Relation and Event Extraction and Temporal Reasoning

Natalie Parde

UIC CS 421

Natural language is unstructured.

There was nothing so *very* remarkable in that; nor did Alice think it so *very* much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be late!" (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually *took a watch out of its waistcoat-pocket*, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.

Many natural language reasoning tasks require that language is converted to structured representations.

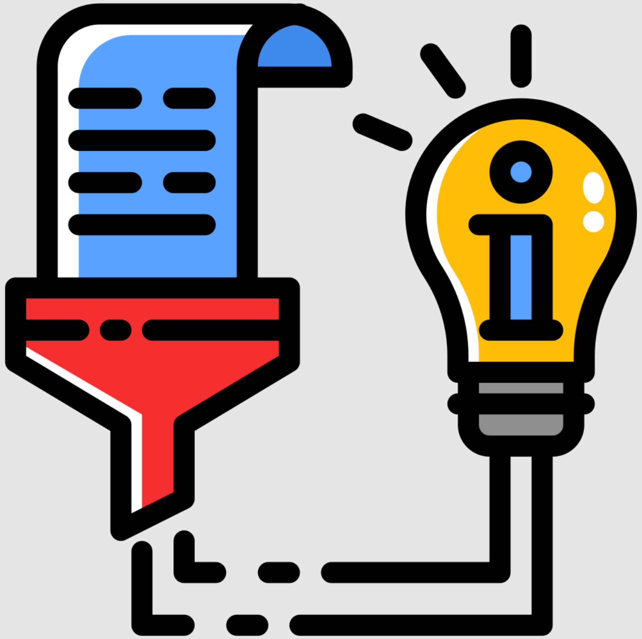
There was nothing so *very* remarkable in that; nor did Alice think it so *very* much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be late!" (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually *took a watch out of its waistcoat-pocket*, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.

Interdimensional Travel

- Leader: Rabbit
- Follower: Alice
- Vehicle: Rabbit-hole

Information Extraction



- The process of converting unstructured information to **structured data**
- Many forms of information extraction, with important tasks including:
  - Relation extraction
  - Event extraction

Relation Extraction

- **The process of finding and classifying semantic relations among entities mentioned in text**
- Semantic relations may be structural, geospatial, or hierarchical
- Example relations include:
  - Physical location (e.g., relating a person to a geopolitical entity)
  - Organizational subsidiaries (e.g., relating a specialized company to its parent company)
  - Social roles (e.g., relating a person to their sibling)

This Week's Topics

Tuesday

- Named Entity Recognition
- Relation Extraction

Thursday

- Event Extraction
- Temporal Logic
- Temporal Analysis

This Week's Topics

Tuesday

- Named Entity Recognition
- Relation Extraction

Thursday

- Event Extraction
- Temporal Logic
- Temporal Analysis

What is an entity?

- **Entities** in text are specific subjects that have been referenced
  - People
  - Locations
  - Times
  - Organizations
- Entities are typically identified and classified using a process known as **named entity recognition**

Named Entity Recognition

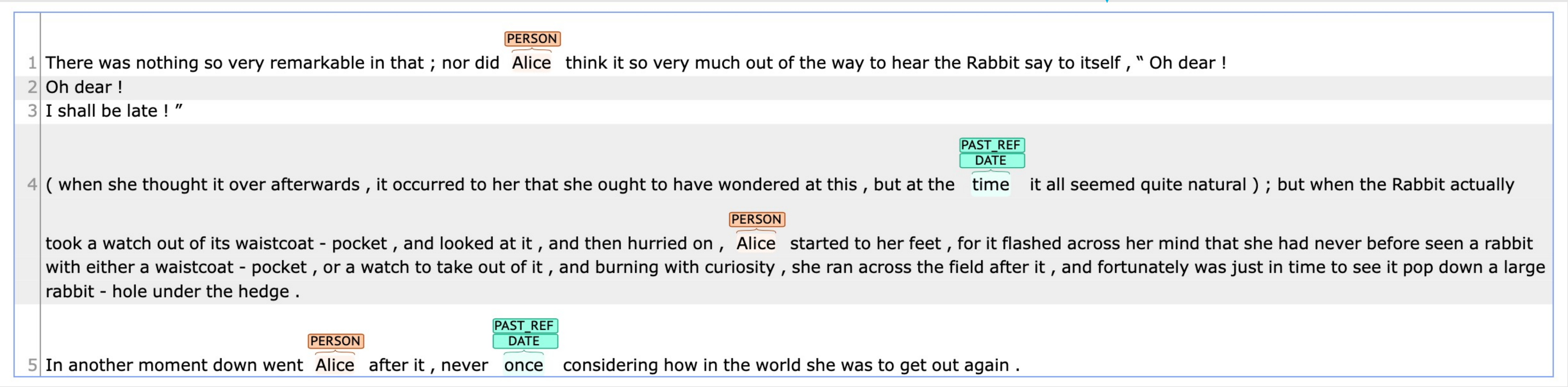
- Goal:
  1. Find spans of text that constitute proper names
  2. Tag the type of entity
- **Named entity tagsets** define the specific types of entities identified by a named entity recognizer

- Common entity tags:
  - **PER:** Person
    - People or characters
  - **LOC:** Location
    - Regions, mountains, or seas
  - **ORG:** Organization
    - Companies or sports teams
  - **GPE:** Geopolitical entity
    - Countries or states
- Entities can also include **temporal** and **numerical** expressions

Sample Named Entity Tagger Output

There was nothing so *very* remarkable in that; nor did Alice think it so *very* much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be late!" (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually *took a watch out of its waistcoat-pocket*, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.



Named entity recognition is challenging!

- Named entity recognition is challenging for several reasons:
  - Text segmentation can be ambiguous
    - What is part of the entity and what isn't (where are the entity's boundaries)?
  - Type determination can be ambiguous
    - Some words refer to different types of entities depending on the context

- Did [$_{Org}$ Chicago] win the game?
- I'm visiting friends in [$_{Loc}$ Chicago].
- [$_{GPE}$ Chicago] proposed a new tax ordinance.

How does named entity recognition work?

- Typically framed as a sequence labeling problem that performs span recognition
- BIO tags can be used to capture both span boundaries and named entity types

B-PER I-PER O O O O O O B-ORG I-ORG I-ORG I-ORG O O O O O B-LOC

Natalie Parde is a faculty member at the University of Illinois Chicago, a large public university in Chicago.

