CSE 574 Project Proposal: Automatically detecting conflicting statements using TextRunner

Alan Ritter[*]

January 29, 2008

1 Introduction

1.1 Problem

I would like to use TextRunner to automatically detect conflicting statements on the web. Although there are many types of conflicts, initially the most promising approach seems to be detecting conflicts between functional relations. I have done some initial investigation into the problem, and an initial set of conflicts found using TextRunner for the *Invented* relation can be found here. For the class project, I would like to extend this approach to discover conflicting statements between arbitrary functional relations, instead of just a few hand-selected ones.

1.2 Motivation

Investigating how to effectively discover conflicting statements in a set of noisy facts extracted from unstructured text could viewed as an initial step towards building an automatic "fact checker" similar to a spelling or grammar checker. If we can achieve high precision and recall, in addition to constructing a reliable knowledge base of generally true facts, then we can automatically evaluate facts extracted from sources such as e-mail, blogs, documents, ...

1.3 Functional Relations

Functional relations are those where one argument to the relation is a function of the other. *Invented*(**person**, **thing**) is an example of a functional relation because for a given **thing**, it can only have been invented by one **person**[1] For

[*] A pdf version is available here

[1] There are of course exceptions, for example Calculus was independently "invented" by both Newton and Leibniz

Emoticons were invented by Scott Fahlman on the CMU bulletin board system
Vladimir Nabokov invented emoticons!

Table 1: Two conflicting sentences found for the functional relation, *Invented*

Wikipedia:	André Gide	*death_place*	Paris
TextRunner	André Gide	*died in*	Paris

Table 2: Matching Wikipedia infobox attributes to TextRunner predicates

Invented, the **person** argument can be thought of as a *key* for the relation. As an example, consider the two conflicting sentences in table 1 which were automatically detected as a potential conflict using TextRunner.

2 Milestones / Decomposition

The project will proceed in 4 steps. Each is described below:

2.1 Week 1: Generate a large database of functional relations

In order to discover conflicts between arbitrary functional relations, we need to know which relations are functional. Many hand-built ontologies explicitly label functional relations. In addition, most Wikipedia infobox attributes are functional (we can easily weed out attributes which have more than one value per article). For the project, I am planning on investigating the use of functional relations from infoboxes, and ResearchCyc.

2.2 Weeks 2-3: Map functional relations to TextRunner predicates (Milestone 1)

The names of attributes used in Cyc and infoboxes are in a very different form than those used by TextRunner. For example the infobox class *Writer* contains an attribute *death_place*, which does not match the TextRunner predicate *died in*.

To solve this problem I plan to query TextRunner with instances of the class. As an example consider the *death_place* infobox attribute from the article on André Gide and the associated TextRunner triple which expresses the same information (Table 2). This example suggests that by querying TextRunner with instances of the functional attributes from Wikipedia infoboxes, we should be able to map these non-textual attribute names to the more textual form of TextRunner's predicates.

2.3 Weeks 4-5: Discover conflicting statements between functional relations (Milestone 2)

Once we have a large database of functional relations which have been mapped to TextRunner predicates, we can simply collect all triples from TextRunner which match these predicates, group them by their key argument (for example group all triples of the form (André Gide, died in, X) together to build groups of potential conflicts.

Some work will need to be done to filter out synonyms, anaphora, and extractor errors. I will likely only have time to do some simple heuristic-based filtering within the scope of the course project, however, future work could use a self-supervised machine learning approach. Because conflicts found between high quality reliable sources, such as Wikipedia articles or news text are likely to be erroneous, they could be used as negative training examples. Similarly, conflicts between more opinionated sources such as blogs could be used as positive examples.

2.4 Week 6: Build a web interface to display the results

To display results I will build an index of conflicting statements which is queryable by key attribute, non-key or both. Results will be displayed in a similar manner to the prototype for *Invented.*

3 Evaluation

Recall will be difficult to measure directly, since it's imposible to know exactly how many sentences in TextRunner's corpus conflict with eachother. Instead I will try to estimate the total number of true conflicts discovered. To estimate precision in addition to the total number of conflicts, I will hand-label a random sample of the output of the system.

I will hand-label a selection of the functional TextRunner predicates which were mapped from the infoboxes and Cyc, and I will try to evaluate whether Cyc or the infoboxes were a better source of functional relations. I will also plot histograms of the frequency of the predicates in TextRunner's corpus in addition to the number of conflicts found by each predicate and their accuracy.