It's a Contradiction—No, it's Not: A Case Study using Functional Relations

Alan Ritter, Doug Downey, Stephen Soderland and Oren Etzioni
Turing Center
Department of Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98195, USA
{aritter,ddowney,soderlan,etzioni}@cs.washington.edu

Abstract

Contradiction Detection (CD) in text is a difficult NLP task. We investigate CD over functions (*e.g.*, BornIn(Person)=Place), and present a domain-independent algorithm that automatically discovers phrases denoting functions with high precision. Previous work on CD has investigated hand-chosen sentence pairs. In contrast, we automatically harvested from the Web pairs of sentences that *appear* contradictory, but were surprised to find that most pairs are in fact consistent. For example, "Mozart was born in Salzburg" does *not* contradict "Mozart was born in Austria" despite the functional nature of the phrase "was born in". We show that background knowledge about meronyms (*e.g.*, Salzburg is in Austria), synonyms, functions, and more is essential for success in the CD task.

1 Introduction and Motivation

Detecting contradictory statements is an important and challenging NLP task with a wide range of potential applications including analysis of political discourse, of scientific literature, and more (de Marneffe et al., 2008; Condoravdi et al., 2003; Harabagiu et al., 2006). De Marneffe *et al.* present a model of CD that defines the task, analyzes different types of contradictions, and reports on a CD system. They report 23% precision and 19% recall at detecting contradictions in the RTE-3 data set (Voorhees, 2008). Although RTE-3 contains a wide variety of contradictions, it does not reflect the prevalence of *seeming* contradictions and the paucity of *genuine* contradictions, which we have found in our corpus.

1.1 Contradictions and World Knowledge

Our paper is motivated in part by de Marneffe *et al.*'s work, but with some important differences. First, we introduce a simple logical foundation for the CD task, which suggests that extensive world knowledge is essential for building a domain-independent CD system. Second, we automatically generate a large corpus of *apparent* contradictions found in arbitrary Web text. We show that most of these apparent contradictions are actually consistent statements due to meronyms (Alan Turing was born in London *and* in England), synonyms (George Bush is married to *both* Mrs. Bush and Laura Bush), hypernyms (Mozart died of *both* renal failure and kidney disease), and reference ambiguity (one John Smith was born in 1997 and a *different* John Smith in 1883). Next, we show how background knowledge enables a CD system to discard seeming contradictions and focus on genuine ones.

De Marneffe *et al.* introduced a typology of contradiction in text, but focused primarily on contradictions that can be detected from linguistic evidence (*e.g.* negation, antonymy, and structural or lexical disagreements). We extend their analysis to a class of contradictions that can only be detected utilizing background knowledge. Consider for example the following sentences:

1) "Mozart was born in Salzburg."
2) "Mozart was born in Vienna."
3) "Mozart visited Salzburg."
4) "Mozart visited Vienna."

Sentences 1 & 2 are contradictory, but 3 & 4 are not. Why is that? The distinction is not syntactic. Rather, sentences 1 and 2 are contradictory because

the relation expressed by the phrase "was born in" can be characterized here as a *function* from people's names to their unique birthplaces. In contrast, "visited" does not denote a functional relation.[1]

We cannot assume that a CD system knows, in advance, all the functional relations that might appear in a corpus. Thus, a central challenge for a function-based CD system is to determine which relations are functional based on a corpus. Intuitively, we might expect that "functional phrases" such as "was born in" would typically map person names to unique place names, making function detection easy. But, in fact, function detection is surprisingly difficult because name ambiguity (*e.g.*, John Smith), common nouns (*e.g.*, "dad" or "mom"), definite descriptions (*e.g.*, "the president"), and other linguistic phenomena can mask functions in text. For example, the two sentences "John Smith was born in 1997." and "John Smith was born in 1883." can be viewed as either evidence that "was born in" does not denote a function or, alternatively, that "John Smith" is ambiguous.

1.2 A CD System Based on Functions

We report on the AUCONTRAIRE CD system, which addresses each of the above challenges. First, AUCONTRAIRE identifies "functional phrases" statistically (Section 3). Second, AUCONTRAIRE uses these phrases to automatically create a large corpus of apparent contradictions (Section 4.2). Finally, AUCONTRAIRE sifts through this corpus to find genuine contradictions using knowledge about synonymy, meronymy, argument types, and ambiguity (Section 4.3).