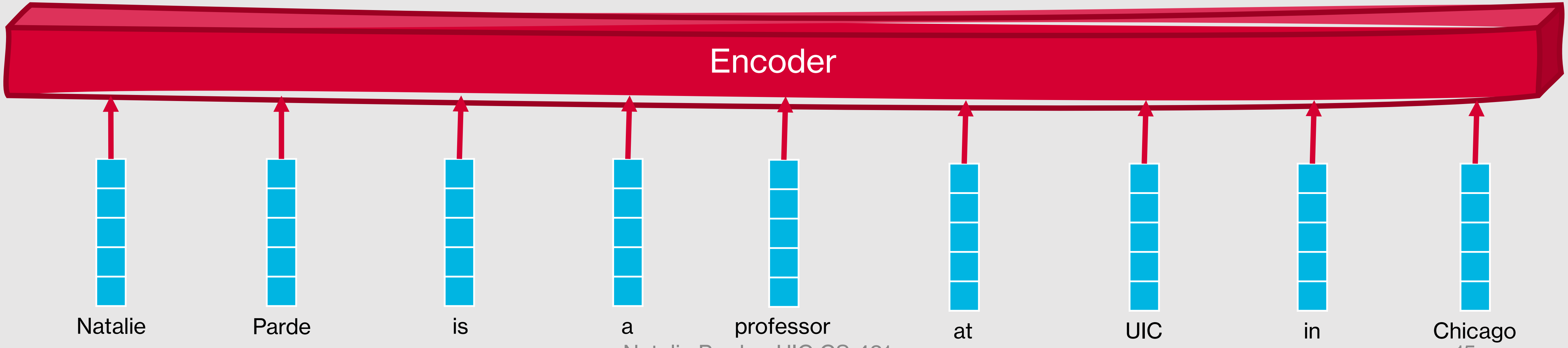
BIO Tagging for Named Entity Recognition

- Create distinct B and I tags for each named entity class
    - This results in 2n + 1 tags (only one O tag is needed)
- Assign a single label to each input word
- Other common variations of BIO tagging:
    - **IO** tagging (no B labels)
    - **BIOES** tagging (adds E tags for ends of spans, and S tags for single-word spans)

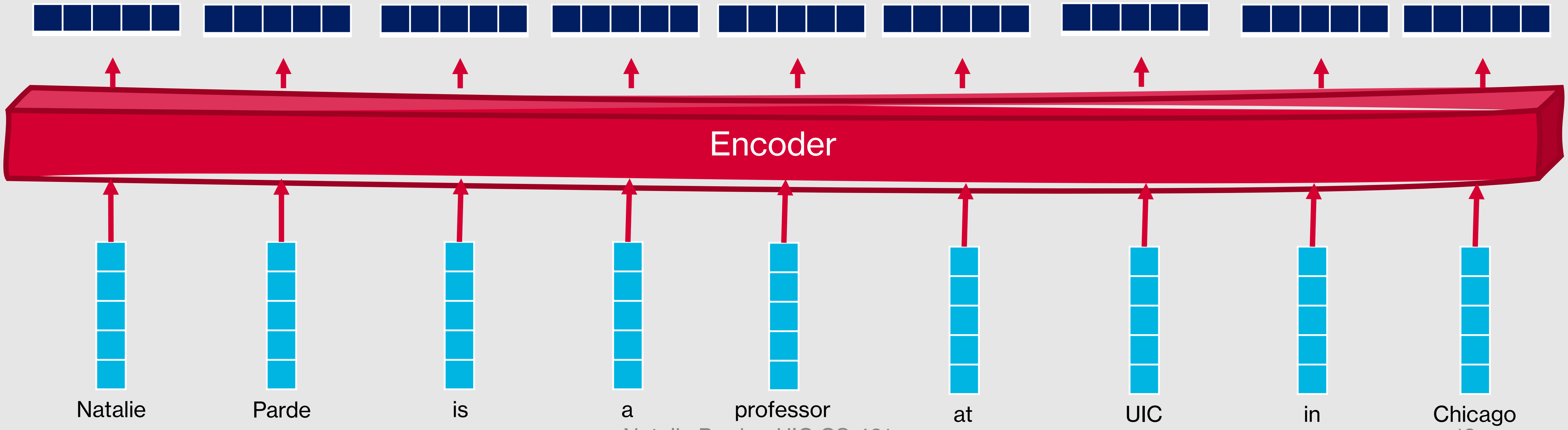
What kind of approaches do we use to implement named entity recognizers?

- Typically we will use supervised machine learning approaches to perform BIO tagging for named entity recognition
- Popular general-purpose named entity corpus:
  - **OntoNotes:** https://catalog.ldc.upenn.edu/LDC2013T19
    - Available in English, Chinese, and Arabic
- Popular biomedical named entity corpus:
  - **CRAFT:** https://github.com/UCDenver-ccp/CRAFT
- Popular literary named entity corpus:
  - **LitBank:** https://github.com/dbamman/litbank

