# Distant Supervision

- **Distant supervision** uses a large, external knowledge base to provide positive examples of relations for training.

Freebase, DBpedia, YAGO, …

- Typically, instances of these examples are harvested from the Web or a large text corpus and the contexts are used as noisy, labeled training data.

GOOD: large volumes of free positive examples!

BAD: “distant” supervision, so not every context will capture the relation. Hence the labeled data is noisy.

BAD: this approach is limited to relations found in large KBs.

# Distant Supervision for Relation Extraction without Labeled Data

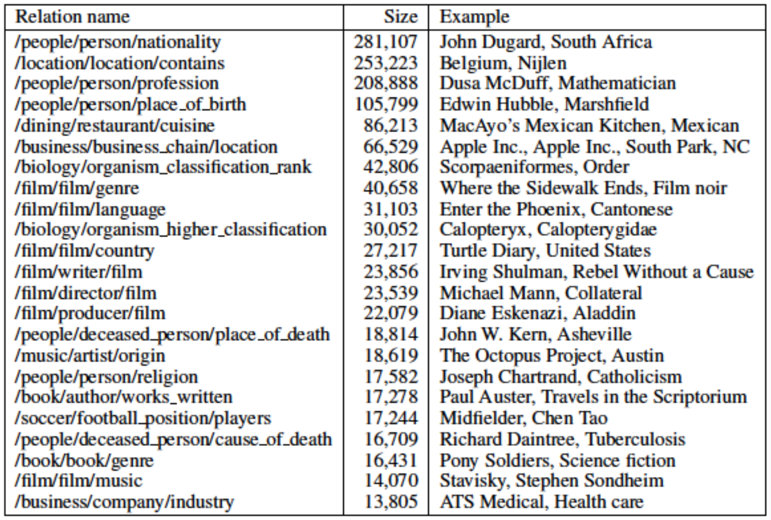
[Mintz et al., ACL-IJCNLP 2009] used distant supervision from Freebase to train a relation extractor.

- In July 2008 , Freebase contained 116 million instances of 7,300 relations between 9 million entities.
- They used data for the 102 largest relations, which had 1.8 million instances connecting 940,000 entities.

“relation” is an ordered, binary relation between entities.
Example: *person-nationality*

“relation instance” is an ordered pair of specific entities that participate in the relation.
Example: (John Steinbeck, United States)

# Sample of Freebase Relations



| Relation name | Size | Example |
|---|---|---|
| /people/person/nationality | 281,107 | John Dugard, South Africa |
| /location/location/contains | 253,223 | Belgium, Nijlen |
| /people/person/profession | 208,888 | Dusa McDuff, Mathematician |
| /people/person/place_of_birth | 105,799 | Edwin Hubble, Marshfield |
| /dining/restaurant/cuisine | 86,213 | MacAyo’s Mexican Kitchen, Mexican |
| /business/business_chain/location | 66,529 | Apple Inc., Apple Inc., South Park, NC |
| /biology/organism_classification_rank | 42,806 | Scorpaeniformes, Order |
| /film/film/genre | 40,658 | Where the Sidewalk Ends, Film noir |
| /film/film/language | 31,103 | Enter the Phoenix, Cantonese |
| /biology/organism_higher_classification | 30,052 | Calopteryx, Calopterygidae |
| /film/film/country | 27,217 | Turtle Diary, United States |
| /film/writer/film | 23,856 | Irving Shulman, Rebel Without a Cause |
| /film/director/film | 23,539 | Michael Mann, Collateral |
| /film/producer/film | 22,079 | Diane Eskenazi, Aladdin |
| /people/deceased_person/place_of_death | 18,814 | John W. Kern, Asheville |
| /music/artist/origin | 18,619 | The Octopus Project, Austin |
| /people/person/religion | 17,582 | Joseph Chartrand, Catholicism |
| /book/author/works_written | 17,278 | Paul Auster, Travels in the Scriptorium |
| /soccer/football_position/players | 17,244 | Midfielder, Chen Tao |
| /people/deceased_person/cause_of_death | 16,709 | Richard Daintree, Tuberculosis |
| /book/book/genre | 16,431 | Pony Soldiers, Science fiction |
| /film/film/music | 14,070 | Stavisky, Stephen Sondheim |
| /business/company/industry | 13,805 | ATS Medical, Health care |

Table 2: The 23 largest Freebase relations we use, with their size and an instance of each relation.

