Hermes:
A distributed messaging tool for NLP

Ilaria Bordino, Andrea Ferretti, Marco Firrincieli, Francesco Gullo, Marcello Paris, and Gianluca Sabena
UniCredit R&D

{ilaria.bordino, andrea.ferretti2, marco.firrincieli, francesco.gullo, marcello.paris, gianluca.sabenag}@unicredit.eu

August 26th, 2016

Natural Language Processing (NLP)

"Set of techniques for automated generation, manipulation and analysis of human (natural) languages"

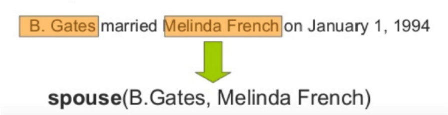
Major tasks:

- Language modeling
- Part-of-speech (POS) tagging
- Entity recognition and disambiguation
- Sentiment analysis
- Word sense disambiguation

What for? Information Extraction Tasks

Entity recognition and disambiguation

Relation Extraction



What for? Information Extraction Tasks

Event Extraction

„San Bernardino, California was struck by a moderate earthquake on Thursday night, with shaking felt from Los Angeles to Orange County.

A preliminary reading by the U.S. Geological Survey showed a 4.5-magnitude quake struck at 7:49pm. ...“

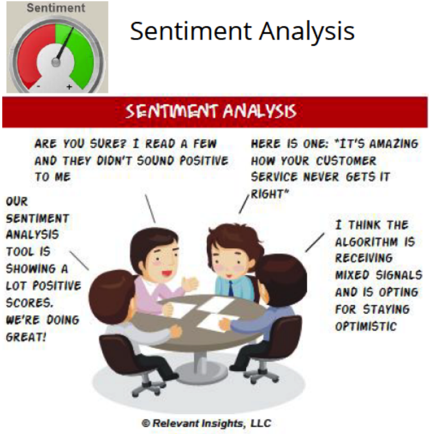
Event type: earthquake

Roles:

- **magnitude** - *What was the magnitude of the earthquake?*
- **location** - *Where did the earthquake occur?*
- **time** - *At what time did the earthquake occur?*
- ...

What for? Information Extraction Tasks

Sentiment Analysis



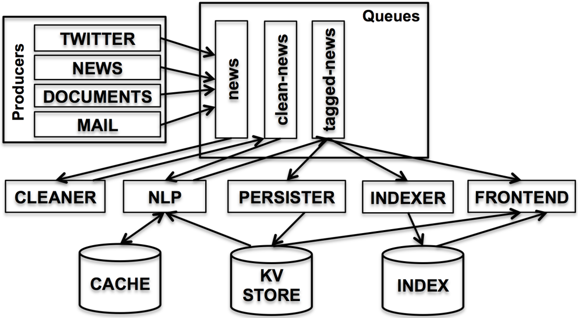
Use Cases

- Online Reputation Management
- Opinion Mining
- Automatic Summarization
- Question Answering

A distributed-messaging tool for NLP

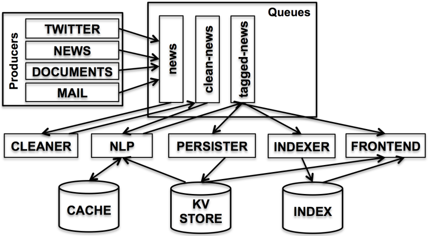
1. Efficient and extendable architecture: independent modules interact via message passing
2. Large scale processing
3. Completeness
4. Versatility

- Three queues implemented as kafka topics
- All modules written in Scala
- All messages are JSON strings



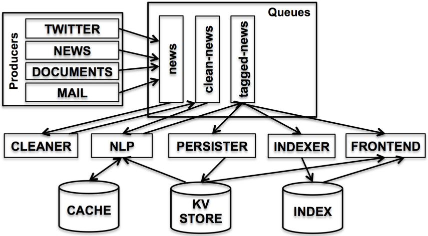
Producers

- Retrieve the text sources to be analyzed, and feed them into the system
- Four different source types are currently supported:
  1. Twitter
  2. News articles
  3. Documents
  4. Mail messages
- Producers perform minimal processing and push on the *news* queue



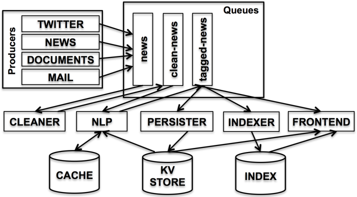
Cleaner

- Consumes raw news pushed on the news queue
- Performs text extraction
  - *Goose* is used for text extraction
  - *Tika* for content extraction and language recognition
- Pushes extracted text onto the *clean-news* queue



