ENTITY LINKING METHODS: TAGME VS FEL

Luca Lovagnini, Natural Language seminar. University of Pisa.
Entity linking deals with identifying entities from a knowledge base in a given piece of text.

TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities) Paolo Ferragina (University of Pisa), Ugo Scaiella (University of Pisa).

Fast and Space-Efficient Entity Linking in Queries, Edgar Meij (Yahoo Labs) Giuseppe Ottaviano (ISTI-CNR), Roi Blanco (Yahoo labs) (FEL).

We can identify three steps:
1. Identifying candidate mentions, i.e., which part(s) of the text to link
2. Identifying candidate entries for each mention
3. Disambiguating the candidate entities based on some notion of context and coherence

Rolling Stones didn’t play at Woodstock
Jan 2015
Specifically designed for search engine queries
Efficiency is crucial (usually not so considered in literature)
Forward-backward scanning procedure implemented with dynamic programming in $O(k^2)$
Candidates (aliases) are compressed through hashing and bit encoding
Probabilistic Model
Fast Entity Linker (FEL) compute a probabilistic score for each segment-entity pair
Wikipedia's anchor text used in FEL
Yahoo's Webscope search query log
(Almost) parameterless fashion (unlike TAGME)
- **aliases**: textual representation of an entity (for example «rock» could be an alias both for stone or music genre)
- **SxE event space** where S are all the sequences and E the entities
- s is a sequence of terms t∈s
- s represents a segmentation (sequence of sequences of terms) where s ∈ s is drawn from the set S
- e represents a set of entities e ∈ e, where each e is drawn from the set E
- a_s indicates if s is an alias
- a_s,e indicates if s is an alias pointing (linking/clicked) to e
- c indicates which collection acts as a source of information query log or Wikipedia (c_q or c_w)
- n(s,c) is the count of s in c
- n(e,c) is the count of e in c
- q input query
- S_q all possible segmentations of its tokens t_1,t_2,t_3,t_4 ...

**ANNOTATION**
Fel returns the set of entities \( e \), along with their scores, that maximizes (notice entities independence assumption)

\[
\arg\max_{e \in E} \log P(e|q) = \arg\max_{e \in E, s \in S_q} \sum_{e \in e} \log P(e|s) \quad (1)
\]

A possible alternative to (1) would be to select the segmentation that optimizes the score of the top ranked entity:

\[
\arg\max_{e \in E} \log P(e|q) = \arg\max_{e \in E, s \in S_q} \max_{e \in e, s \in s} P(e|s) \quad (2)
\]

FEL – LINKING SEGMENTS TO ENTITIES
Now we define the probability that a segment $s$ is linked to an entity $e$:

$$P(e|s) = \sum_{c \in \{c_q, c_w\}} P(c|s) P(e|c, s)$$

Knowing that $s$ is in $c$, then it points to $e$:

$$= \sum_{c \in \{c_q, c_w\}} P(c|s) \sum_{a_s = \{0, 1\}} P(a_s | c, s) P(e|a_s, c, s)$$

$s$ in $c$ is an alias:

$$s \text{ is an alias in } c \quad \sum_{s:as=1} n(s,c) \quad \sum_{s:a_s=1} n(s,c)$$

alias $s$ in $c$ points to entity $e$:

$$\frac{\sum_{s:as=1} n(s,c)}{n(s,c)} \quad \frac{\sum_{s:a_s=1} n(s,c)}{\sum_{s:a_s=1} n(s,c)}$$

**FEL - LINKING SEGMENTS TO ENTITIES**
The maximum likelihood probabilities reported in the previous slide can be smoothed (estimated) with an add-one, Dirichlet and Laplace smoothings.

Some term of the input query might not be covered by any segment $s$, so it was defined a special entity $not\_linked$ where $P(not\_linked \mid s) = l$ where $l$ is a parameter which value depends on how many links we want to obtain from the algorithm.
Let $t = t_1, t_2, t_3, ..., t_k$ a sequence of terms and let $[t_i, ..., t_{i+j}], \forall i, j \geq 0$ any segment of the sequence.