Instead of analyzing sentences directly, AUCONTRAIRE relies on the TEXTRUNNER Open Information Extraction system (Banko et al., 2007; Banko and Etzioni, 2008) to map each sentence to one or more tuples that represent the entities in the sentences and the relationships between them (*e.g.*, *was_born_in(Mozart,Salzburg)*). Using extracted tuples greatly simplifies the CD task, because numerous syntactic problems (*e.g.*, anaphora, relative clauses) and semantic challenges (*e.g.*, quantification, counterfactuals, temporal qualification) are delegated to TEXTRUNNER or simply ignored. Nevertheless, extracted tuples are a convenient approximation of sentence content, which enables us to focus on function detection and function-based CD.

Our contributions are the following:

- We present a novel model of the Contradiction Detection (CD) task, which offers a simple logical foundation for the task and emphasizes the central role of background knowledge.
- We introduce and evaluate a new EM-style algorithm for detecting whether phrases denote functional relations and whether nouns (*e.g.*, "dad") are ambiguous, which enables a CD system to identify functions in arbitrary domains.
- We automatically generate a corpus of seeming contradictions from Web text, and report on a set of experiments over this corpus, which provide a baseline for future work on statistical function identification and CD. [2]

2 A Logical Foundation for CD

On what basis can a CD system conclude that two statements T and H are contradictory? Logically, contradiction holds when $T \models \neg H$. As de Marneffe *et al.* point out, this occurs when T and H contain antonyms, negation, or other lexical elements that suggest that T and H are directly contradictory. But other types of contradictions can only be detected with the help of a body of background knowledge K: In these cases, T and H *alone* are mutually consistent. That is,

$$T \not\models \neg H \wedge H \not\models \neg T$$

A contradiction between T and H arises only in the context of K. That is:

$$((K \wedge T) \models \neg H) \vee ((K \wedge H) \models \neg T)$$

Consider the example of Mozart's birthplace in the introduction. To detect a contradiction, a CD system must know that A) "Mozart" refers to the same entity in both sentences, that B) "was born in" denotes a functional relation, and that C) Vienna and Salzburg are inconsistent locations.

[1] Although we focus on function-based CD in our case study, we believe that our observations apply to other types of CD as well.

[2] The corpus is available at `http://www.cs.washington.edu/research/aucontraire/`

Of course, world knowledge, and reasoning about text, are often uncertain, which leads us to associate probabilities with a CD system's conclusions. Nevertheless, the knowledge base K is essential for CD.

We now turn to a probabilistic model that helps us simultaneously estimate the functionality of relations (B in the above example) and ambiguity of argument values (A above). Section 4 describes the remaining components of AUCONTRAIRE.

3 Detecting Functionality and Ambiguity

This section introduces a formal model for computing the probability that a phrase denotes a function based on a set of extracted tuples. An extracted tuple takes the form $R(x, y)$ where (roughly) x is the subject of a sentence, y is the object, and R is a phrase denoting the relationship between them. If the relation denoted by R is functional, then typically the object y is a function of the subject x. Thus, our discussion focuses on this possibility, though the analysis is easily extended to the symmetric case.

Logically, a relation R is functional in a variable x if it maps it to a unique variable y: $\forall x, y_1, y_2 \ R(x, y_1) \wedge R(x, y_2) \Rightarrow y_1 = y_2$. Thus, given a large random sample of ground instances of R, we could detect with high confidence whether R is functional. In text, the situation is far more complex due to ambiguity, polysemy, synonymy, and other linguistic phenomena. Deciding whether R is functional becomes a probabilistic assessment based on aggregated textual evidence.

The main evidence that a relation $R(x, y)$ is functional comes from the distribution of y values for a given x value. If R denotes a function and x is unambiguous, then we expect the extractions to be predominantly a single y value, with a few outliers due to noise. We aggregate the evidence that R is *locally* functional for a particular x value to assess whether R is *globally* functional for all x.

We refer to a set of extractions with the same relation R and argument x as a *contradiction set* $R(x, \cdot)$. Figure 1 shows three example contradiction sets. Each example illustrates a situation commonly found in our data. Example A in Figure 1 shows strong evidence for a functional relation. 66 out of 70 TEXTRUNNER extractions for *was_born_in (Mozart, PLACE)* have the same y value. An ambiguous x argument, however, can make a functional relation *appear* non-functional. Example B depicts a distribution of y values that appears less functional due to the fact that "John Adams" refers to multiple, distinct real-world individuals with that name. Finally, example C exhibits evidence for a non-functional relation.

A. was_born_in(Mozart, PLACE):
Salzburg(66), Germany(3), Vienna(1)

B. was_born_in(John Adams, PLACE):
Braintree(12), Quincy(10), Worcester(8)

C. lived_in(Mozart, PLACE):
Vienna(20), Prague(13), Salzburg(5)

Figure 1: Functional relations such as example A have a different distribution of y values than non-functional relations such as C. However, an ambiguous x argument as in B, can make a functional relation *appear* non-functional.