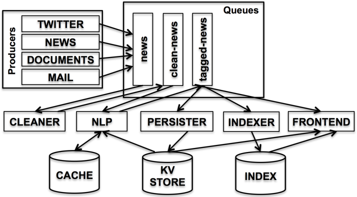
NLP Module

- Handles sentence splitting, tokenization, HTML/Creole parsing, entity linking, topic detection, clustering of related news, sentiment analysis
- *Client/Server Design*: The client news on the clean-news queue, asks for NLP annotations to the service, and places the result on the tagged-news queue
- The service is an Akka application providing APIs to the NLP tasks



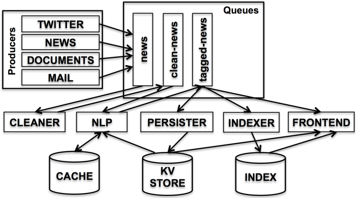
Persister and Indexer

- Index service: ElastichSearch
- Key-value store: HBase
- Two long-running Akka applications listen to the clean-news and tagged-news queues, and respectively index and persist raw and decorated news



Frontend

- A single-page client (written in Coffee-Script using Facebook React) interacts with a Play application
- The client home page shows annotated news ranked by a relevance function that combines various metrics but users can also search.
- The Play application retrieves news from the index and enriches them with content from the key-value store.



NLP: dealing with (named) entities

Entity: concept of interest in a text (e.g., a person, a place, a company)

Entity Recognition and Disambiguation (**ERD**):

- Entity Recognition (**ER**):
  identification of (candidate) entities in a plain text (i.e., which parts of the text to be linked)
- Entity Disambiguation (**ED**), aka Entity Linking (**EL**):
  resolving (i.e., "linking") named entity mentions to entries in a structured knowledge base

Non-uniform terminology: in some cases EL $\equiv$ ERD

Solving ERD

We need a knowledge base! $\Rightarrow$ e.g., Wikipedia

- Mentions: anchor text of all Wikipedia hyperlinks (pointing to a Wikipedia page)
- Entities: all Wikipedia pages
- Mentions and entities are connected by a one-to-many relationship (a specific anchor text can point to several Wikipedia pages)
- Entities are connected to each other in a graph structure (arcs $\equiv$ hyperlinks)

Offline step: scan Wikipedia corpus and take (1) anchor text of all Wikipedia hyperlinks, (2) all Wikipedia pages (=entities) pointed by each anchor text, and (3) all hyperlinks among Wikipedia pages (to infer the Wikipedia graph structure)

Entity linking: voting approach

WIKIFY! [Mihalcea and Csomai, CIKM'07]

TAGME [Ferragina and Scaiella, CIKM'10]

WAT [Piccinno and Ferragina, ERD'14]

Main idea

Compute a score for each candidate mention-entity linking $a \mapsto e$ (based on the other possible mention-entity linkings $b \mapsto e'$ derived from the input text), and link each mention a to the entity e^* that maximizes that score, i.e., $e^* = \arg\max_e score(a \mapsto e)$.

Entity linking: voting approach

Relatedness between two entities (Wikipedia pages) e_1 and e_2 (directly proportional to the in-neighbors shared by e_1 and e_2) [Milne and Witten, CIKM'08]:

$$rel(e_1, e_2) = 1 - \frac{\max\{\log |in(e_1)|, \log |in(e_2)|\} - \log |in(e_1) \cap in(e_2)|}{|W| - \min\{\log |in(e_1)|, \log |in(e_2)|\}}$$

Vote given by mention b to the candidate mention-entity linking $a \mapsto e$:

$$vote(a \mapsto e \mid b) = \frac{1}{|E(b)|} \sum_{e' \in E(b)} rel(e, e') \Pr(e' \mid b)$$

Ultimate score for the candidate mention-entity linking $a \mapsto e$:

$$score(a \mapsto e) = \sum_{b \in \mathcal{M}_T \setminus \{a\}} vote(a \mapsto e \mid b)$$

Voting-based entity linking: critical steps

- $rel(e_1, e_2) = 1 - \frac{\max\{\log|in(e_1)|, \log|in(e_2)|\} - \log|in(e_1) \cap in(e_2)|}{|W| - \min\{\log|in(e_1)|, \log|in(e_2)|\}}$