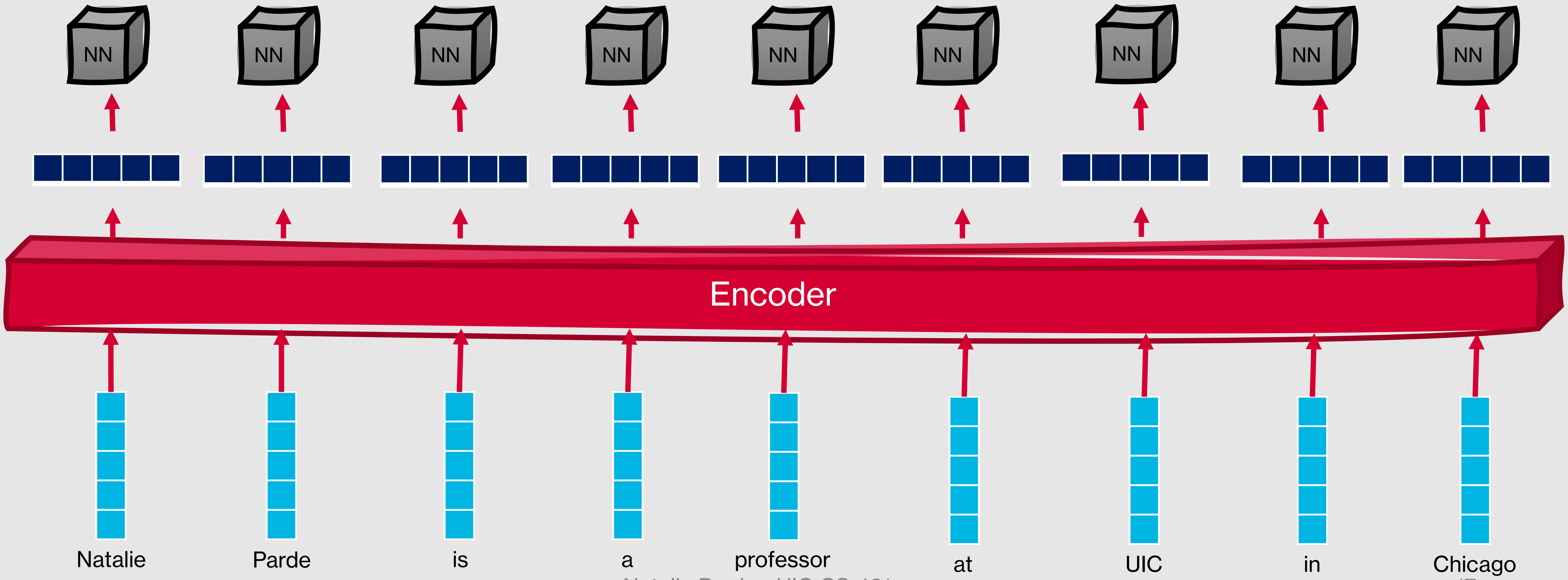
Example Neural Named Entity Recognizer



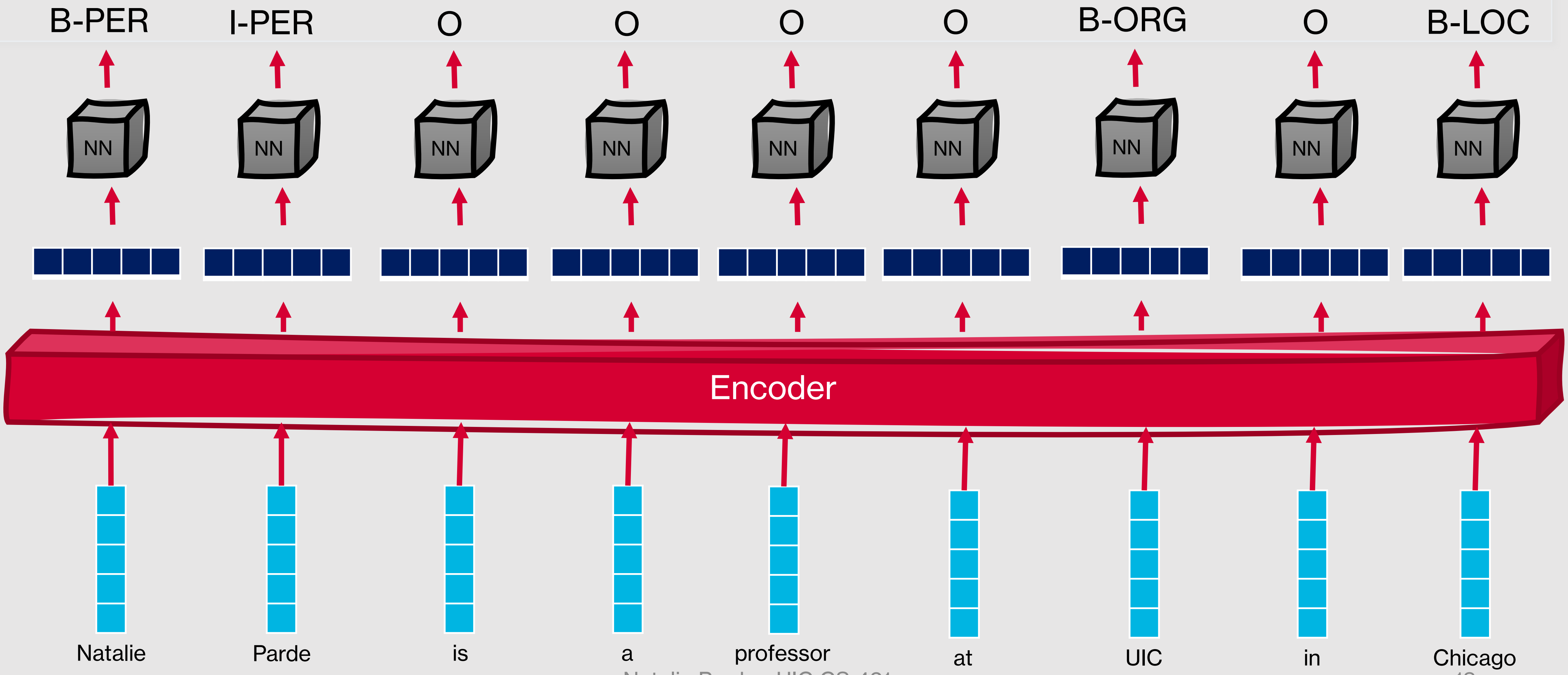
Example Neural Named Entity Recognizer



Example Neural Named Entity Recognizer



Example Neural Named Entity Recognizer



Rule-Based Named Entity Recognition

- Commercial approaches often combine supervised models with rule-based methods to ensure high precision in specialized settings
- One way this is done is by first performing rule-based passes (e.g., using regex) over the text to tag unambiguous entity mentions, and then using supervised learning methods that incorporate the output of these passes as features or input