# General Approach

- Apply an NER tagger to identify entities.
- Extract sentences that contain two entities of types that can participate in a relation.
- Group all contexts that correspond to the same relation instance. The collective contexts serve as a single positive training example.

  Example: if a pair of entities occurs in 10 sentences, the features for all 10 sentences are combined.

- Train a multiclass logistic regression classifier to predict a relation between a pair of entities.

# Motivation for Collective Contexts

A key advantage of merging contexts from multiple instances is that some mentions will occur in relation contexts, some in ambiguous contexts, and some in non-relation contexts.

Example:

*S1: Steven Spielberg 's film Saving Private Ryan is loosely based on the brothers' story.*

**actor?**
**writer?**
**producer?**
**director?**

*S2: Allison co-produced the Academy Award-winning Saving Private Ryan, directed by Steven Spielberg.*

**book?**
**TV show?**
**play?**
**movie?**

# Lexical Features

Given a context containing two entities, the following information is extracted:

- the sequence of words between them
- the POS tags for the words between them
- a flag indicating which entity appeared first
- a window of *k* words to the left of Entity #1 and their POS tags
- a window of *k* words to the right of Entity #2 and their POS tags
- the named entity tags for the two entities

Each lexical feature is the **conjunction** of this information.

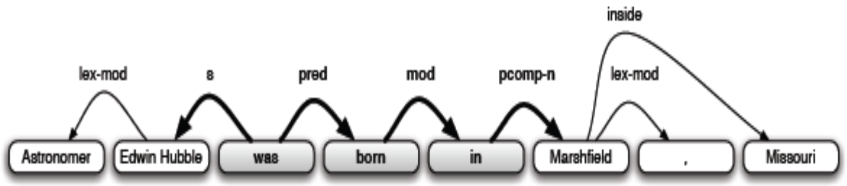
# Syntactic Features

Syntactic features are also generated from a dependency parse of the sentence.

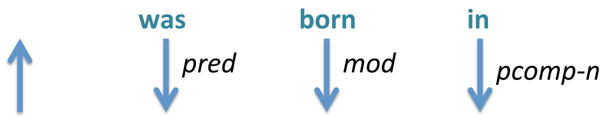
- A **dependency path** between the two entities, which is a series of dependencies, directions, and words/chunks representing a traversal of the parse.
- A pair of left and right window node for each entity, which capture the words to the left and right of the entities that are not part of the dependency path.
- the named entity tags for the two entities

Each syntactic feature is a **conjunction** of this information.

# Dependency Path Example

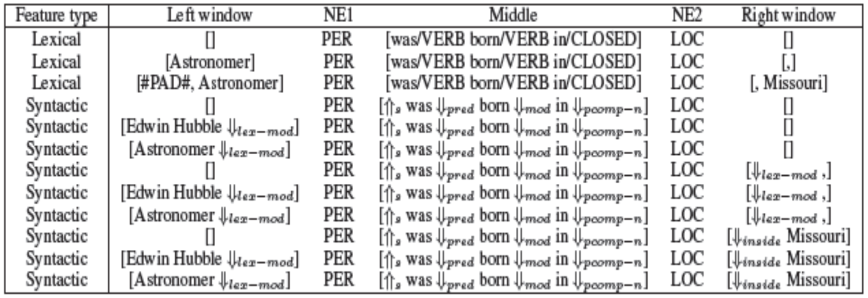


The dependency path begins with "Edwin Hubble" and includes the link traversals that lead to "Marshfield":