3.1 Formal Model of Functions in Text

To decide whether R is functional in x for all x, we first consider how to detect whether R is *locally functional* for a particular value of x. The local functionality of R with respect to x is the probability that R is functional estimated solely on evidence from the distribution of y values in a contradiction set $R(x, \cdot)$.

To decide the probability that R is a function, we define global functionality as the average local functionality score for each x, weighted by the probability that x is unambiguous. Below, we outline an EM-style algorithm that alternately estimates the probability that R is functional and the probability that x is ambiguous.

Let R^*_x indicate the event that the relation R is locally functional for the argument x, and that x is locally unambiguous for R. Also, let D indicate the set of observed tuples, and define $D_{R(x,\cdot)}$ as the multi-set containing the frequencies for extractions of the form $R(x, \cdot)$. For example the distribution of extractions from Figure 1 for example A is

$D_{\text{was\_born\_in}(\text{Mozart},\cdot)} = \{66, 3, 1\}$.

Let θ^f_R be the probability that $R(x, \cdot)$ is locally functional for a random x, and let Θ^f be the vector of these parameters across all relations R. Likewise, θ^u_x represents the probability that x is locally unambiguous for random R, and Θ^u the vector for all x.

We wish to determine the *maximum a posteriori* (MAP) functionality and ambiguity parameters given the observed data D, that is $\arg\max_{\Theta^f,\Theta^u} P(\Theta^f,\Theta^u|D)$. By Bayes Rule:

$$P(\Theta^f,\Theta^u|D) = \frac{P(D|\Theta^f,\Theta^u)P(\Theta^f,\Theta^u)}{P(D)} \quad (1)$$

We outline a generative model for the data, $P(D|\Theta^f,\Theta^u)$. Let us assume that the event R_x^* depends only on θ_R^f and θ_x^u, and further assume that given these two parameters, local ambiguity and local functionality are conditionally independent. We obtain the following expression for the probability of R_x^* given the parameters:

$$P(R_x^*|\Theta^f,\Theta^u) = \theta_R^f\theta_x^u$$

We assume each set of data $D_{R(x,\cdot)}$ is generated independently of all other data and parameters, given R_x^*. From this and the above we have:

$$P(D|\Theta^f,\Theta^u) = \prod_{R,x}\Big(P(D_{R(x,\cdot)}|R_x^*)\theta_R^f\theta_x^u + P(D_{R(x,\cdot)}|\neg R_x^*)(1-\theta_R^f\theta_x^u)\Big) \quad (2)$$

These independence assumptions allow us to express $P(D|\Theta^f,\Theta^u)$ in terms of distributions over $D_{R(x,\cdot)}$ given whether or not R_x^* holds. We use the URNS model as described in (Downey et al., 2005) to estimate these probabilities based on binomial distributions. In the single-urn URNS model that we utilize, the extraction process is modeled as draws of labeled balls from an urn, where the labels are either correct extractions or errors, and different labels can be repeated on varying numbers of balls in the urn.

Let $k = \max D_{R(x,\cdot)}$, and let $n = \sum D_{R(x,\cdot)}$; we will approximate the distribution over $D_{R(x,\cdot)}$ in terms of k and n. If $R(x,\cdot)$ is locally functional and unambiguous, there is exactly one correct extraction label in the urn (potentially repeated multiple times). Because the probability of correctness tends to increase with extraction frequency, we make the simplifying assumption that the most frequently extracted element is correct.[3] In this case, k is the number of correct extractions, which by the URNS model has a binomial distribution with parameters n and p, where p is the precision of the extraction process. If $R(x,\cdot)$ is *not* locally functional and unambiguous, then we expect k to typically take on smaller values. Empirically, the underlying frequency of the most frequent element in the $\neg R_x^*$ case tends to follow a Beta distribution.

Under the model, the probability of the evidence given R_x^* is:

$$P(D_{R(x,\cdot)}|R_x^*) \approx P(k,n|R_x^*) = \binom{n}{k}p^k(1-p)^{n-k}$$

And the probability of the evidence given $\neg R_x^*$ is:

$$\begin{aligned} P(D_{R(x,\cdot)}|\neg R_x^*) &\approx P(k,n|\neg R_x^*) \\ &= \tbinom{n}{k}\int_0^1 \frac{p'^{k+\alpha_f-1}(1-p')^{n+\beta_f-1-k}}{B(\alpha_f,\beta_f)}dp' \\ &= \frac{\binom{n}{k}\Gamma(n-k+\beta_f)\Gamma(\alpha_f+k)}{B(\alpha_f,\beta_f)\Gamma(\alpha_f+\beta_f+n)} \end{aligned} \quad (3)$$

where n is the sum over $D_{R(x,\cdot)}$, Γ is the Gamma function and B is the Beta function. α_f and β_f are the parameters of the Beta distribution for the $\neg R_x^*$ case. These parameters and the prior distributions are estimated empirically, based on a sample of the data set of relations described in Section 5.1.