This Week's Topics

Named Entity Recognition
Relation Extraction

Tuesday

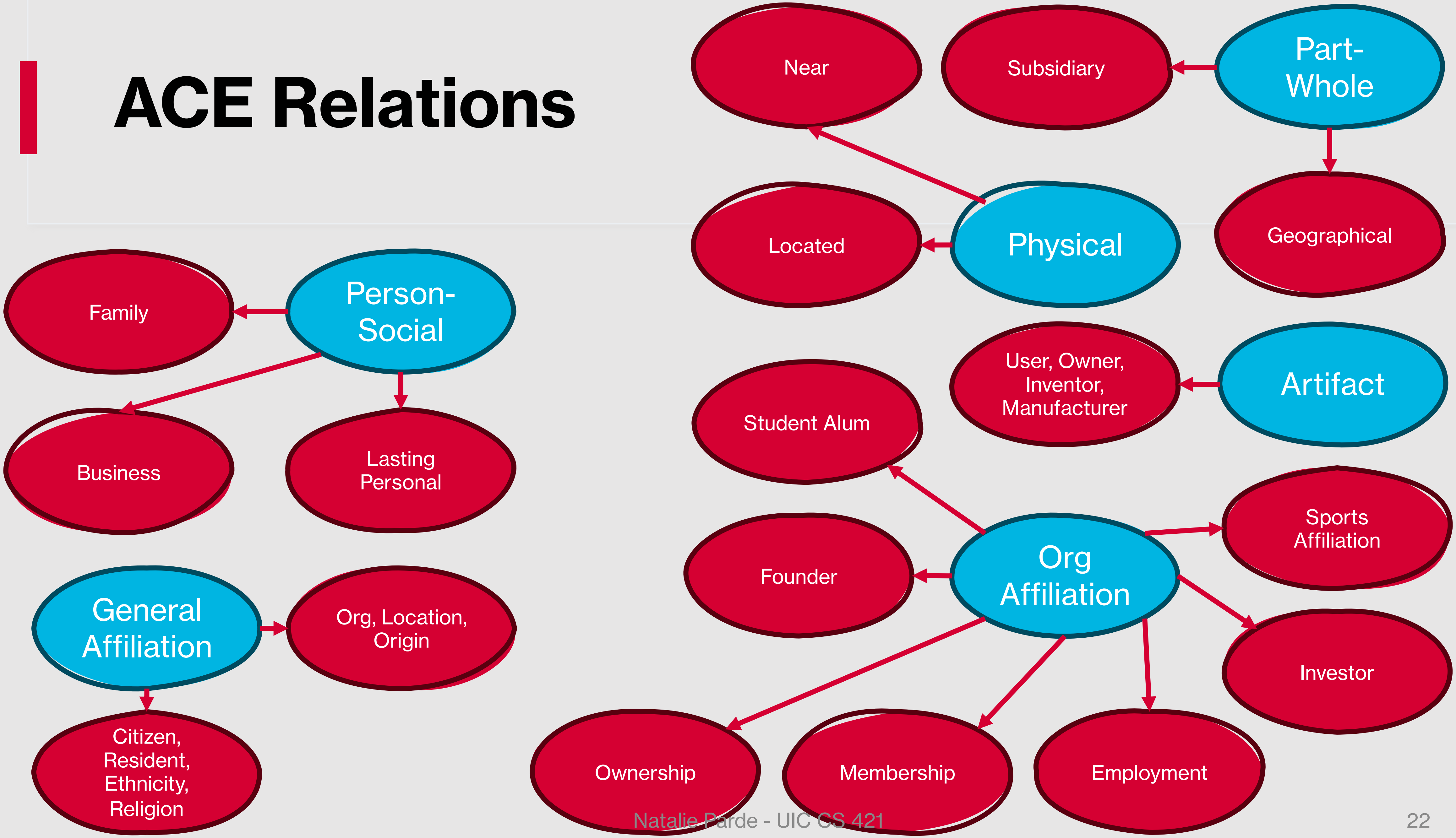
Thursday

Event Extraction
Temporal Logic
Temporal Analysis

Once we've detected named entities, we can determine relationships between them.

- Just like with part-of-speech tagging and text parsing tasks, a variety of relational tagsets can be used
- Popular tagset: **Automatic Content Extraction (ACE) Relations**
  - 17 physical, membership, affiliation, citizenship, and discourse relations
  - Each relation links two entities

ACE Relations



Also useful from the perspective of model-theoretic semantics!

- Logical relations consist of ordered tuples over elements within a domain
- The relations we can extract from text nicely fit this format
  - Domain elements → named entities

Updated version of our model-based example....

Domain:
Natalie, Devika, Nikolaos, Mina = {a, b, c, d}
Giordano's, IDOF, Artopolis = {e, f, g}

Classes:
PER = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}
ORG = {Giordano's, IDOF, Artopolis} = {e, f, g}

Relations:
Org Affiliation = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}

Domain-Specific Relation Sets

- For specific applications, domain-specific sets of relations may be used instead
- Popular set of entities and relations in the medical domain: **UMLS**
  - 134 broad subject categories and entity types
  - 54 relations between entities
- Browse the UMLS semantic network: https://uts.nlm.nih.gov/uts/umls/semantic-network/root

UMLS Example

Entity or event	Relation	Entity or event
Injury	Disrupts	Physiological function
Medical device	Diagnoses	Disease or syndrome
Bodily location	Location-of	Biologic function
Anatomical structure	Part-of	Organism
Pharmacologic substance	Causes	Pathological function
Pharmacologic substance	Treats	Pathologic function

Doppler echocardiography can be used to diagnose left anterior descending artery stenosis in patients with type 2 diabetes

doppler echocardiography diagnoses *acquired stenosis*

Other Sources of Relations

- Wikipedia contains structured tables associated with certain articles that can be turned into a metalanguage known as the **Resource Description Framework (RDF)**
  - RDF tripes are relationship(entity, entity) relations created using this metalanguage, which map to subject-predicate-object expressions
    - Location(Learning Center Building C, UIC)
- **DBpedia** was created from Wikipedia: https://www.dbpedia.org/
- **TACRED** contains 106,264 examples of relation triples from news and web text, with 41 relation types: https://nlp.stanford.edu/projects/tacred/
- **SemEval 2010 Task 8** dataset contains 10,717 examples of relation triples, with 9 relation types: http://www.kozareva.com/downloads.html

How can we extract relations?

