

Fast Search for NER

Basic Algorithms for Computational Linguistics

Project

Sibel Ciddi*, Noushin Fadaie†, David Przybilla‡

July 14, 2012

1 Introduction

NERSimString is a Java library based on SimString [1] paper for doing fast approximate String retrieval.

The main idea of *NERSimString* is to provide a library for creating Name Entity dictionaries and a fast NE annotation tool given the generated entities dictionaries.

Gathering Name entities has been a popular tasks during the past years, there are both statistical methods and rule based methods which are good at the task of extracting Name Entities(NE) from raw text.

By Using this tools long list of NE can be created and stored in dictionaries, so that later can be used as a fixed list.

However the following problems arise:

- Even in a restricted domain, the number of NE in a dictionary can be overwhelming.
- Given fixed dictionaries the final goal of finding all the entities contained in a corpus can be a hard task given the size of the corpus.
- Even semi-supervised methods for learning to recognize NE require fast search tools, since learning requires for example to find where a given set of NE are located in a corpus.

So a way for looking for words quickly in a huge dictionary is necessary in order to make the annotation task of NE a tractable problem, this is the aim of *NERSimString*.

*sciddi@coli.uni-saarland.de

†fadaei.noushin@gmail.com

‡dav.alejandro@gmail.com

Why not exact lookups?

Exact lookups are ok for some tasks, how ever not for NE recognition and specially when dealing with user-generated data (i.e: social networks) why? because there can be variations or miss spelled words, that's why *NERSimString* offers fast-Approximate String Retrieval

What does NERSimString offer?

It offers a quick way to make search on the cost of space.

Basically *NERSimString* receives a dictionary with a NE entry per line(this entry can be multiword) as a result it creates a *NerSimString* dictionary in which fast queries can be done. Those dictionaries come in different flavors, according to different low-level-implementation

- **SuffixTree:** it is a dictionary based on a suffixTree hold in memory
- **Naive-HashTable:** it is a dictionary based on hashTables hold in memory
- **MemoryMapped Hashtable:** it is a dictionary based on hashmaps, it is meant to be for huge amounts of data since it is stored dynamically in the disk in order to avoid being loaded completely into memory.

What kind of similarities?

In order to provide approximate String retrieval the following similarity measures are supported:

- Jaccard
- Dice
- Cosine

As a result this tool can be used either to annotate or to extend current systems that learn to predict NE.

Why is it fast?

NERSimString works creating an inverted list of ngrams-size. Basically given a similarity measure, and a threshold it measures how many ngrams of which size should for sure match the given query in order to accomplish the minimum given threshold. by doing this many possible alternatives are discarded and thus fast search is possible.

Where is the code available?

the repository is hosted at github : <https://github.com/dav009/NERSimString>
The Benchmarking
We
Future work relating this can involved:

* adding support for more memoryMapped structures * building a bootstrapper (for learning NE) based on NERSimString * adding extra similarity Measures alternatives

*

Sections of possible document: Introduction Problem Description Solution (Description of SimString algorithm) Implementation Results Conclusions

[1] <http://www.chokkan.org/software/simstring/>

2 Implementation

3 Testing and Results

4 Conclusion and Future Work