3.2 Estimating Functionality and Ambiguity

Substituting Equation 3 into Equation 2 and applying an appropriate prior gives the probability of parameters Θ^f and Θ^u given the observed data D. However, Equation 2 contains a large product of sums—with two independent vectors of coefficients, Θ^f and Θ^u—making it difficult to optimize analytically.

If we knew which arguments were ambiguous, we would ignore them in computing the functionality of a relation. Likewise, if we knew which relations were non-functional, we would ignore them in computing the ambiguity of an argument. Instead, we initialize the Θ^f and Θ^u arrays randomly, and then execute an algorithm similar to Expectation-Maximization (EM) (Dempster et al., 1977) to arrive at a high-probability setting of the parameters.

Note that if Θ^u is fixed, we can compute the expected fraction of locally unambiguous arguments x for which R is locally functional, using $D_{R(x',\cdot)}$ and

[3] As this assumption is invalid when there is not a unique maximal element, we default to the prior $P(R_x^*)$ in that case.

Equation 3. Likewise, for fixed Θ^f, for any given x we can compute the expected fraction of locally functional relations R that are locally unambiguous for x.

Specifically, we repeat until convergence:

1. Set $\theta^f_R = \frac{1}{s_R} \sum_x P(R^*_x | D_{R(x,\cdot)}) \theta^u_x$ for all R.
2. Set $\theta^u_x = \frac{1}{s_x} \sum_R P(R^*_x | D_{R(x,\cdot)}) \theta^f_R$ for all x.

In both steps above, the sums are taken over only those x or R for which $D_{R(x,\cdot)}$ is non-empty. Also, the normalizer $s_R = \sum_x \theta^u_x$ and likewise $s_x = \sum_R \theta^f_R$.

As in standard EM, we iteratively update our parameter values based on an expectation computed over the unknown variables. However, we alternately optimize two disjoint sets of parameters (the functionality and ambiguity parameters), rather than just a single set of parameters as in standard EM. Investigating the optimality guarantees and convergence properties of our algorithm is an item of future work.

By iteratively setting the parameters to the expectations in steps 1 and 2, we arrive at a good setting of the parameters. Section 5.2 reports on the performance of this algorithm in practice.

4 System Overview

AUCONTRAIRE identifies phrases denoting functional relations and utilizes these to find contradictory assertions in a massive, open-domain corpus of text.

AUCONTRAIRE begins by finding extractions of the form $R(x, y)$, and identifies a set of relations R that have a high probability of being functional. Next, AUCONTRAIRE identifies contradiction sets of the form $R(x, \cdot)$. In practice, most contradiction sets turned out to consist overwhelmingly of seeming contradictions—assertions that do not actually contradict each other for a variety of reasons that we enumerate in section 4.3. Thus, a major challenge for AUCONTRAIRE is to tease apart which pairs of assertions in $R(x, \cdot)$ represent genuine contradictions.

Here are the main components of AUCONTRAIRE as illustrated in Figure 2:

Extractor: Create a set of extracted assertions $\mathcal{E}$ from a large corpus of Web pages or other documents. Each extraction $R(x, y)$ has a probability p of being correct.

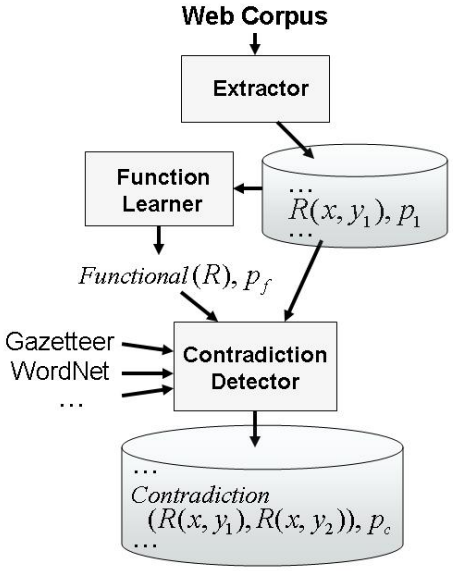

Figure 2: AUCONTRAIRE architecture

Function Learner: Discover a set of functional relations $\mathcal{F}$ from among the relations in $\mathcal{E}$. Assign to each relation in $\mathcal{F}$ a probability p_f that it is functional.