Let $\gamma(s)$ be any scoring function that maps segment to real numbers.

Let $\Phi(a, b)$ an aggregation function ($\Phi(a, b) = a + b$ for (1) and $\Phi(a, b) = \max(a, b)$).

Then the maximum score of a segmentation is defined as follows:

$$m(t_1, t_2, ..., t_k) = \max(\Phi(\gamma(t_1), m(t_2, ..., t_k)), \Phi(\gamma([t_1, t_2]), m(t_3, ..., t_k)), ..., \Phi(\gamma([t_1, ..., t_k]))$$
Algorithm 1 Entity-linking algorithm

Require: A user query $q$, a function $\text{HIGHESTSCORE}(\cdot)$, and an aggregation function $\phi(\cdot, \cdot)$.

1: $p \leftarrow \text{TOKENIZE}(q)$
2: $l \leftarrow \text{LENGTH}(p)$
3: maxscore[] $\leftarrow$ new array[$l + 1$]
4: previous[] $\leftarrow$ new array[$l + 1$]
5: for $i = 0$ to $l$ do
6:     for $j = 0$ to $i$ do
7:         score $\leftarrow \phi(\text{maxscore}[j], \text{HIGHESTSCORE}(p[j : i], q))$
8:         if score $>$ maxscore[$i$] then
9:             maxscore[$i$] $\leftarrow$ score
10:            previous[$i$] $\leftarrow j$
11:        end if
12:     end for
13: end for
14: return maxscore[$l$]
FEL – AN EXAMPLE
Cannot distinguish «Brad Pitt seven» from «Brad Pitt olympics»

The contextual relevance model estimates the probability that an entity e is relevant to the whole query (the context)

\[ \gamma(s) = \max_{e \in E} \log(P(e|s, q)) = P(e|s) \prod_t \frac{P(t|e)}{P(q)} \] where \( P(t|e) \) is the probability that a term \( t \) is relevant to \( e \)

Supposing the distributional semantics hypothesis (if \( t_1 \) is relevant to \( e \) and \( t_2 \) similar to \( t_1 \), then \( t_2 \) is relevant to \( e \) as well), a the Google’s word2vec tool (3 billion of words from Wikipedia) is used so words that are close in meaning are mapped to vectors close in cosine distance.

So we define \( P(t|e) = \sigma([v_t, 1] v_e) \) where \( v_t, v_e \in \mathbb{R}^D \) are the vectors relative to the word \( t \) and the entity \( e \) (where \( D \) is usually 200) and \( \sigma(x) = \frac{1}{1+e^{-x}} \)

Without going too much in details (for time constraints) if \( (D=200) \) then the complexity to score each entity \( e \) with a query is \( O(kD) \) and the space needed would be \( 4(E(D + 1) + WD) \) (vector compression implemented)

Centroids can be used to obtain \( O(D) \) (negligible improvement)
- 2010
- Google Research Awards winner
- Considered the «state-of-the-art» about entity linking
- Uses Wikipedia as knowledge base
- High precision/recall, annotates short fragments of text with pertinent hyperlinks to Wikipedia articles.
- The main feature is that it may annotate texts which are short and poorly composed, such as snippets of search-engine results, tweets, news, etc.

TAGME
The spots are the sequence of terms in the input text which are to be annotated.

Wikipedia anchor texts as spots and pages linked to them in Wikipedia as their possible senses (about 3 millions as dictionary).

Ambiguity and polysemy solved between the potentially many available anchor-page mappings by finding the collective agreement among them via new scoring functions.

The time complexity of TAGME’s annotation is linear in the number of processed anchors.