- There are many ways to perform automated relation extraction:
  - Rule-based methods
  - Supervised learning
  - Semi-supervised learning
  - Unsupervised learning

Rule-Based Relation Extraction

- Early and still common approach
- Works by searching for specific lexicosyntactic patterns originally proposed by Marti Hearst, known as **Hearst patterns**
  - https://aclanthology.org/C92-2082/
- Originally worked using specific types of grammatical constituents, but modern versions also extend it using named entity constraints

Example Hearst Patterns

- Identifying hypernyms (NP_H)
  - NP {, NP}* {,} (and|or) other NP_H → temples, treasuries, and other important **civic buildings**
  - NP_H such as {NP,}* {(or|and)} NP → **red algae** such as Gelidium
  - such NP_H as {NP,}* {(or|and)} NP → such **authors** as Herrick, Goldsmith, and Shakespeare
  - NP_H {,} including {NP,}* {(or|and)} NP → **common-law countries**, including Canada and England
  - NP_H {,} especially {NP,}* {(or|and)} NP → **European countries**, especially France, England, and Spain

Extended Hearst Patterns with Named Entity Constraints

- **PER**, **POSITION** of **ORG** → **George Marshall**, **Secretary of State** of **the United States**
- **PER** (be)? ((named)|(appointed)) Prep? **ORG** **POSITION** → **George Marshall** was named **US** **Secretary of State**

Advantages and Disadvantages of Hearst Patterns

Advantages:

- High precision
- Can be tailored to specific domains

Disadvantages:

- Tend to have low-recall
- Require substantial human effort for pattern creation

Supervised Relation Extraction

- Typically requires two steps:
  1. Find all possible pairs of named entities in the text (typically restricted to the same sentence)
  2. Classify the relation for each pair (either non-existent or one of the allowable relation types)
- Optional filtering step: First make a binary decision regarding whether a given pair of entities are related in any way (and then only classify relations for those that are expected to be related)

Supervised Relation Extraction Algorithm

```
function FindRelations(words) returns relations

      relations ← []
      entities ← FindEntities(words)
      forall entity pairs {e1, e2} in entities do
            if Related?(e1, e2)
                  relations ← relations + ClassifyRelation(e1, e2)
      return relations
```

Feature-Based Implementations

- Feature-based implementations of this algorithm (e.g., using logistic regression for relation classification) may use:
  - **Word features:** Unigrams and bigrams related to specific mentions of the entities and the words around them
  - **Named entity features:** Named entity types and high-level categorizations of those types, and number of entities between these two entities
  - **Syntactic structure features:** Constituent and dependency paths between the entities

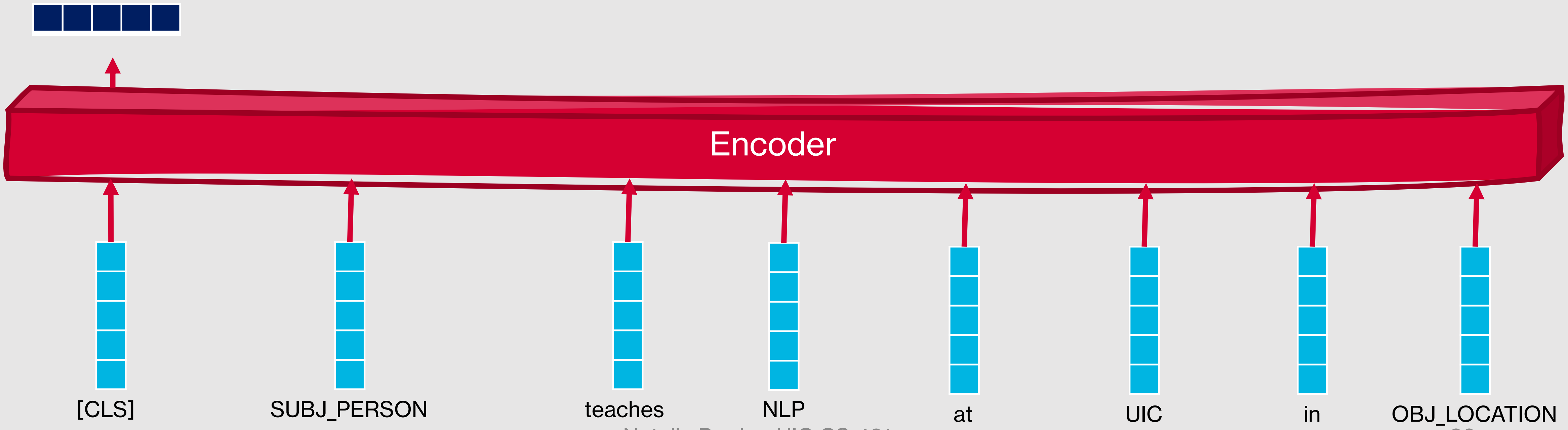
Neural Implementations

- Create a partially **delexicalized** version of the input by replacing the entities being classified with their named entity tags
- Finetune a pretrained model to predict the correct relation for the entities using the [CLS] token
  - Ideally, the base model should be pretrained on tasks that do not specify a sequence [SEP] token