Contradiction Detector: Query $\mathcal{E}$ for assertions with a relation R in $\mathcal{F}$, and identify sets $\mathcal{C}$ of potentially contradictory assertions. Filter out seeming contradictions in $\mathcal{C}$ by reasoning about synonymy, meronymy, argument types, and argument ambiguity. Assign to each potential contradiction a probability p_c that it is a genuine contradiction.

4.1 Extracting Factual Assertions

AUCONTRAIRE needs to explore a large set of factual assertions, since genuine contradictions are quite rare (see Section 5). We used a set of extractions $\mathcal{E}$ from the Open Information Extraction system, TEXTRUNNER (Banko et al., 2007), which was run on a set of 117 million Web pages.

TEXTRUNNER does not require a pre-defined set of relations, but instead uses shallow linguistic analysis and a domain-independent model to identify phrases from the text that serve as relations and phrases that serve as arguments to that relation. TEXTRUNNER creates a set of extractions in a single pass over the Web page collection and provides an index to query the vast set of extractions.

Although its extractions are noisy, TEXTRUNNER provides a probability that the extractions are cor-

rect, based in part on corroboration of facts from different Web pages (Downey et al., 2005).

4.2 Finding Potential Contradictions

The next step of AUCONTRAIRE is to find contradiction sets in $\mathcal{E}$.

We used the methods described in Section 3 to estimate the functionality of the most frequent relations in $\mathcal{E}$. For each relation R that AUCONTRAIRE has judged to be functional, we identify contradiction sets $R(x, \cdot)$, where a relation R and domain argument x have multiple range arguments y.

4.3 Handling Seeming Contradictions

For a variety of reasons, a pair of extractions $R(x, y_1)$ and $R(x, y_2)$ may not be actually contradictory. The following is a list of the major sources of false positives—pairs of extractions that are not genuine contradictions, and how they are handled by AUCONTRAIRE. The features indicative of each condition are combined using Logistic Regression, in order to estimate the probability that a given pair, $\{R(x, y_1), R(x, y_2)\}$ is a genuine contradiction.

Synonyms: The set of potential contradictions *died_from(Mozart,·)* may contain assertions that Mozart died from *renal failure* and that he died from *kidney failure*. These are distinct values of y, but do not contradict each other, as the two terms are synonyms. AUCONTRAIRE uses a variety of knowledge sources to handle synonyms. WordNet is a reliable source of synonyms, particularly for common nouns, but has limited recall. AUCONTRAIRE also utilizes synonyms generated by RESOLVER (Yates and Etzioni, 2007)— a system that identifies synonyms from TEXTRUNNER extractions. Additionally, AUCONTRAIRE uses edit-distance and token-based string similarity (Cohen et al., 2003) between apparently contradictory values of y to identify synonyms.

Meronyms: For some relations, there is no contradiction when y_1 and y_2 share a meronym, *i.e.* "part of" relation. For example, in the set *born_in(Mozart,·)* there is no contradiction between the y values "Salzburg" and "Austria", but "Salzburg" conflicts with "Vienna". Although this is only true in cases where y occurs in an upward monotone context (MacCartney and Manning, 2007), in practice genuine contradictions between y-values sharing a meronym relationship are extremely rare. We therefore simply assigned contradictions between meronyms a probability close to zero. We used the Tipster Gazetteer[4] and WordNet to identify meronyms, both of which have high precision but low coverage.

Argument Typing: Two y values are not contradictory if they are of different argument types. For example, the relation *born_in* can take a date or a location for the y value. While a person can be born in only one year and in only one city, a person can be born in both a year and a city. To avoid such false positives, AUCONTRAIRE uses a simple named-entity tagger[5] in combination with large dictionaries of person and location names to assign high-level types (person, location, date, other) to each argument. AUCONTRAIRE filters out extractions from a contradiction set that do not have matching argument types.

Ambiguity: As pointed out in Section 3, false contradictions arise when a single x value refers to multiple real-world entities. For example, if the contradiction set *born_in(John Sutherland, ·)* includes birth years of both 1827 and 1878, is one of these a mistake, or do we have a grandfather and grandson with the same name? AUCONTRAIRE computes the probability that an x value is unambiguous as part of its Function Learner (see Section 3). An x value can be identified as ambiguous if its distribution of y values is non-functional for multiple functional relations.

