Next Improvement towards Linear Named Entity Recognition using Character Gazetteers

Nguyen G., Dlugolinsky S., Laclavik M., Seleng M., Tran V. Institute of Informatics, Slovak Academy of Sciences giang.ui@savba.sk
Content

- Context:
  - Big Data,
  - Natural Language Processing (NLP),
  - Named Entity Recognition (NER)
- Gazetteers
  - Tree structures: design and realizations HMT, CST, PHT
  - NER with linear matching complexity
  - Overlapping and embedded entities
- Evaluations
- Future work
Work context

- **Big Data** produced daily in
  - Social media: Twitter, Google+, Facebook, Instagram, etc.
  - Wikipedia, Wikia, newspapers ...
  - Other internal sources like transactions, logs, emails, ...

- **Knowledge** and **Information** hidden in (un|semi-)structured data
  - useful for
    - business or political sentiment analysis
    - public opinion assessment
    - emergency response, etc.
  - text, images, audio, video processing

- **Text ➔ NLP ➔ Information**
Natural Language Processing (NLP)

- Incoming text comes continuously from websites, portals, social media, etc.
- The need to recognize well-known NEs and theirs occurrences with references
- NER is important task in order to gain information
Gazetteer and NER (1)
Gazetteers and NER (2)

- Basic, independent and very effective NER technique for NE identification in text
- Processing approaches
  - Token-based: split input text into a sequence of tokens (words)
  - Character-based: processing input text character by character
- NE recognitions
  - Machine learning techniques
  - Finite-state machines (FSM)
Related work

- **Ontotext Hash Gazetteer**
  - Based on hash tables
  - Authors: “3x faster and 4x less memory than FSM equivalent”
  - As a part of the GATE only

- **Ontotext Stand-Alone Gazetteer**
  - Stand-alone version of the Hash Gazetteer
  - No longer available

- **Ontotext Large Knowledge Base Gazetteer**
  - Support for ontology-aware NLP
  - As a part of the GATE only

- Other gazetteers implemented as a proprietary look-up piece of code or complex solutions
Our requirements

- Standalone
  - no 3rd party libraries needed
  - does not rely on external preprocessing; e.g. tokenization
- Linear complexity lookup algorithm
  - fast and effective processing of input text as a stream, especially for Big Data
- Editable data structure
  - add/remove NEs between lookups
- Memory efficient data structure
  - “learn” tens of millions of entities
- Robust
  - input texts of any size
  - any language
Three realizations of character gazetteer

- **HMT**: Hash-map Multi-way Tree
  - each tree node is represented by one character
  - tree structure is implemented using Java HashMap with $O(1)$ in average for basic operations (get and put)

- **CST**: first-Child next-Sibling binary Tree
  - each tree node is represented by one character
  - pure and simple Java structure for tree nodes

- **PHT**: Patricia Hash-map Tree
  - tree nodes are collapsible and can contain more characters
  - tree structure is implemented using Java HashMap with $O(1)$ in average for basic operations (get and put)
Gazetteer tree structures (1)
HMT (left) and CST (right) realizations

Figure 1. Two different representations of the same character gazetteer tree: (left) multiway tree and (right) first child-next sibling binary tree
Gazetteer tree structures (2)
HMT (left); collapsible nodes (middle) and PHT (right)
HMT, CST and PHT gazetteers

All (HTM, CST and PHT):
- (+) Linear matching complexity
- (+) Precious, deals with embedded and overlapping cases

HMT: Hash-map Multi-way Tree
- (-) consumes a lot of memory i.e. nearly 3.7GB memory for 2.6 millions well-known entity tree structure
- (+) very fast

CST: first-Child next-Sibling binary Tree
- (+) memory efficiency: uses only 33% vs. HMT
- (-) 30 times slower vs. HMT for big data

PHT: Patricia Hash-map Tree
- (+) memory efficiency: only 45% vs. HMT
- (+) very fast, only 1.37 times slower vs. HMT for big data,
Embedded and overlapping entities (1)

- Complicated combinations of all possible entity occurrences
- Hard to find all embedded and overlapping entities in input text when the linear complexity is kept
Embedded and overlapping entities (2)