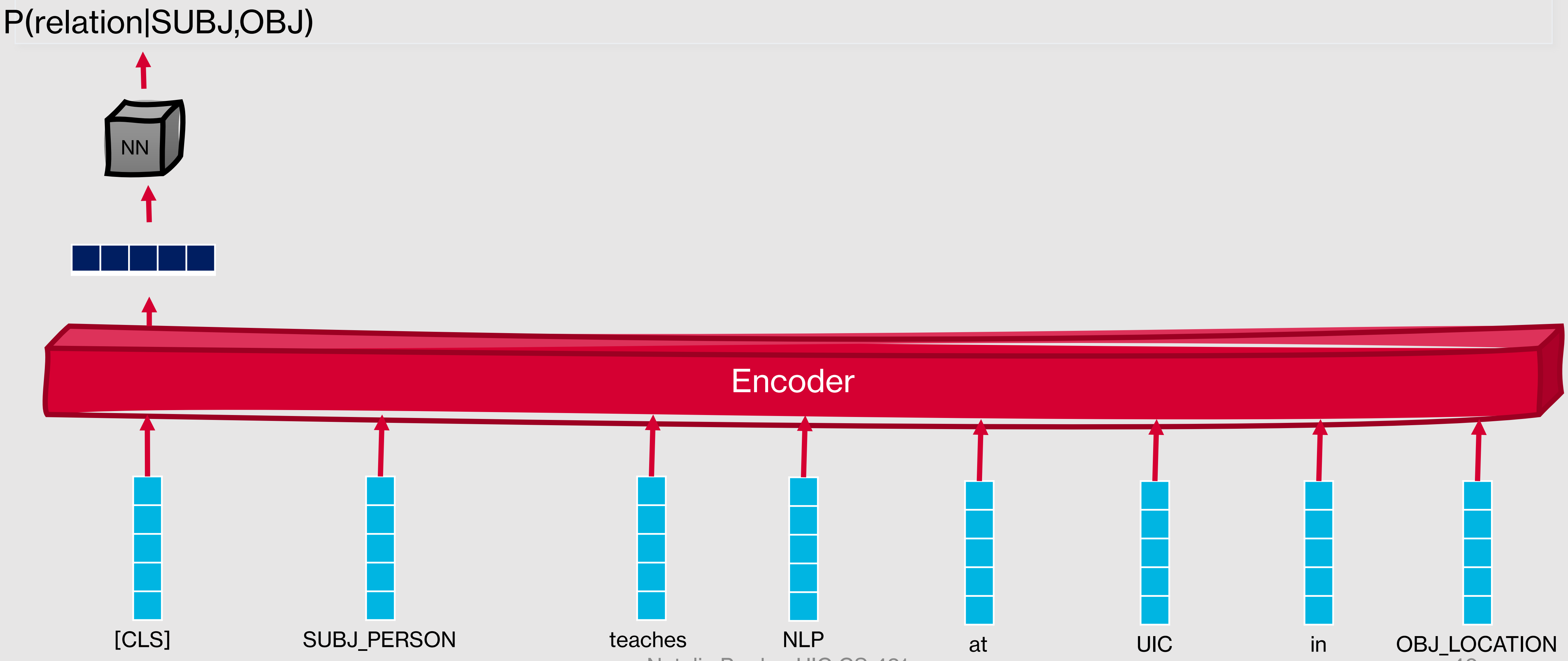
Example Neural Relation Extraction

Example Neural Relation Extraction

Example Neural Relation Extraction



Example Neural Relation Extraction



Advantages and Disadvantages of Supervised Relation Extraction

Advantages:

- Given sufficient training data and test data from a similar distribution, generally results in strong performance

Disadvantages:

- Expensive
- Tends not to generalize well to out-of-domain text

Semi-Supervised Relation Extraction

- Since building high-quality relation datasets is expensive and time-consuming, there has been substantial interest in **semi-supervised** and **unsupervised** approaches for this task
- Semi-supervised learning requires a small number of high-quality **seed examples**, such as:
  - Hearst patterns
  - Seed tuples

Bootstrapping Algorithm

- One way to perform semi-supervised learning is by **bootstrapping** a classifier
  - Take the seed examples and find data matching those samples in some other data source
  - Extract and generalize the context in those new samples to learn new patterns
  - Repeat
- For relation extraction, the seed examples would be seed pairs of entities

Bootstrapping Algorithm

```
function Bootstrap(Relation R) returns new relation tuples

        tuples ← Gather a set of seed tuples that have relation R
        iterate
                sentences ← Find sentences that contain entities in tuples
                patterns ← Generalize the context between and around entities in sentences
                newpairs ← Use patterns to identify more tuples
                newpairs ← newpairs with high confidence
                tuples ← tuples + newpairs
        return tuples
```

Bootstrapping in Practice

Task: Create a list of airline/hub pairs from free text.

Seed Fact:
hub(Ryanair, Charleroi)

Bootstrapping in Practice

Task: Create a list of airline/hub pairs from free text.

Seed Fact:
hub(Ryanair, Charleroi)

Corpus search:
“Ryanair Charleroi hub”

Bootstrapping in Practice

Task: Create a list of airline/hub pairs from free text.

Seed Fact:
hub(Ryanair, Charleroi)

Corpus search:
“Ryanair Charleroi hub”

Results:
- Budget airline Ryanair, which uses Charleroi as a hub, scrapped all weekend flights out of the airport.
- All flights in an out of Ryanair’s hub at Charleroi airport were grounded on Friday…
- A spokesperson at Charleroi, a main hub for Ryanair, estimated that 8000 passengers had already been affected.

Bootstrapping in Practice

Task: Create a list of airline/hub pairs from free text.

Seed Fact:
hub(Ryanair, Charleroi)

Results:
- Budget airline Ryanair, which uses Charleroi as a hub, scrapped all weekend flights out of the airport.
- All flights in an out of Ryanair's hub at Charleroi airport were grounded on Friday…
- A spokesperson at Charleroi, a main hub for Ryanair, estimated that 8000 passengers had already been affected.

Pattern Extraction:
- / [ORG], which uses [LOC] as a hub /
- / [ORG]'s hub at [LOC] /
- / [LOC], a main hum for [ORG] /

Bootstrapping in Practice

Task: Create a list of airline/hub pairs from free text.

Seed Fact:
hub(Ryanair, Charleroi)

Patterns:
- / [ORG], which uses [LOC] as a hub /
- / [ORG]'s hub at [LOC] /
- / [LOC], a main hum for [ORG] /

Bootstrapping in Practice

Task: Create a list of airline/hub pairs from free text.

Seed Fact:
hub(Ryanair, Charleroi)

Patterns:
- / [ORG], which uses [LOC] as a hub /
- / [ORG]'s hub at [LOC] /
- / [LOC], a main hum for [ORG] /

New Facts:
- hub(United, O'Hare)
- hub(American Airlines, DFW)
- hub(Southwest, Dallas)

Semantic Drift

- With bootstrapping, an erroneous pattern can lead to erroneous facts being added to the data
- Erroneous facts can then in turn lead to more erroneous patterns being created
- This problematic cycle is known as **semantic drift**

How can we address semantic drift?

- Assign confidence values to new patterns and tuples!
- Confidence values for patterns should balance:
  - The pattern's performance with respect to the current set of tuples
    - hits(*p*): The set of tuples in the larger set *T* that *p* matches while searching the document collection *D*
  - The pattern's productivity (number of matches it produces)
    - finds(*p*): The total set of tuples that *p* finds in *D*
  - $\text{Confidence}(p) = \frac{|\text{hits}(p)|}{|\text{finds}(p)|} \log(|\text{finds}(p)|)$

Confidence Values for Tuples

- Combine the evidence supporting a tuple from all the patterns *P’* that match that tuple in *D*
- For example, the **noisy-or** technique assumes:
  - For a proposed tuple to be false, all of its supporting patterns must have been made in error
  - The sources of those patterns’ individual failures are independent
- This means that the probability of an individual pattern *p* failing could be framed as $1 - \text{Confidence}(p)$
- In turn, the probability of all supporting patterns for a tuple *t* being wrong is the product of their individual failure probabilities:
  - $\text{Confidence}(t) = 1 - \prod_{p \in P'} (1 - \text{Confidence}(p))$

Distant Supervision for Relation Extraction

- Another way to avoid expensive manual labeling of relation labels is to perform **distant supervision** with indirect sources of training data
- Combines advantages of bootstrapping and supervised learning by:
  - Using a large database to acquire many seed examples
  - Create many noisy pattern features from these examples
  - Combine these in a supervised classifier

Distant Supervision Algorithm

```
function DistantSupervision(Database D, Text T) returns relation classifier C
        observations = []
        foreach relation R in D
                foreach tuple (e1, e2) of entities with relation R
                        sentences ← Sentences in T that contain e1 and e2
                        f ← Frequent features in sentences
                        observations ← observations + new training tuple (e1, e2, f, R)
        C ← Train supervised classifier on observations
        return C
```

Distant Supervision in Practice

Task: Learn the place-of-birth relationship between people and their birth cities from free text.