**was** *pred* **born** *mod* **in** *pcomp-n*

# Example Features



| Feature type | Left window | NE1 | Middle | NE2 | Right window |
|---|---|---|---|---|---|
| Lexical | [] | PER | [was/VERB born/VERB in/CLOSED] | LOC | [] |
| Lexical | [Astronomer] | PER | [was/VERB born/VERB in/CLOSED] | LOC | [,] |
| Lexical | [#PAD#, Astronomer] | PER | [was/VERB born/VERB in/CLOSED] | LOC | [, Missouri] |
| Syntactic | [] | PER | [$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$] | LOC | [] |
| Syntactic | [Edwin Hubble $\Downarrow_{lex-mod}$] | PER | [$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$] | LOC | [] |
| Syntactic | [Astronomer $\Downarrow_{lex-mod}$] | PER | [$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$] | LOC | [] |
| Syntactic | [] | PER | [$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$] | LOC | [$\Downarrow_{lex-mod}$ ,] |
| Syntactic | [Edwin Hubble $\Downarrow_{lex-mod}$] | PER | [$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$] | LOC | [$\Downarrow_{lex-mod}$ ,] |
| Syntactic | [Astronomer $\Downarrow_{lex-mod}$] | PER | [$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$] | LOC | [$\Downarrow_{lex-mod}$ ,] |
| Syntactic | [] | PER | [$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$] | LOC | [$\Downarrow_{inside}$ Missouri] |
| Syntactic | [Edwin Hubble $\Downarrow_{lex-mod}$] | PER | [$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$] | LOC | [$\Downarrow_{inside}$ Missouri] |
| Syntactic | [Astronomer $\Downarrow_{lex-mod}$] | PER | [$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$] | LOC | [$\Downarrow_{inside}$ Missouri] |

Table 3: Features for ‘Astronomer Edwin Hubble was born in Marshfield, Missouri’.

# Conjunctive Features

- Note that the conjunctive features are very specific!
- Normally, such specific features would match very few contexts and not be useful for a classifier
- But, this scenario is using very large amounts of data, so the expectation is that they will match some contexts and serve as *low-recall, high-precision* features.

# The Classifier

- As negative training data, random entity pairs that do not participate in a Freebase relation are used to generate feature vectors for an “**unrelated**” relation. This may produce some noise, but the effect should be small.
- They randomly sample 1% of entity pairs that are not in a Freebase relation.
- Testing: a multi-class logistic classifier takes an entity pair as input, constructs a feature vector for it, and returns a relation name with a confidence score.
- All entity pairs can then be ranked by their confidence scores to identify the N most likely new relation instances.

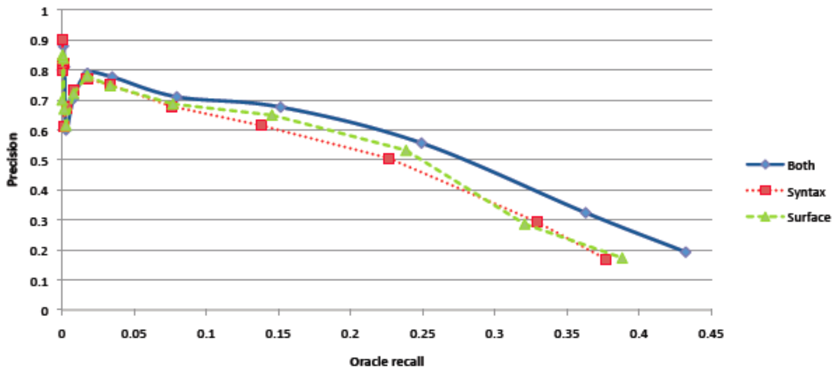
# Text Corpus

- Corpus: full text of all Wikipedia articles.
  – 1.8 million articles, 14.3 sentences per article on average
  – 800,000 used for training, 400,000 used for testing
- Wikipedia texts chosen because:
  – “sentences tend to make explicit many facts that might be omitted in newswire”
  – “must of the information in Freebase is derived from tabular data from Wikipedia, meaning that Freebase relations are more likely to appear in sentences in Wikipedia”

