Relation Extraction

- **Relation extraction** tasks involve identifying relationships between entities or concepts.
- A relation is typically a static fact that is true for a substantial period of time.
- Relations are often expressed as triples, with the relation represented as a predicate. Common relations are:
  - LocatedIn(Microsoft, Redmond)
  - Birthyear(Mozart, 1756)
  - FatherOf(Bill Clinton, Chelsea Clinton)

Applications for Relation Extraction

- Automatically create and augment large structured knowledge bases, such as FreeBase, DBpedia, and YAGO.
- Automatic creation of biographical and organizational profiles.
- Information retrieval and question answering. For example, give the name of a famous entity (e.g., Mozart) to Google and see the information box that pops up!
- Natural language understanding. ☺ Identifying relations is essential to understand stories!

ACE

- The **Automated Content Extraction (ACE)** program has conducted community-wide performance evaluations of IE systems.
- The ACE tasks have focused on the detection and classification of entities, relations, and events as well as within-document and cross-document coreference resolution.
- Terms introduced by the ACE evaluations include *mentions* (instances irrespective of type) and *geo-political entities (GPEs)*. GPE was introduced to distinguish uses of (typically) location names that correspond to different entity types.

the riots in *Miami* → Location

Miami imposed a curfew → Organization

Miami railed against the curfew → Person

ACE 2008 Relation Types

Table 4 ACE08 Relation Types and Subtypes
(Relations marked with an * are symmetric relations.)

Type	**Subtype**
ART (artifact)	User-Owner-Inventor-Manufacturer
GEN-AFF (General affiliation)	Citizen-Resident-Religion-Ethnicity, Org-Location
METONYMY*	*None*
ORG-AFF (Org-affiliation)	Employment, Founder, Ownership, Student-Alum, Sports-Affiliation, Investor-Shareholder, Membership
PART-WHOLE (part-to-whole)	Artifact, Geographical, Subsidiary
PER-SOC* (person-social)	Business, Family, Lasting-Personal
PHYS* (physical)	Located, Near

Relations for Medical Texts

Identifying and classifying relations is also important for analyzing medical texts. For example, from the i2b2 relation annotation guidelines:

Treatment_Improves(Treatment, Problem)

hypertension was controlled on hydrochlorothiazide

Treatment_Worsens(Treatment, Problem)

the tumor was growing despite the chemotherapy regimen

Test_Reveals_Problem(Test, Disease)

an echocardiogram revealed a pericardial effusion

Problem_Indicates_Problem(Problem, Problem)

azotemia presumed secondary to sepsis

Relation Extraction Approaches

- **Hand-crafted Patterns:** manually define patterns that are likely to identify the targeted relation. For some relations, a small number of phrases will capture many instances of the relation.
- **Supervised Classifiers:** A common approach is to identify contexts that contain an entity pair and then classify the context as being positive or negative with respect to the relation.
- **Weakly Supervised Learning:** bootstrapped learning from seed examples and *distant supervision* have been used for relation extraction.

Basic Features for Relation Extraction

- Entity Features
  - the types of the named entities (e.g., Location, Person, etc.)
  - the mention types of the entities (name, nominal, or pronoun)
  - the head noun of the entities
- Lexical Context
  - the words before, between, and after the entities
  - the distance between the entities
  - whether other mentions occur between the entities

Syntactic Features for Relation Extraction

- Chunking Features
  - phrasal heads containing the entities
  - phrasal heads of the before/between/after contexts
- Dependency Parsing Features
  - dependency relations linked to the entities
  - pairs of heads or entity types and dependent words
- Parse Tree Features
  - parse tree paths connecting one entity to the other

Semantic Features for Relation Extraction

- Semantic class information can be used to distinguish between relation subtypes. For example:
  - *CitizenOf* must link to a country, while *ResidentOf* can link to any location).
  - *Social* relations (e.g., family members) are different from other person relations (e.g., EmployeeOf).
- Semantic knowledge is typically based on WordNet, lists harvested from the Web, or manually defined (e.g., family member terms are a relatively small set).

