Vec**

- Developed by Google in 2013
- Produce word embeddings
- Takes a large corpus of text as input and produce a vector space as output
- Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the vector space
- The goal is to provide semantically similar words to a given word
Word2Vec: How it works?

- Map every word to an embedding
- Use a window around a selected word
- Use the embedding of the selected word to predict the context of the word

The quick brown fox jumps over the lazy dog

window

Logistic classifiers

$V_{fox}$
Word2Vec: How it works?

- Measure the closeness of two word embeddings with cosine similarity is better than with L2 because the length is not relevant for the classification.

- Normalize all embeddings to get them in unit norm form.

\[
L_2 = \|V_{\text{cat}} - V_{\text{kitty}}\|_2^2
\]

\[
\text{cosine} = \frac{V_{\text{cat}} \cdot V_{\text{kitty}}}{\|V_{\text{cat}}\| \cdot \|V_{\text{kitty}}\|}
\]
Word2Vec schema

- Compares a target from the context of the input word
- Compute softmax over a huge vocabulary vector can be very inefficient
- To solve this issue, use sampled softmax

\[ \text{softmax}(\mathbf{W} \mathbf{V}_{\text{cat}} + \mathbf{b}) \]

random sample

\[
\begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
. \\
. \\
. \\
. \\
0 \\
0 \\
0 \\
. \\
. \\
. \\
\end{bmatrix}
\]

logistic classifier

cross entropy

purr
DSSM: Deep Structured Semantic Model

- Developed by Microsoft in 2013
- Compute similarity between vectors
- Generic enough for being applied to many more cases than what Word2Vec can do (Web search, ads, question answering, machine translation, word embeddings...)
- Training made with backpropagation
- The layers can be either: DNN, CNN or RNN
DSSM schema with CNN

Relevance measured by cosine similarity

Word sequence $x_t$

$w_1, w_2, \ldots, w_{TX}$

$w_1, w_2, \ldots, w_{TY}$

Learning: maximize the similarity between X (source) and Y (target)
DSSM schema with CNN

Relevance measured by cosine similarity

\[ \text{sim}(X,Y) \]

\[ 128 \]

\[ 128 \]

Learning: maximize the similarity between X (source) and Y (target)

Representation: use DNN to extract abstract semantic representations

Word sequence \( x_t \)

\[ w_1, w_2, \ldots, w_{TX} \]

\[ w_1, w_2, \ldots, w_{TY} \]

\[ f(.) \]

\[ g(.) \]
DSSM schema with CNN

Relevance measured by cosine similarity

\[
sim(X,Y)
\]

Learning: maximize the similarity between X (source) and Y (target)

Representation: use DNN to extract abstract semantic representations

Convolutional and max pooling layer: identify keywords [concepts] in X and Y

Word hashing: use letter-trigram as raw input to handle very large vocabulary

Semantic layer \( h \)

Max pooling layer \( v \)

Convolutional layer \( c_t \)

Word hashing layer \( f_t \)

Word sequence \( x_t \)

\[
X \rightarrow W_{1,TX} \rightarrow 300 \rightarrow 128 \rightarrow sim(X,Y) \rightarrow 128 \rightarrow 300 \rightarrow Y \rightarrow W_{1,TY}
\]
Conclusion

- Current methods for entity linking do not exploit enough semantics
- Deep Learning technics might be used to better take into account the semantic
- Using Word2Vec to rank the entity candidates from the most semantically similar to the less one
- Using DSSM to measure the relatedness between the candidates of each extracted mention
Questions?

Thank you for listening!

http://multimediasemantics.github.io/adel

http://jplu.github.io

julien.plu@eurecom.fr

@julienplu

http://www.slideshare.net/julienplu