$\Rightarrow \mathcal{O}(\min\{deg(e_1), deg(e_2)\})$

- $score(a \mapsto e) = \sum_{b \in \mathcal{M}_T \setminus \{a\}} vote(a \mapsto e \mid b) = \frac{1}{|E(b)|} \sum_{\substack{b \in \mathcal{M}_T \setminus \{a\}, \\ e' \in E(b)}} rel(e, e') \Pr(e' \mid b)$

  for all possible $a \mapsto e$

$\Rightarrow \mathcal{O}(N^2)$ $(N = \sum_{m \in \mathcal{M}_T} |E(m)|)$

MinHash

Method for quickly estimating the similarity between two sets

- U: universe of elements, $A, B \subseteq U$: any two sets
- Jaccard similarity coefficient: $J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$
- Hash function $h : U \to I \subseteq \mathbb{N}$
- For any set $S \subseteq U$, let $h_{min}(S) = \min_{x \in S} h(x)$

$$\Downarrow$$

MinHash argument:

- $h_{min}(A) = h_{min}(B)$ if $x_{min} = \arg\min_{x \in A \cup B} h(x) \ \in A \cap B$
  $\Rightarrow \Pr[h_{min}(A) = h_{min}(B)] = \frac{|A \cap B|}{|A \cup B|} = J(A,B)$
  $\Rightarrow$ rnd variable $r := \mathbb{1}[h_{min}(A) = h_{min}(B)]$ is an unbiased estimator of $J(A,B)$
- Problem: r has a too large variance ($r \in \{0,1\}$, while $J \in [0,1]$)
  $\Rightarrow$ Use multiple hash functions $h^{(1)}, \ldots, h^{(K)}$ and estimate $J(A,B)$ as
  $\frac{1}{K} \sum_{i=1}^{K} \mathbb{1}[h^{(i)}_{min}(A) = h^{(i)}_{min}(B)]$

MinHash applied to Milne-Witten function

Problem: given two entities e_1 and e_2, and their corresponding neighbor sets $\mathcal{N}_1$ and $\mathcal{N}_2$ (with $|\mathcal{N}_1| = deg(e_1)$, $|\mathcal{N}_1| = deg(e_2)$), quickly estimate $|\mathcal{N}_1 \cap \mathcal{N}_2|$

Offline (n:#entities, m:#edges in the entity-interaction graph (e.g., Wikipedia)):

- Choose K hash functions $h^{(1)}, \ldots, h^{(K)} \rightarrow [\mathcal{O}(Kn)]$
  - basically, if our universe $U = \{1, \ldots, n\}$ corresponds to the id of the n entities in our dataset, each $h^{(i)}$ is a random permutation of U
- Compute min-hash signature of each entity e as a K-dimensional real-valued vector $\vec{v}_e = [h^{(1)}_{min}(\mathcal{N}(e)), \ldots h^{(K)}_{min}(\mathcal{N}(e))] \rightarrow [\mathcal{O}(K \sum_e deg(e)) = \mathcal{O}(Km)]$

Online:

- Estimate $J(\mathcal{N}(e_1), \mathcal{N}(e_2))$ as $\frac{1}{K} \sum_{i=1}^{K} \mathbb{1}[\vec{v}_{e_1}(i) = \vec{v}_{e_2}(i)]$
- Estimate $|\mathcal{N}(e_1) \cap \mathcal{N}(e_2)|$ as $\frac{J}{1+J}(|\mathcal{N}(e_1)| + |\mathcal{N}(e_2)|)$
- $\rightarrow$ $[\mathcal{O}(K)]$ (rather than $\mathcal{O}(\min\{deg(e_1), deg(e_2)\})$)

LSH to speed-up voting-based EL

Offline:

- Compute LSH buckets $lsh(e) = [b_1(e), \ldots, b_L(e)]$ for each entity e, where $b_i(e) = lsh(i, minhash(e)) \rightarrow$ $[\mathcal{O}(Ln\frac{K}{L}) = \mathcal{O}(Kn)]$ (+ $[\mathcal{O}(Km)]$ for MinHash)

Online (given an input text T):

- Retrieve LSH buckets for all entities in T
- Compute inverted index: for each bucket b, $entities(b) = \{e \mid b(e) \in lsh(e)\}$
- Approximate $score(a \mapsto e) = \frac{1}{|E(b)|} \sum_{\substack{b \in \mathcal{M}_T \setminus \{a\},\\ e' \in E(b)}} rel(e, e') \Pr(e' \mid b)$ as

  $\frac{1}{|E(b)|} \sum_{e' \in buckets(e)} rel(e, e') \Pr(e' \mid b)$

Instead of $\mathcal{O}(N^2)$ comparisons, only need comparisons between entities in the same bucket

Check out our tool at
`hermes.rnd.unicredit.it:9603`
(Email me to get access credentials)

Thanks!