Snowball

[Agichtein & Gravano, 2000]

- Snowball is a weakly supervised, bootstrapping method for learning patterns and instances of relations between two named entities.
- Snowball begins with "seed tuples" that represent instances of the targeted relation, and iteratively learns relation patterns as well as new instance pairs (tuples).
- Snowball relies on a named entity recognizer to identify contexts that contain targeted types of entities. For example:

  <LOCATION> *-based* <ORG> → *Redmond-based Microsoft*

  whereas arbitrary contexts can be quite general, e.g.

  <STRING> *-based* <STRING> → *alcohol-based solvent*

Snowball's Flow of Control

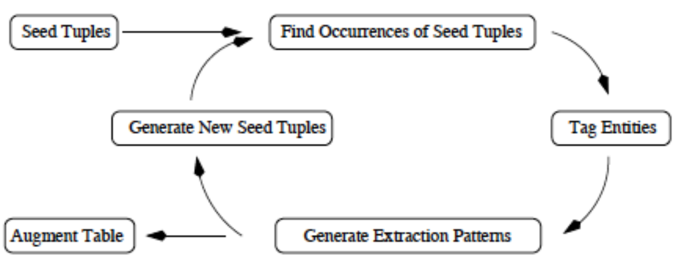

Figure 2: The main components of *Snowball*.

Pattern Representation

- Each pattern is a 5-tuple of the form:

  <Left, Tag1, Middle, Tag2, Right>

- *Left*, *Middle*, and *Right* are vectors of terms with weights representing contexts.
- *Tag1* and *Tag2* are named entity classes.

Example

	Left	*Tag1*	*Middle*	*Tag2*	*Right*	
<	{<*the*, 0.2>},	LOCATION,	{<-, 0.5>,<*based*, 0.5>},	ORG,	{}	>

"*the Irving – based Exxon Corporation*"

Pattern Matching Function

- Given two tuples:

$T_P = \langle L_P, T_1, M_P, T_2, R_P \rangle \qquad T_S = \langle L_S, T'_1, M_S, T'_2, R_S \rangle$

- The degree of match is defined as:

Match(T_P,T_S) =

$$\begin{cases} L_P \bullet L_S + M_P \bullet M_S + R_P \bullet R_S & \text{if the tags match} \\ 0 & \text{otherwise} \end{cases}$$

The dot • indicates the vector dot product operation.

Learning Patterns

- Snowball generates a 5-tuple for each context where a seed instance pair occurs.
- A clustering algorithm groups the 5-tuples that are similar based on the Match function, using a minimum similarity threshold.
- The left, middle, and right context vectors are collapsed into left, middle, and right **centroid vectors**, which then form a **Snowball pattern**: $\langle \overline{L}_S, T_1, \overline{M}_S, T_2, \overline{R}_S \rangle$

```
Best = FindClosestCluster(T, τSIM);
if (Best)
    UpdateCentroid(Best, T);
else
    CreateNewCluster(T);
```

Evaluating Patterns

Patterns that extract < τ_{SUP} seed tuples are filtered, and the rest are assigned a confidence value. Two confidence measures were tried:

$$\textit{Confidence}(P) = \frac{P.positive}{P.positive + P.negative}$$

$$\textit{Confidence}_{RlogF}(P) = \textit{Confidence}(P) * \log_2(P.positive)$$

Since confidence values should range from 0 to 1, *Confidence*$_{RlogF}$ values are normalized by the largest confidence value of any pattern.

Learned Pattern Examples

Conf	*middle*	*right*
1	<based, 0.53> <in, 0.53>	<, , 0.01>
0.69	<', 0.42> <s, 0.42> < headquarters, 0.42> <in, 0.12>	
0.61	<(, 0.93>	<), 0.12>

Table 2: Actual patterns discovered by *Snowball*. (For each pattern the *left* vector is empty, *tag1* = *ORGANIZATION*, and *tag2* = *LOCATION*.)

Discovering New Tuples

- To discover new *tuples* (entity pairs: <E_1, E_2>), Snowball first extracts sentences that contain entities of the desired types.
- For each sentence, a 5-tuple is created: T = <L_P, T_1, M_P, T_2, R_P>, where T_1 is the class of E_1 , and T_2 is the class of E_2.
- The 5-tuple is matched against the patterns and a candidate tuple (entity pair) is generated for every pattern X such that $\text{Match}(T, T_X) \geq \tau_{SIM}$
- Each candidate tuple is linked with the set of patterns that generated it and then scored to decide which ones to keep and use for subsequent learning.