# Evaluation

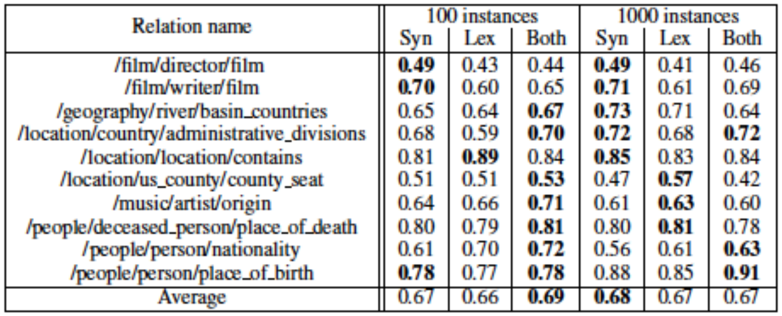
- **Automatic:** half of the instances for each relation are held out for testing. Newly identified instances are compared to the held out data.
  - 900,000 training instances; 900,000 test instances
- **Manual:** humans review each labeled entity pair and determine whether a relation exists between them.
  - all 1.8 million instances were used for training.
  - 3 experiments: only lexical features, only syntactic features, lexical and syntactic features.
  - evaluated 10 relations most frequent in test data by sampling from first 100 and 1,000 instances generated by the classifier.

# Automatic Evaluation Results



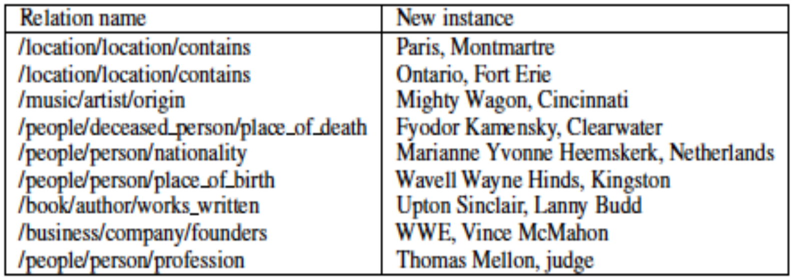
For the higher recall levels, most of the instances were labeled into 3 relations: *location-contains (60%), person-place-of-birth (13%), person-nationality (10%)*

# Human Evaluation Results



| Relation name | 100 instances | | | 1000 instances | | |
|---|---|---|---|---|---|---|
| | Syn | Lex | Both | Syn | Lex | Both |
| /film/director/film | **0.49** | 0.43 | 0.44 | **0.49** | 0.41 | 0.46 |
| /film/writer/film | **0.70** | 0.60 | 0.65 | **0.71** | 0.61 | 0.69 |
| /geography/river/basin_countries | 0.65 | 0.64 | **0.67** | **0.73** | 0.71 | 0.64 |
| /location/country/administrative_divisions | 0.68 | 0.59 | **0.70** | **0.72** | 0.68 | **0.72** |
| /location/location/contains | 0.81 | **0.89** | 0.84 | **0.85** | 0.83 | 0.84 |
| /location/us_county/county_seat | 0.51 | 0.51 | **0.53** | 0.47 | **0.57** | 0.42 |
| /music/artist/origin | 0.64 | 0.66 | **0.71** | 0.61 | **0.63** | 0.60 |
| /people/deceased_person/place_of_death | 0.80 | 0.79 | **0.81** | 0.80 | **0.81** | 0.78 |
| /people/person/nationality | 0.61 | 0.70 | **0.72** | 0.56 | 0.61 | **0.63** |
| /people/person/place_of_birth | **0.78** | 0.77 | **0.78** | 0.88 | 0.85 | **0.91** |
| Average | 0.67 | 0.66 | **0.69** | **0.68** | 0.67 | 0.67 |

# Learned Examples not in Freebase



| Relation name | New instance |
|---|---|
| /location/location/contains | Paris, Montmartre |
| /location/location/contains | Ontario, Fort Erie |
| /music/artist/origin | Mighty Wagon, Cincinnati |
| /people/deceased_person/place_of_death | Fyodor Kamensky, Clearwater |
| /people/person/nationality | Marianne Yvonne Heemskerk, Netherlands |
| /people/person/place_of_birth | Wavell Wayne Hinds, Kingston |
| /book/author/works_written | Upton Sinclair, Lanny Budd |
| /business/company/founders | WWE, Vince McMahon |
| /people/person/profession | Thomas Mellon, judge |

# Analysis

- The syntactic features showed benefits over just the lexical features, so they inspected examples to understand how they helped.
- The syntactic features consistently helped with the *director-film* and *writer-film* relations, which are particularly ambiguous.
- They observed many examples with a large distance between the director's name and the film, for example:

  *Back Street is a 1932 film made by Universal Pictures, directed by John M. Stahl, and produced by Carl Laemmle Jr.*

- These cases would have long lexical features, but often short dependency paths.

