Fast Search for NE
Basic Algorithms for Computational Linguistics Project

Sibel Ciddi[*] Noushin Fadaie[†] David Przybilla[‡]

July 11, 2012

1 Introduction

NERSimString

It 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 entitiy dictionaries and a fast implementation for corpora annotation 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. 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 two problems arise: *even in a restricted domain, the number of NE in a dictionary can be overwhelming. *the final goal is to annotate all the name entities given in a dictionary in a huge corpora.

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.

Why not exact lookups? Well, 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-Approxiamte String Retireval

What does NERSimString offer?

Well, it offers a quick way to make lookups on the cost of space.

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

[*]sciddi@coli.uni-saarland.de
[†]fadaei.noushin@gmail.com
[‡]dav.alejandro@gmail.com

*SuffixTree: it is a dictionary based on a suffixTree hold in memory *NaiveHashTable: it i a dictionary based on hashTabes hold in memory *MemoryMapped Hashtable: it is a dictionary based on hashmaps,it is meant to be for huge amoutns of data since it is stored dinamically in the disk in order to avoid being loaded completely into memory. *MemoryMapped SuffixTree: ToDO in a future version

The memory mapped hashtable comes with a memory cost, that is since the speed of a hashtable is to be kept but it will be stored in differnt files, then a lot of markers and a fix offset of bytes has to be used in order to assure what is in each slot, this means extra memory will be necessary when compared ot the other dictionaries implementations.

To developers NerSimString was implemented in such way so that adding a new dictionary implementation is ndependant of the simString algorithm, in such way new experiments and datastructures can be added.

What kind of similarities?

In order to provide approximate String retrieval the following similarity measures are supported: -Jaccard -Dice -Cosine

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