If a pair of extractions, $\{R(x, y_1), R(x, y_2)\}$, does not fall into any of the above categories and R is functional, then it is likely that the sentences underlying the extractions are indeed contradictory. We combined the various knowledge sources described above using Logistic Regression, and used 10-fold cross-validation to automatically tune the weights associated with each knowledge source. In addition, the learning algorithm also utilizes the following features:

- Global functionality of the relation, θ_R^f
- Global unambiguity of x, θ_x^u

[4] http://crl.nmsu.edu/cgi-bin/Tools/CLR/clrcat

[5] http://search.cpan.org/~simon/Lingua-EN-NamedEntity-1.1/NamedEntity.pm

- Local functionality of $R(x, \cdot)$
- String similarity (a combination of token-based similarity and edit-distance) between y_1 and y_2
- The argument types (person, location, date, or other)

The learned model is then used to estimate how likely a potential contradiction $\{R(x, y_1), R(x, y_2)\}$ is to be genuine.

5 Experimental Results

We evaluated several aspects of AUCONTRAIRE: its ability to detect functional relations and to detect ambiguous arguments (Section 5.2); its precision and recall in contradiction detection (Section 5.3); and the contribution of AUCONTRAIRE's key knowledge sources (Section 5.4).

5.1 Data Set

To evaluate AUCONTRAIRE we used TEXTRUNNER's extractions from a corpus of 117 million Web pages. We restricted our data set to the 1,000 most frequent relations, in part to keep the experiments tractable and also to ensure sufficient statistical support for identifying functional relations.

We labeled each relation as functional or not, and computed an estimate of the probability it is functional as described in section 3.2. Section 5.2 presents the results of the Function Learner on this set of relations. We took the top 2% (20 relations) as $\mathcal{F}$, the set of functional relations in our experiments. Out of these, 75% are indeed functional. Some examples include: *was born in*, *died in*, and *was founded by*.

There were 1.2 million extractions for all thousand relations, and about 20,000 extractions in 6,000 contradiction sets for all relations in $\mathcal{F}$.

We hand-tagged 10% of the contradiction sets $R(x, \cdot)$ where $R \in \mathcal{F}$, discarding any sets with over 20 distinct y values since the x argument for that set is almost certainly ambiguous. This resulted in a data set of 567 contradiction sets containing a total of 2,564 extractions and 8,844 potentially contradictory pairs of extractions.

We labeled each of these 8,844 pairs as contradictory or not. In each case, we inspected the original sentences, and if the distinction was unclear, consulted the original source Web pages, Wikipedia articles, and Web search engine results.

In our data set, genuine contradictions over functional relations are surprisingly rare. We found only 110 genuine contradictions in the hand-tagged sample, only 1.2% of the potential contradiction pairs.

5.2 Detecting Functionality and Ambiguity

We ran AUCONTRAIRE's EM algorithm on the thousand most frequent relations. Performance converged after 5 iterations resulting in estimates of the probability that each relation is functional and each x argument is unambiguous. We used these probabilities to generate the precision-recall curves shown in Figure 3.

The graph on the left shows results for functionality, while the graph on the right shows precision at finding unambiguous arguments. The solid lines are results after 5 iterations of EM, and the dashed lines are from computing functionality or ambiguity without EM (*i.e.* assuming uniform values of Θ^c when computing Θ^f and vice versa). The EM algorithm improved results for both functionality and ambiguity, increasing area under curve (AUC) by 19% for functionality and by 31% for ambiguity.

Of course, the ultimate test of how well AUCONTRAIRE can identify functional relations is how well the Contradiction Detector performs on automatically identified functional relations.

5.3 Detecting Contradictions

We conducted experiments to evaluate how well AUCONTRAIRE distinguishes genuine contradictions from false positives.

The bold line in Figure 4 depicts AUCONTRAIRE performance on the distribution of contradictions and seeming contradictions found in actual Web data. The dashed line shows the performance of AUCONTRAIRE on an artificially "balanced" data set that we constructed to contain 50% genuine contradictions and 50% seeming ones.

Previous research in CD presented results on manually selected data sets with a relatively balanced mix of positive and negative instances. As Figure 4 suggests, this is a much easier problem than CD "in the wild". The data gathered from the Web is badly skewed, containing only 1.2% genuine contradictions.