TAGME
TAGME WORKFLOW

PARSING

DISAMBIGUATION
by a voting scheme

PRUNING
2 simple features

Voting scheme

Select top-\(\epsilon\) pages

Select the best in commonness
Wikipedia page $p$
Text $a$ of $p$ (maybe an anchor)
$Pg(a)$ is the set of all Wikipedia pages linked by $a$ (same anchor may occur many times pointing to different pages)
$freq(a)$ number of times that $a$ occurs in Wikipedia (as an anchor or not)
$link(a)$ number of times that $a$ occurs as an anchor in Wikipedia (so $link(a) \leq freq(a)$)
$lp(a) = \frac{link(a)}{freq(a)}$ to denote the link-probability that an occurrence of $a$ has been set as an anchor (for example: $lp(\text{house})$ low $<$ $lp(\text{Barack})$). 
$Pr(p | a)$ is the commonness, so the probability that an occurrence of an anchor $a$ point to $p \in Pg(a)$ (for example: $Pr(\text{Apple Inc. page} \mid \text{«apple»}) > Pr(\text{Apple fruit page} \mid \text{«apple»})$)
$a \rightarrow p$ represents the annotation of an anchor $a$ with some page $p \in Pg(a)$
If $Pg(a) > 1$ (so $a$ has more senses), we call disambiguation the process of selecting one of the possible senses of $a$ from $Pg(a)$.
- Short text as inputs
- Tokenization
- Anchor detection by querying the Anchor Dictionary for sequences up to 6 words
- Define «anchor boundaries»: if anchor $a_1$ is a substring of anchor $a_2$ we drop $a_1$ if $lp(a_1) < lp(a_2)$ (since $a_1$ is usually more ambiguous)
- For example $a_1=«\text{jaguar}»$ $a_2=«\text{jaguar car}»$ ($lp(a_1) < lp(a_2)$): the anchor jaguar would slow down the process
- While $a_1=«\text{act}»$ $a_2=«\text{the act}»$ (band, small number of link occurrences, $lp(a_1) > lp(a_2)$: both words are kept because we’re not able to make a principled pruning.
Let $A_t$ be the set of all anchors occurring in a short input text $T$.

For each anchor $a \in A_t$ and for each possible sense of $p_a$ of $a$, TAGME computes a score via a «collective agreement» between the sense $p_a$ and the sense of all others anchors in $T$.

The agreement score of $a \rightarrow p_a$ is evaluated by means of a voting scheme, where each anchor $b$ gives a vote (score) to $p_a$. 
The vote that $b$ gives to $p_a$ is based on the average relatedness between each sense $p_b$ of $b$ and the sense $p_a$ that we want to associate to $a$.

$$rel(p_a, p_b) = \frac{\log(\max(\text{in}(p_a), \text{in}(p_b))) - \log(\text{in}(p_a) \cap \text{in}(p_b))}{\log(W) - \log(\min(\text{in}(p_a), \text{in}(p_b)))}$$

Where $\text{in}(p)$ is the number of Wikipedia pages pointing to page $p$ and $W$ is the total number of pages in Wikipedia.

Pages that contain both terms indicate relatedness, while pages with only one of the terms suggest the opposite.

ANCHOR DISAMBIGUATION - RELATENESS
Hence the vote given by an anchor $b$ to the annotation $a \rightarrow pa$ is

$$vote_b(pa) = \sum_{p_b \in Pg(b)} \frac{rel(p_b, pa) \ast Pr(p_b|b)}{|Pg(b)|}$$

$Pr(p_b|b)$ is used because not all senses of $b$ have the same statistical significance.

Finally, the total score for the annotation $a \rightarrow pa$ results

$$rel_a(pa) = \sum_{b \in A \setminus a} vote(b, pa)$$

This score is combined with the commonness of the sense $p_a$.

All senses with a commonness smaller than a given threshold are discarded.
Now that we have computed the score for each possible sense of \( p_a \in Pg(a) \), we have to choose the best one.

First, we determine the sense \( p_{\text{best}} \) that achieves the highest relatedness \( rel_a (p_{\text{best}}) \) with the anchor \( a \).

Then identifies the set of other senses in \( Pg(a) \) that yield about the same value of \( rel_a (p_{\text{best}}) \), according to some fixed threshold \( \epsilon \).

