Combining multiple signals for semanticizing tweets: University of Amsterdam at #Microposts2015
Gârbacea, C.; Odijk, D.; Graus, D.P.; Sijaranamual, I.; de Rijke, M.

Published in:
5th workshop on ‘Making Sense of Microposts’ at World Wide Web Conference 2015

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Combining multiple signals for semanticizing tweets: University of Amsterdam at #Microposts2015

Cristina Gârbacea, Daan Odijk, David Graus, Isaac Sijaranamual, Maarten de Rijke
University of Amsterdam, Science Park 904, Amsterdam, The Netherlands
{G.C.Garbacea, D.Odijk, D.P.Graus, I.B.Sijaranamual, deRijke}@uva.nl

ABSTRACT
In this paper we present an approach for extracting and linking entities from short and noisy microblog posts. We describe a diverse set of approaches based on the Semanticizer, an open-source entity linking framework developed at the University of Amsterdam, adapted to the task of the #Microposts2015 challenge. We consider alternatives for dealing with ambiguity that can help in the named entity extraction and linking processes. We retrieve entity candidates from multiple sources and process them in a four-step pipeline. Results show that we correctly manage to identify entity mentions (our best run attains an F1 score of 0.809 in terms of the strong mention match metric), but subsequent steps prove to be more challenging for our approach.

Keywords
Named entity extraction; Named entity linking; Social media

1. INTRODUCTION
This paper describes our participation in the named entity extraction and linking challenge at #Microposts2015. Information extraction from microblog posts is an emerging research area which presents a series of problems for the natural language processing community due to the shortness, informality and noisy lexical nature of the content. Extracting entities from tweets is a complex process typically performed in a sequential fashion. As a first step, named entity recognition (NER) aims to detect mentions that refer to entities, e.g., names of people, locations, organizations or products (also known as entity detection), and subsequently to classify the mentions into predefined categories (entity typing). After NER, named entity linking (NEL) is performed: linking the identified mentions to entries in a knowledge base (KB). Due to its richness in semantic content and coverage, Wikipedia is a commonly used KB for linking mentions to entities, or deciding when a mention refers to an entity that is not in the KB, in which case it is referenced by a NIL identifier. DBpedia aims to extract structured information from Wikipedia, and combines this information into a huge, cross-domain knowledge graph which provides explicit structure between concepts and the relations among them.

Our participation in this challenge revolves around the existing open-source entity linking software developed at the University of Amsterdam. We use Semanticizer, a state-of-the-art entity linking framework. So far Semanticizer has been successfully employed in linking entities in search engine queries [1] and in linking entities in short documents in streaming scenarios [6]. Moreover, it has been further extended to deal with additional types of data like television subtitles [3]. In what follows we explain how we use Semanticizer for the task at hand, and describe each of our submitted runs to the competition.

2. SYSTEM ARCHITECTURE
Our system processes each incoming tweet in four stages: mention detection, entity disambiguation and typing, NIL identification and clustering, and overlap resolution. We explain each stage in turn.

Mention detection: The first step aims to identify all entity mentions in the input text, and is oriented towards high recall. We take the union of the output of two mention identification methods: Semanticizer, the state-of-the-art system performs lexical matching of entities’ surface forms. These surface forms are derived from the KB, and comprise anchor texts that refer to Wikipedia pages, disambiguation and redirect pages, and page titles as described in Table 1. For this, we use two instances of Semanticizer, running on two Wikipedia dumps: one dated May 2014 (the version used to build DBpedia 3.9), and a more recent one, dated February 2015.

We perform three separate preprocessing steps on the tweet text, the results of which get sent to the Semanticizer. These steps are: i) the raw text, ii) the cleaned text (replacing @-mentions with corresponding Twitter account names, and splitting hashtags using dynamic programming), and iii) the normalized text (e.g., case-folding, removing diacritics).

NER: For identifying entity mentions that do not exist in Wikipedia, i.e., out of KB entities, we employ a state-of-the-art named entity recognizer, previously applied to finding mentions of emerging entities on Twitter [2]. We train five different NER models, three using the ground truth data from the Microposts challenges (2013 through 2015), one using pseudo-ground truth (generated by linking tweets as in [2]) and one trained on all data.

Given the candidate mentions identified by NER and Semanticizer, we include a binary feature to express whether the mention has been detected by both systems. For each mention we end up with the set of features described in Table 1 that we use in training a Random Forest classifier (using 100 trees and rebalancing the classes per tweet by modifying instance weights), to predict whether a candidate mention is an entity mention (actually refers to an entity).
We expect the NER runs to be superior to Semanticizer or ES-only (non-linked mentions get a fixed score). We then find the highest threshold of all non-overlapping mentions, and assign them their link score that are output by the mention identification step. We create a graph mapping of DBpedia classes to the #Microposts2015 taxonomy. For NIL detection. We cluster NILs by linking identical mentions to instances of RankSVM to rank the entity candidates for each mention, and take the top ranked candidate as the entity to link. We map the entity to its DBpedia URI, and determine its type through a manual query. Finally, we incorporate Wikipedia page view statistics. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features. We index Wikipedia using ElasticSearch (ES), and issue features.