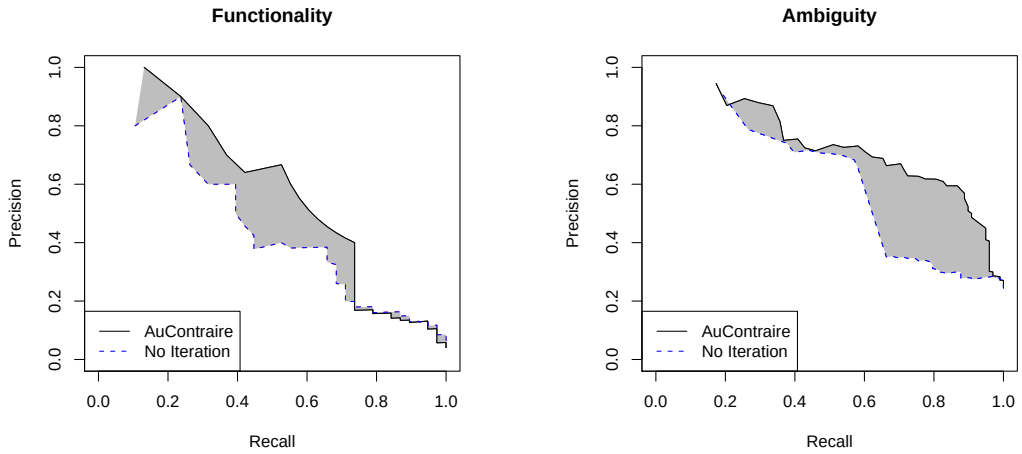

Figure 3: After 5 iterations of EM, AUCONTRAIRE achieves a 19% boost to area under the precision-recall curve (AUC) for functionality detection, and a 31% boost to AUC for ambiguity detection.

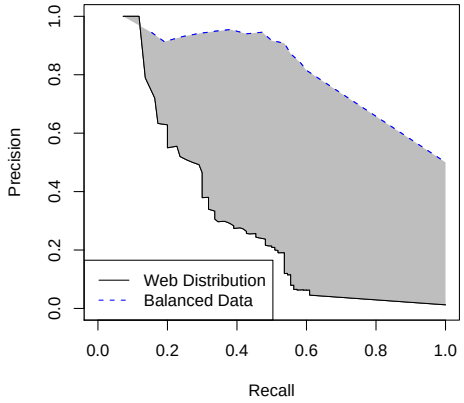

Figure 4: Performance of AUCONTRAIRE at distinguishing genuine contradictions from false positives. The bold line is results on the actual distribution of data from the Web. The dashed line is from a data set constructed to have 50% positive and 50% negative instances.

5.4 Contribution of Knowledge Sources

We carried out an ablation study to quantify how much each knowledge source contributes to AUCONTRAIRE's performance. Since most of the knowledge sources do not apply to numeric argument values, we excluded the extractions where y is a number in this study. As shown in Figure 5, performance of AUCONTRAIRE degrades with no knowledge of synonyms (NS), with no knowledge of meronyms (NM), and especially without argument typing (NT). Conversely, improvements to any of these three components would likely improve the performance of AUCONTRAIRE.

The relatively small drop in performance from no meronyms does not indicate that meronyms are not essential to our task, only that our knowledge sources for meronyms were not as useful as we hoped. The Tipster Gazetteer has surprisingly low coverage for our data set. It contains only 41% of the y values that are locations. Many of these are matches on a different location with the same name, which results in incorrect meronym information. We estimate that a gazetteer with complete coverage would *increase* area under the curve by approximately 40% compared to a system with meronyms from the Tipster Gazetteer and WordNet.

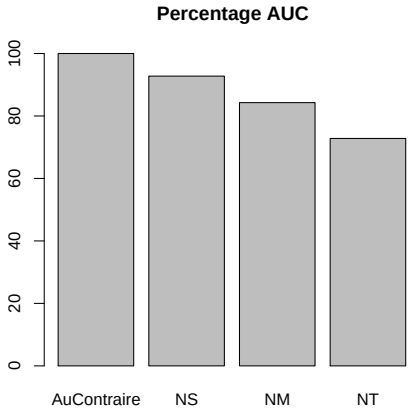

Figure 5: Area under the precision-recall curve for the full AUCONTRAIRE and for AUCONTRAIRE with knowledge removed. NS has no synonym knowledge; NM has no meronym knowledge; NT has no argument typing.

To analyze the errors made by AUCONTRAIRE, we hand-labeled all false-positives at the point of maximum F-score: 29% Recall and 48% Precision.

Figure 6 reveals the central importance of world knowledge for the CD task. About half of the errors (49%) are due to ambiguous x-arguments, which we found to be one of the most persistent obstacles to discovering genuine contradictions. A sizable portion is due to missing meronyms (34%) and missing synonyms (14%), suggesting that lexical resources with broader coverage than WordNet and the Tipster Gazetteer would substantially improve performance. Surprisingly, only 3% are due to errors in the extraction process.

