RSS Join Engine

Presented by: MATTA Elie et al.
Introduction

What is RSS? Problem?

Background research

Theoretical part

Practical part

Adopted solution

Framework

Pseudo-code

Test hypothesis

Time of response

Precision

Conclusion
Introduction

- What is RSS?
- Problem statement
  - Waste of time for the users.
  - Too much information.
- RSS Join Engine
Background research

Divided into 2 parts: theatrical and practical.

Theoretical part:

Several types of relations between feeds:
- Disjointness
- Inclusion
- Intersection
- Equality
- Oppositiness

Semantic relatedness: 2 main concepts:

1. String, word and text similarity [1]:
   a. String similarity: Works on shape/syntax of the sentence.
   b. Word similarity: Word-to-word similarity metrics (Distance oriented measure, Knowledge based, corpus based).
   c. Text similarity: Similarity of the common words in a text placed in order.
2. RSS merger framework – 4 modules [2]:

- **Pre-processing**
- **Item Relatedness**
  - Element relatedness
    - Text relatedness
    - Vector space generator
- **Clustering**
- **Merger**
  - User's profile
  - User's-merging rule
Background research

Data streams management [3].

Sliding-window concept: 2 types of windows:
1. Count-based: Contains the last T items.
2. Time-based: Contains the items that arrived in the last t time units.

Possible strategies:
1. Eager re-evaluation: Generates new results after each new tuple.
2. Lazy re-evaluation: Re-executes the query periodically.

Testing tuples: \( \forall u \in S1 \text{ and } k.ts - T1 \leq u.ts \leq k.ts \)
Background research

Practical part

- XML comparators (BeyondCompare, ExamDiff…)
- RSS Aggregators (Feed reader, RSS bandit…) [4]
- RSS Merger [2]
  a. Measures relatedness between news items (+stemming and generating vectors)
  b. Clusters the RSS items based on the relatedness.
  c. Merges the news based on the users rules.
Adopted solution

RSS Join engine based on the XML comparators technique, RSS aggregators and RSS merger:

Figure 1 – RSS Join Engine Framework
function GetRSS()
    Check(URL)
    if(Check)
        Connect(URL)
        Collect(RSS)
    else
        “Display Error Message”
    end If
end function

function Comp(title1,title2)
    Open an instance on WorldNet Knowledge Base
    Compare each title with the other one using semantic relatedness measures (xSim,...)
    return the Comparison as type (intersection, disjointness or equality)
end function
function JoinRSS()
    Comp(title1, title2)
    If Comp = Disjointness then
        Show both titles
    end If
    If Comp = Equality then
        Show one of the titles
    end If
    If Comp = Intersection then
        If one of the titles is totally included in another (Inclusion) then
            Show the title including the other title
        else If one of the titles is intersecting with another but the content is referring to opposite meaning (Oppositeness) then
            Show both titles
        else
            Show the intersection of one of the titles
        end If
    end If
Test hypothesis

Time of response chart

Figure 2 - Time of response chart
Test hypothesis

Figure 3 – Precision chart

Copyright © 2010-2011, eliematta.com. All rights reserved
Conclusion

We calculated the semantic relatedness between RSS to obtain one of the five item relations.

Figure 2 shows that no matter what was the threshold value, the time of response is identical which is abnormal because it should vary with the threshold value proportionally.

Figure 3 shows a big success for the precision, that’s because while using per example 0.9 as threshold value we obtained ~100% as result.

As a closure for this study, we still need to find a solution to reduce the time of response in order to make it acceptable w.r.t human scale, and extend the join process to cover the description and other elements of the RSS feeds; these ideas and issues will be discussed in our next paper.
References


Any Questions