Examples of CandidateTuples

Organization	*Location of Headquarters*
3COM CORP	SANTA CLARA
3M	MINNEAPOLIS
AIR CHINA	BEIJING
FEDERAL EXPRESS CORP	MEMPHIS
FRUIT JELLIES	APPLE
MERRILL LYNCH & CO	NEW YORK
NETSCAPE	MOUNTAIN VIEW
NINTENDO CORP	TOKYO

Table 2: Some tuples discovered during *Snowball*'s first iteration.

Some tuples will be incorrect, in this case due to NER errors.
So Snowball assigns a confidence score to each tuple.

Scoring Tuples

For tuple T, Snowball implements the following intuition:

Prob(T) = 1 – (probability that all patterns fired incorrectly)

$$\text{Prob}(T) = 1 - \prod_{i=0}^{|P|} (1 - \text{Prob}(p_i))$$

where $P=\{p_i\}$ is the set of patterns that generated tuple T

$$\text{Confidence}(T) = 1 - \prod_{i=0}^{|P|} (1 - (\textit{Confidence}(p_i) * \textit{Match}(C_i, p_i))$$

where,

C_i is the context associated with T that matched p_i

Tuple Scoring Example

Suppose the candidate tuple T = <Microsoft, Redmond>
was generated by two patterns with the following confidence values:

<{}, ORG, {in}, LOCATION, {}> 0.5

<{}, ORG, {of}, LOCATION, {}> 0.6

Conf(T) = 1 – ((1-0.5) * (1-0.6)) = 1 – (0.5*0.4) = .80

Even though both patterns are likely to produce both positive and negative examples, a tuple that is generated by both of them is likely to be a positive example!

Updating Confidence Scores

During the learning process, the confidence scores for patterns and tuples are updated as a weighted combination of old and new scores.

$$Confidence(P) = Confidence_{NEW}(P) * W_{UPDATE} + Confidence_{OLD}(P) * (1-W_{UPDATE})$$

$$Confidence(T) = Confidence_{NEW}(T) * W_{UPDATE} + Confidence_{OLD}(T) * (1-W_{UPDATE})$$

Parameter Values in Snowball

Parameter	*Value*	*Description*
τ_{sim}	0.6	minimum degree of match (Section 2.1)
τ_t	0.8	minimum tuple confidence (Section 2.3)
τ_{sup}	2	minimum pattern support (Section 2.1)
I_{max}	3	number of iterations of *Snowball*
W_{middle}	0.6	weight for the *middle* context (Section 2.1)
W_{left}	0.2	weight for the *left* context (Section 2.1)
W_{right}	0.2	weight for the *right* context (Section 2.1)

Table 4: Parameter values used for evaluating *Snowball* on the test collection.

Evaluation

- Snowball was designed to learn LocatedIn(ORG,LOC) relations and produce entity pairs for this relation.
- Snowball's goal was to generate tables of entity pairs from a corpus, as opposed to typical IE systems that want to find every instance of a relation.
- An "Ideal" set of entity pairs was created by:
  - compiling (ORG,LOC) pairs from "Hoover's Online" web site
  - retained pairs for which the organization name appears in the corpus with its location nearby
- However Hoover's is far from complete. So manual samples of extracted tuples were evaluated by hand.

Snowball Results

100 extracted tuples were evaluated by hand for each system.

Three types of errors were labeled:

Location Errors = mistagging a location (NER error)
Organization Errors = mistagging an organization (NER error)
Relationship Errors = misidentifying the relation

			Type of Error			
	Correct	*Incorrect*	*Location*	*Organization*	*Relationship*	P_{Ideal}
DIPRE	74	26	3	18	5	90%
Snowball (all tuples)	52	48	6	41	1	88%
Snowball ($\tau_t = 0.8$)	93	7	3	4	0	96%
Baseline	25	75	8	62	5	66%