Database Search: place-of-birth

DBpedia

Distant Supervision in Practice

Task: Learn the place-of-birth relationship between people and their birth cities from free text.

Database Search:
place-of-birth

Retrieved Seed Tuples:
<Edwin Hubble, Marshfield>
<Albert Einstein, Ulm>
(and many more!)

DBpedia

Distant Supervision in Practice

Task: Learn the place-of-birth relationship between people and their birth cities from free text.

Database Search: place-of-birth

Retrieved Seed Tuples:
<Edwin Hubble, Marshfield>
<Albert Einstein, Ulm>
(and many more!)

Sentences with two named entities that match a seed tuple:
Hubble was born in Marshfield....
Einstein, born (1879), Ulm....
Hubble's birthplace in Marshfield....
(and many more!)

Distant Supervision in Practice

Task: Learn the place-of-birth relationship between people and their birth cities from free text.

Database Search: place-of-birth

Retrieved Seed Tuples:
<Edwin Hubble, Marshfield>
<Albert Einstein, Ulm>
(and many more!)

Sentences with two named entities that match a seed tuple:
Hubble was born in Marshfield….
Einstein, born (1879), Ulm….
Hubble's birthplace in Marshfield….
(and many more!)

With this information, extract features to train the classification model!

Advantages and Disadvantages of Distant Supervision

Advantages:

- Allows for the use of a large amount of training data
- Allows for detailed pattern learning
- Prevents semantic drift
- Doesn't require labeled training data

Disadvantages:

- Tends to produce low-precision results
- Can only work when a large enough relevant database already exists

Unsupervised Relation Extraction

- Uses in situations with no labeled training data, such as when you are working with:
  - Low-resource domains
  - New domains
- Often referred to as **open information extraction (Open IE)**
- Relations are generally strings of words, rather than more formal relations

How does open information extraction work?

- Generally the following stages:
  1. Run a part-of-speech tagger and named entity recognizer over the sentence
  2. For each verb in that sentence, find the longest sequence of words w that starts with a verb and satisfies any specified syntactic and lexical constraints
  3. For each retrieved sequence of words, find the nearest noun phrase x prior to the sequence that is not a relative pronoun, wh-word, or existential "there"
  4. Likewise, find the nearest noun phrase y after the sequence
  5. Assign a confidence c to the relation $r = (x, w, y)$ using a confidence classifier

What syntactic and lexical constraints should be specified?

- Syntactic constraints:
  - The sequence should structurally be able to include nouns
- Lexical constraints:
  - Relations should occur with a sufficient number of distinct argument types (long, rare relation strings are undesirable)

Advantages and Disadvantages of Unsupervised Relation Extraction

Advantages:

- Can handle many relations without having to specify them in advance
- No labeled training data or seed examples are necessary

Disadvantages:

- All recognized relation strings need to be mapped to canonical form prior to storage in databases or knowledge graphs
- Current methods focused on verb-based relations, so nominal relations may not be adequately captured

Evaluating Methods for Relation Extraction

- Supervised relation extraction systems can be evaluated using standard NLP metrics (e.g., precision, recall, and F-measure)
- Semi-supervised and unsupervised relation extraction systems are more challenging to evaluate since there is typically not an existing gold standard for comparison

Evaluating Semi-Supervised and Unsupervised Systems

- **Approximate precision** by randomly sampling output relations and asking a human to manually evaluate them
  - $\hat{P} = \frac{\text{\# correctly extracted relation tuples in the sample}}{\text{\# extracted relation tuples in the sample}}$
- **Approximate recall** by computing precision at different sample sizes
  - Precision for most-confident 1000 new relations
  - Precision for most-confident 10,000 new relations
  - And so forth!

Summary: Relation and Event Extraction

- Natural language can be converted to **structured representations** using information extraction techniques
- **Named entity recognition** identifies the specific entities that participate in structured relationships or events
- **Relations** can be extracted using a variety of approaches
- **Hearst patterns** are specialized rules for extracting relations
- Supervised learning can be used to train feature-based or neural approaches to relation extraction
- **Semi-supervised** learning or **distant supervision** can be used to extract relations without using a large labeled training set
- **Open information extraction** uses unsupervised methods to extract relations

This Week's Topics

Named Entity Recognition
Relation Extraction

Tuesday

Thursday

Event Extraction
Temporal Logic
Temporal Analysis

Event Extraction

- **The process of finding events in which entities in the text participate**

What is an event mention?

Any expression denoting an event or state that can be assigned to a particular point or interval in time

Most event mentions in English correspond to verbs (and most English verbs introduce events) although this is not a requirement

[$_{EVENT}$ Citing] high fuel prices, United Airlines **[$_{EVENT}$ said]** Friday it has **[$_{EVENT}$ increased]** fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately **[$_{EVENT}$ matched]** **[$_{EVENT}$ the move]**, spokesman Tim Wagner **[$_{EVENT}$ said]**. United, a unit of UAL Corp., **[$_{EVENT}$ said]** **[$_{EVENT}$ the increase]** took effect Thursday and **[$_{EVENT}$ applies]** to most routes where it **[$_{EVENT}$ competes]** against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

How are events extracted?