- Case 1, 4: classically
  - Matching by traveling character gazetteer tree
  - Choices: match all or match the first or match the longest
- Case 2, 3, 5: problem - where and how many times are entities overlapping or embedded
- “Carry back” solution:
  - Bring back a part of matched entity to input until the first boundary occurrence
  - Negligible increment of matching time due to the fact that entity length is usually short
Named entity recognition
Character-based with Linear matching complexity

Algorithm 2 Matching algorithm of the character gazetteer

1: \( buf \leftarrow \) empty
2: \( node \leftarrow \) root node
3: while characters on input do
4: \( ch \leftarrow \) next character from input
5: normalize whitespace or skip multiple
6: if \( node \) has a child node mapped on \( ch \) then
7: add \( ch \) to \( buf \)
8: \( node \leftarrow \) child node mapped on \( ch \)
9: if \( node \) is marked as a matching node then
10: found an entity
11: end if
12: else
13: if \( buf \) contains character other than letter or digit then
14: unread characters from \( buf \) back on input until
15: the first occurrence of the character that is not
16: letter or digit
17: end if
18: \( buf \leftarrow \) empty
19: \( node \leftarrow \) root node
20: end if
21: end while
Memory consumptions

Figure 2. Memory consumption of character gazetteer tree according to different tree representation and fill data
Evaluation results:
Linear matching times

![Graphs showing matching times for PHT/HMT≈1.37 and CST/HMT≈30 with increasing number of documents.]
Evaluation results (2)
Memory consumptions

Memory consumption
HMT vs. PHT vs. CST (1 > 0.45 > 0.33)

FreeBase person dataset
- HMT
- PHT
- CST
Evaluation datasets

- Gazetteer datasets:
  - Freebase persons: 2,614,401 unique entities
  - Other datasets are available and tested

- Incoming input data sets
  - 2586 documents, 5107 documents, 7564 documents and 9909 documents acquired from CoNLL-2003 datasets (Reuters’ text) with approximately 7 MB, 15 MB, 22 MB and 29 MB of pure text
  - Here is no restriction, input data sets can be of any size and in any languages
Bing Crosby was born in Tacoma, Washington, on May 3, 1903,[1] in a house his father built at 1112 North J Street.[12] In 1906, Crosby’s family moved to Spokane, Washington. [13] In 1913, Crosby’s father built a house at 508 E. Sharp Ave.[14] The house now sits on the campus of Bing’s alma mater Gonzaga University[15] and formerly housed the Alumni Association in U.S. Gonzaga. He was the fourth of seven children: brothers Larry (1895–1975), Everett (1896–1966), Ted (1900–1973), and Bob (1913–1993); and two sisters, Catherine (1904–1974) and Mary Rose (1906–1990). His parents were Harry Lillis Crosby, Sr. (1870–1950), a bookkeeper, and Catherine Helen (known as Kate) (née Harrigan; 1873–1964). Crosby’s mother was a second generation Irish-American.[16] His father was of English descent; some of his ancestors had emigrated to what would become the U.S. in the 17th century, and included Mayflower passenger William Brewster (c. 1567 – April 10, 1644).[17] In 1910, six-year-old Harry Crosby Jr. was forever renamed. The Sunday edition of the Spokesman-Review published a feature called "The Bingville Bugle".[18][19] Written by humorist Newton Newkirk, The Bingville Bugle was a parody of a hillbilly newsletter filled with gossipy tidbits, minstrel quips, creative spelling, and mock ads. A neighbor, 15-year-old Valentine Hobart, shared Crosby’s enthusiasm for "The Bugle" and noting Crosby’s laugh, took a liking to him and called him "Bingo from Bingville". Eventually the last vowel was dropped and the nickname stuck.[20] In 1917, Crosby took a summer job as property boy at Spokane's "Auditorium," where he witnessed some of the finest acts of the day, including Al Jolson, who held Crosby spellbound with his ad libbing and spoofs of Hawaiian songs. Crosby later described J...
Read more about our gazetteers


Thank you for attention

Viet Tran
viet.ui@savba.sk