Finally, TAGME annotates \( a \) with the sense \( p_a \) that obtains the highest commonness \( Pr(p_a | a) \) among these top- \( \epsilon \) senses.
After the disambiguation phase, we have to prune the unmeaningful annotations from the set of candidate annotations.

The goal of the pruning phase is to keep all anchors whose link probability ($lp$) is high or whose assigned sense (page) is coherent with the senses (pages) assigned to the other anchors.
The coherence is based on the average relatedness between the candidate sense $p_a$ and the candidate senses $p_b$ assigned to all other anchors $b$.

We define as $S$ the set of distinct senses, then the coherence

$$\text{coherence}(a \rightarrow p_a) = \frac{1}{|S-1|} \sum_{p_b \in S \setminus p_a} \text{rel}(p_b, p_a)$$
For each candidate a **pruning score** is computed

\[ \rho(a \rightarrow pa) = \frac{lp(a) + coherence(a \rightarrow pa)}{2} \]

If \( \rho(a \rightarrow pa) < \rho_{na} \) (where \( \rho_{na} \) is a given threshold) then that annotation for \( a \) is discarded.
Diego Maradona won against Mexico.

Which one?

• The text Diego isn’t necessary an anchor
• Instead Diego Maradona is generally an anchor
• $lp(Diego) < lp(Diego\ Maradona)$ so the first one is discarded
Diego Maradona won against Mexico.

High score!

- Diego A. Maradona
- Diego Maradona Jr.
- Maradona Stadium
- Maradona Film

- Mexico Football Team
- Mexico Nation
- Mexico State
- Mexico Baseball Team

TAGME – DISAMBIGUATION
Diego Maradona won against Mexico.

Let’s suppose that the disambiguation threshold $\varepsilon$ is 0.6

| Sense                  | Score | Pr(p | a): Commonness |
|------------------------|-------|---------------------|
| Maradona Stadium       | 1.5   | 0.05                |
| Diego A. Maradona      | 1.4   | 0.75                |
| Maradona Film          | 1.3   | 0.05                |
| Diego Maradona Jr.     | 0.5   | 0.15                |

TAGME - DISAMBIGUATION
Diego Maradona won against Mexico.

DISAMBIGUATION!

Diego A. Maradona

Football at the 1996 Summer Olympics

Sepultura's Album

Mexico Football Team

PRUNED!

With low threshold

TAGME EXAMPLE - PRUNING
TAGME

- $O(d_{in}(ns)^2)$ ($d_{in}$=avg in-degree Wikipedia’s page, n number of anchors, s avg senses per anchor)
- Not specifically designed for efficiency (totally in java!)
- Short text as inputs (more general)
- Only Wikipedia
- 3 phases could be bad: considered only certain entities
- Link probability for parsing
- Voting scheme for disambiguation
- Coherence for pruning
- Much more simple

FEL

- $O(k^2)$
- Efficiency is crucial: optimization techniques implemented (early stopping and compression)
- Queries as input
- Wikipedia and Yahoo
- Probabilistic Model
- Each segment is independent
- Efficient through dynamic programming
- (Almost) parameterless
- Basic version without context
- Much more complicated
From FEL paper

Performance on the Yahoo’s Webscope dataset

- 4 early precision metrics: Precision at Rank 1, Mean Reciprocal Rank, Mean Average Precision and R-Precision (bigger is better)

Unfortunately, FEL efficiency wasn't compared with TAGME: the AVG Time is taken from the two papers

TAGME VS FEL
We have seen two different entity linking methods

TAGME is older and introduced new concepts and improvements

FEL is one of the newest methods, and was specifically designed to overall all the state-of-art methods and for web search engine queries

Typical IR-based approach to indexing, clustering, classification and retrieval -> bag-of-words paradigm

Natural Language Processing, machine learning, big data and concepts like context, relevance, relatedness are the new frontline to create a relation between user input text and entities.
QUESTIONS?