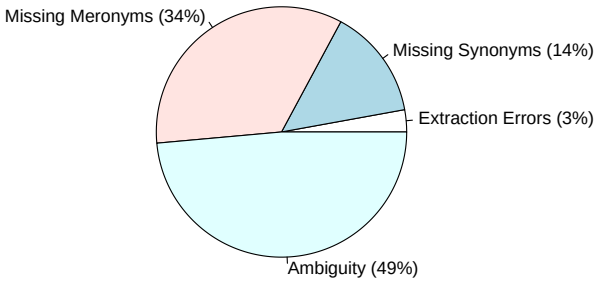

Figure 6: Sources of errors in contradiction detection.

All of our experimental results are based on the automatically discovered set of functions $\mathcal{F}$. We would expect AUCONTRAIRE's performance to improve substantially if it were given a large set of functional relations as input.

6 Related Work

Condoravdi *et al.* (2003) first proposed contradiction detection as an important NLP task, and Harabagiu *et al.* (2006) were the first to report results on contradiction detection using negation, although their evaluation corpus was a balanced data set built by manually negating entailments in a data set from the Recognizing Textual Entailment conferences (RTE) (Dagan et al., 2005). De Marneffe *et al.* (2008) reported experimental results on a contradiction corpus created by annotating the RTE data sets.

RTE-3 included an optional task, requiring systems to make a 3-way distinction: {entails, contradicts, neither} (Voorhees, 2008). The average performance for contradictions on the RTE-3 was precision 0.11 at recall 0.12, and the best system had precision 0.23 at recall 0.19. We did not run AUCONTRAIRE on the RTE data sets because they contained relatively few of the "functional contradictions" that AUCONTRAIRE tackles. On our Web-based data sets, we achieved a precision of 0.62 at recall 0.19, and precision 0.92 at recall 0.51 on the balanced data set. Of course, comparisons across very different data sets are not meaningful, but merely serve to underscore the difficulty of the CD task.

In contrast to previous work, AUCONTRAIRE is the first to do CD on data automatically extracted from the Web. This is a much harder problem than using an artificially balanced data set, as shown in Figure 4.

Automatic discovery of functional relations has been addressed in the database literature as *Functional Dependency Mining* (Huhtala et al., 1999; Yao and Hamilton, 2008). This focuses on discovering functional relationships between sets of attributes, and does not address the ambiguity inherent in natural language.

7 Conclusions and Future Work

We have described a case study of contradiction detection (CD) based on functional relations. In this context, we introduced and evaluated the AUCONTRAIRE system and its novel EM-style algorithm for determining whether an arbitrary phrase is functional. We also created a unique "natural" data set of *seeming contradictions* based on sentences drawn from a Web corpus, which we make available to the research community.

We have drawn two key lessons from our case study. First, many seeming contradictions (approximately 99% in our experiments) are not genuine contradictions. Thus, the CD task may be much harder on natural data than on RTE data as suggested by Figure 4. Second, extensive background knowledge is necessary to tease apart seeming contradictions from genuine ones. We believe that these lessons are broadly applicable, but verification of this claim is a topic for future work.

Acknowledgements

This research was supported in part by NSF grants IIS-0535284 and IIS-0312988, ONR grant N00014-08-1-0431 as well as gifts from the Utilika Foundation and Google, and was carried out at the University of Washington's Turing Center.

References

M. Banko and O. Etzioni. 2008. The tradeoffs between traditional and open relation extraction. In *Proceedings of ACL*.

M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the Web. In *Procs. of IJCAI*.

W.W. Cohen, P. Ravikumar, and S.E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *IIWeb*.

Cleo Condoravdi, Dick Crouch, Valeria de Paiva, Reinhard Stolle, and Daniel G. Bobrow. 2003. Entailment, intensionality and text understanding. In *Proceedings of the HLT-NAACL 2003 workshop on Text meaning*, pages 38–45, Morristown, NJ, USA. Association for Computational Linguistics.

I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 1–8.

Marie-Catherine de Marneffe, Anna Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *ACL 2008*.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38.

D. Downey, O. Etzioni, and S. Soderland. 2005. A Probabilistic Model of Redundancy in Information Extraction. In *Procs. of IJCAI*.

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, contrast and contradiction in text processing. In *AAAI*.

Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. 1999. TANE: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111.

B. MacCartney and C.D. Manning. 2007. Natural Logic for Textual Inference. In *Workshop on Textual Entailment and Paraphrasing*.

Ellen M. Voorhees. 2008. Contradictions and justifications: Extensions to the textual entailment task. In *Proceedings of ACL-08: HLT*, pages 63–71, Columbus, Ohio, June. Association for Computational Linguistics.

Hong Yao and Howard J. Hamilton. 2008. Mining functional dependencies from data. *Data Min. Knowl. Discov.*, 16(2):197–219.

A. Yates and O. Etzioni. 2007. Unsupervised resolution of objects and relations on the Web. In *Procs. of HLT*.