# Kernel Methods

- **Kernel methods** are commonly used for machine learning approaches to relation extraction.
- A **kernel function** is essentially a similarity function that compares two instances.
- Instead of defining a set of features for a classifier to use, you define a kernel function that measures the similarity between instances, often in a detailed way.
- Most commonly used with SVMs, but can be used with some other learning algorithms as well.

# Examples of Relation Extraction Kernels

- **String kernels:** given two strings, compute the number of common subsequences of characters, usually weighted by length and contiguity. This number can be computed in polynomial time without enumerating them all (which would be exponential!).
- **Parse tree kernels:** given two parse trees, compute the number of common subtrees. Has been done with full parse trees and shallow parses.
- **Dependency tree kernels:** given two dependency parses, compute their similarity.

  Attributes of the parse (e.g., POS tags, entity types, etc.) can be considered in the kernel function.

# Question Answering as Relation Extraction

"Factoid" question answering systems find answers to questions that have a short, well-defined answer type (e.g., Who/Where/When).

Many factoid questions are essentially looking for a specific instance of a relation. For example:

| | |
|---|---|
| Q: *When was Mozart born?* | A: *1756* |
| Q: *Where is the University of Utah?* | A: *Salt Lake City* |
| Q: *Who invented Kevlar?* | A: *Stephanie Kwolek* |

Factoid Q/A pairs can be used to learn patterns or train a classifier to recognize specific relations that are common types of questions.

Birthyear *(Mozart, 1756)*
LocationOf (*University of Utah*, *Salt Lake City)*
InventedBy(*Kevlar*, *Stephanie Kwolek)*

# Learning Surface Patterns for Q/A

Surface patterns that link question and answer terms can be learned automatically from the Web [Ravichandran & Hovy 2002].

1. Submit Q/A pairs for a relation (e.g., *Mozart 1756*) to a search engine and download the top 1000 documents.
2. Extract sentences that contain both the Q and A terms.
3. Identify all substrings and their counts using a suffix. Filter substrings that do not contain both the Q and A terms. Replace the Q term with <NAME> and the A term with <ANSWER>.
4. For each phrase in the suffix tree, evaluate its precision.

   Query the Web with just the Q term and extract sentences that match the pattern. Compute the % of sentences that contain the answer.

# Examples of Learned Patterns

| **Birthyear** | |
|---|---|
| 1.0 | <NAME> (<ANS> -) |
| .85 | <NAME> was born on <ANS> |
| .60 | <NAME> was born in <ANS> |
| .59 | <NAME> was born <ANS> |
| .53 | <ANS> <NAME> was born |
| .50 | – <NAME> (<ANS> |
| .36 | <NAME> (<ANS> – |
| .32 | <NAME> (<ANS>) , |
| .28 | born in <ANS> , <NAME> |
| .20 | of <NAME> (<ANS> |

| **Inventor** | |
|---|---|
| 1.0 | <ANS> invents <NAME> |
| 1.0 | the <NAME> was invented by <ANS> |
| 1.0 | <ANS> invented the <NAME> in |
| 1.0 | <ANS>'s invention of the <NAME> |
| 1.0 | <ANS> invents the <NAME> |
| 1.0 | <ANS>'s <NAME> was |
| 1.0 | <NAME>, invented by <ANS> |
| 1.0 | <ANS>'s <NAME> and |
| 1.0 | that <ANS>'s <NAME> |
| 1.0 | <NAME> was invented by <ANS> , |

# Summary

Relation extraction has been studied in a variety of ways.

- patterns vs. classifiers
- finding instances in specific contexts vs. instances extracted from a set of contexts
- within applications such as question answering

Although progress has been made, it is far from solved.
The focus has primarily been on:

- the most common, binary relations
- relations that are explicitly stated locally (within a sentence)