Advancing NLP via a distributed-messaging approach

Ilaria Bordino, Andrea Ferretti, Marco Firrincieli, Francesco Gullo, Marcello Paris, Stefano Pascolutti, and Gianluca Sabena

UniCredit, R&D Dept., Italy

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

2016 IEEE International Conference on Big Data
December 5-8, 2016
Washington D.C., USA

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

Hermes: A distributed-messaging tool for NLP

Motivations:

- Architectural limitations: Existing solutions are stand-alone components focusing on specific micro-tasks, nor really suitable for distributed environments and large-scale data processing
- Algorithmic limitations: Existing entity recognition and disambiguation (core NLP task) methods not really amenable to be deployed in a real-world industrial context (weaknesses in terms of both efficiency and result interpretability)

Hermes: A distributed-messaging tool for NLP

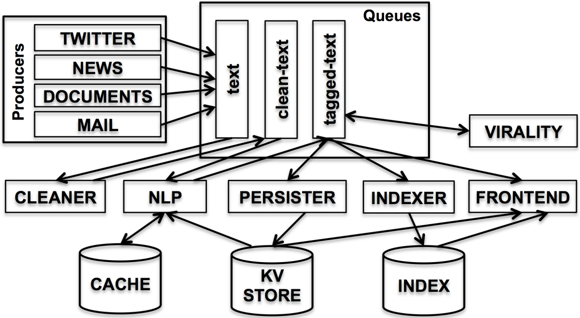
Contributions:

- We design Hermes, a novel NLP tool that overcomes the aforementioned state-of-the-art limitations
- Architectural contribution: Efficient and extendable architecture whose modules interact via message passing
  - Three major requirements satisfied:
    capability of large-scale processing, completeness, versatility
- Algorithmic contribution: Novel solutions to entity recognition and disambiguation aiming at both efficiency and result interpretability

Architecture

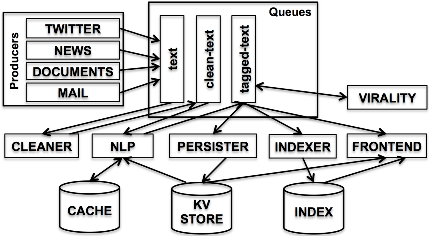
Message queues

- Queues, producers, consumers
- Implementation details: Scala, Apache Kafka, JSON



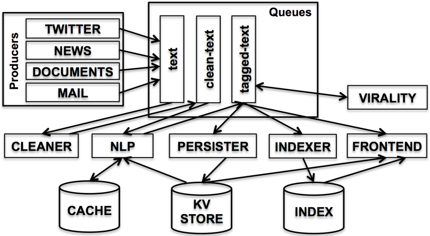
Producers

- 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 *text* queue



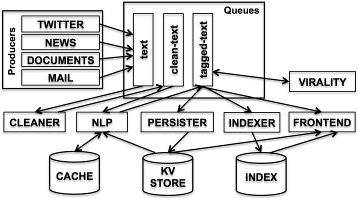
Cleaner

- Consumes raw texts pushed on the text queue
- Performs text extraction
- Pushes extracted text onto the *clean-text* queue
- Implementation details: Goose for text extraction, Tika for content extraction and language recognition



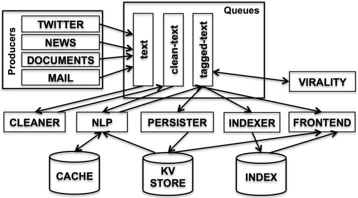
NLP Module

- 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



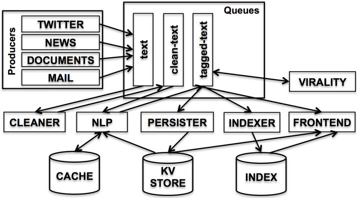
Persister and Indexer

- Index service (ElasticSearch)
- Key-value store (HBase)
- Two long-running (Akka) applications listen to the clean-text and tagged-text 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.



Algorithms

NLP: dealing with (named) entities

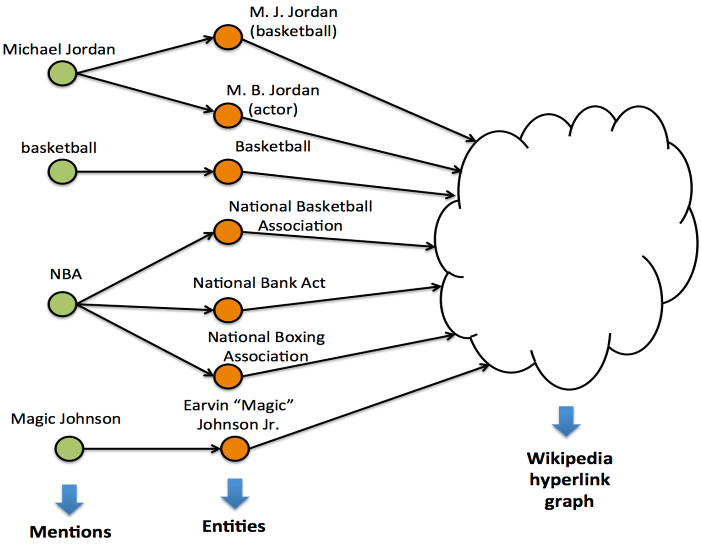
Entity: concept of interest in a text (e.g., a person, a place, a company)

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

Entity linking: scenario



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., $e^* = \arg\max_e score(a \mapsto e)$.

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 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

Experiments

Wikipedia

	Exact	LSH	LSH-MinHash
P	0.825	0.808	0.806
R	0.629	0.638	0.629
FM	0.697	0.692	0.685
Tsetup (ms)	80050	115844	45111
Tdis (ms)	758	153	32

RSS

	Exact	LSH	LSH-MinHash
P	0.67	0.65	0.65
R	0.66	0.64	0.64
FM	0.663	0.643	0.643
Tsetup (ms)	37697	50836	27950
Tdis (ms)	67	49	21

Reuters

	Exact	LSH	LSH-MinHash
P	0.676	0.603	0.604
R	0.524	0.479	0.478
FM	0.579	0.525	0.525
Tsetup (ms)	198734	232641	66847
Tdis (ms)	3106	370	34

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

Thanks!