- In general, event extractors are designed to classify events based on their aspectual and temporal properties
- Different event tagsets exist, but may include:
  - Actions
  - States
  - Reporting events
  - Perception events

Event Extraction Datasets

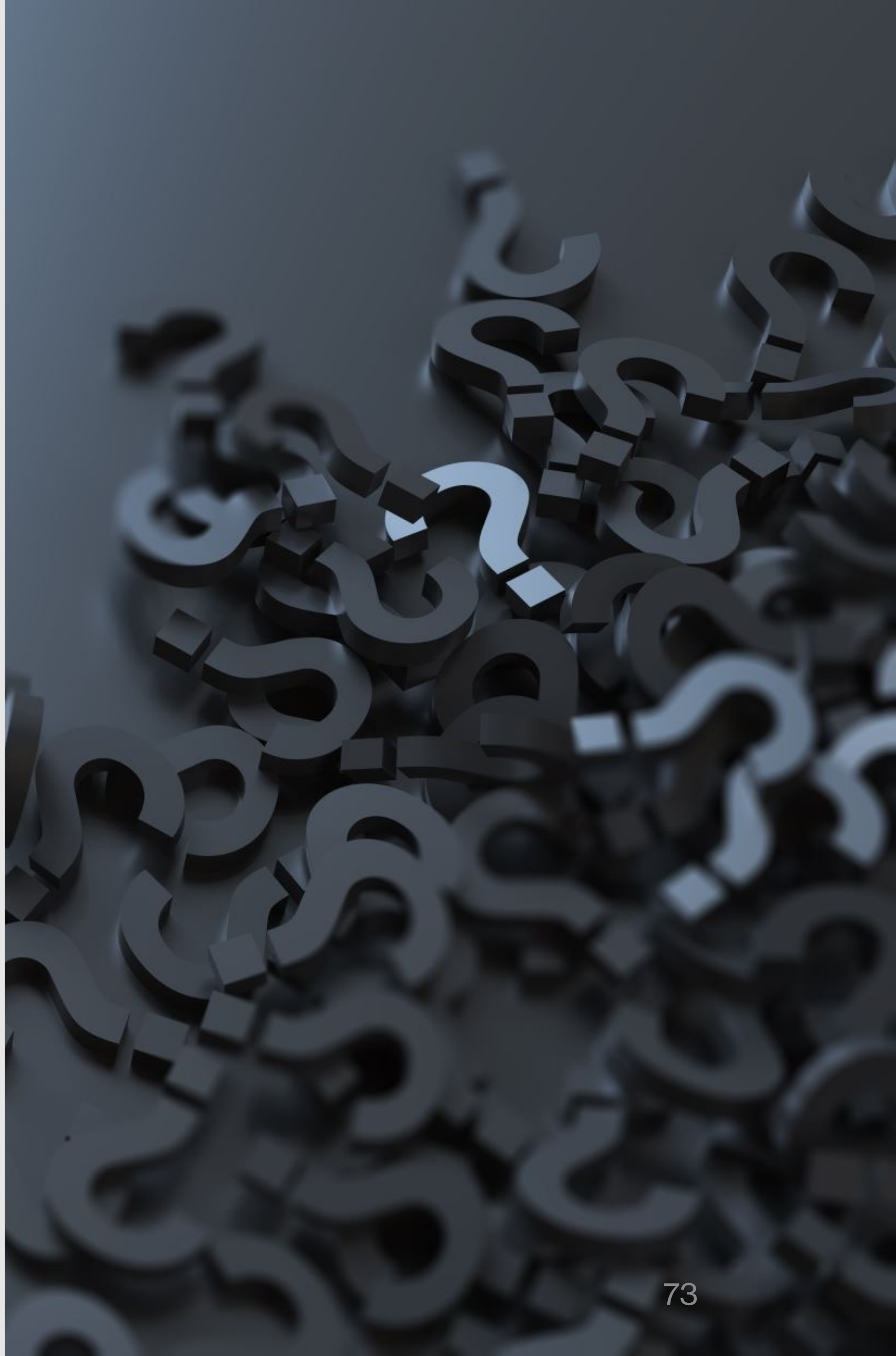
- Typically also include annotations for temporal and aspectual information
- TempEval Shared Tasks:
  - https://aclweb.org/aclwiki/Temporal_Information_Extraction_(State_of_the_art)
  - https://alt.qcri.org/semeval2017/task12/

[EVENT **Citing**] high fuel prices, United Airlines [EVENT **said**] Friday it has [EVENT **increased**] fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately [EVENT **matched**] [EVENT **the move**], spokesman Tim Wagner [EVENT **said**]. United, a unit of UAL Corp., [EVENT **said**] [EVENT **the increase**] took effect Thursday and [EVENT **applies**] to most routes where it [EVENT **competes**] against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

said(class=reporting, tense=past, aspect=perfective)

Event Extraction Approaches

- Generally framed as a supervised sequence labeling problem
- BIO tags are used to assign event classes and attributes
- Neural encoder-based models or feature-based models can be used, depending on dataset size and preference



Scripts and Templates

- Many events correspond to fairly common, stereotypical situations in the world
- **Scripts** are prototypical sequences of sub-events, participants, and their roles
- The expectations provided by scripts can facilitate:
  - Entity classification
  - Assignment of entities into roles and relations
  - Inference of unstated entities or sub-events
- We can represent scripts using simple **templates**

Template Filling

- Templates have fixed sets of **slots**
- Each slot takes **slot-fillers** as values belonging to particular classes
- The task of template filling is to:
  - Find documents that invoke known scripts
  - Fill the slots in the associated templates with fillers extracted from the text
- Slot fillers may be text segments extracted directly from the text, or concepts that have been inferred from text elements through additional preprocessing

Filled Template

Citing high fuel prices, United Airlines said Friday it has increased fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately matched the move, spokesman Tim Wagner said. United, a unit of UAL Corp., said the increase took effect Thursday and applies to most routes where it competes against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

Fair-Raise Attempt:

- Lead Airline: United Airlines
- Amount: $6
- Effective Date: 2006-10-26
- Follower: American Airlines

How do we perform template filling?

- Goal: Create one template for each event in the input, filling in the slots with text spans
- Requires two separate tasks:
  - **Template recognition:** Determines whether a given template is present in the sentence
  - **Role-filler extraction:** Detects each role for a template

What if we find multiple text segments that can fill the same slot?

- This is fine (if role-filler extraction is performed correctly, they likely refer to the same entity!)
- We can resolve this using **coreference resolution**

Citing high fuel prices, United Airlines said Friday it has increased fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately matched the move, spokesman Tim Wagner said. United, a unit of UAL Corp., said the increase took effect Thursday and applies to most routes where it competes against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

Fair-Raise Attempt:

- Lead Airline: United Airlines
- Amount: $6
- Effective Date: 2006-10-26
- Follower: American Airlines

What if we want to have hierarchically linked templates to encompass all information in a text?

- Currently, no easy way to do this with supervised learning approaches
- Can be addressed using cascades of finite state transducers (but requires extensive time for manual development!)

This Week's Topics

Named Entity Recognition
Relation Extraction

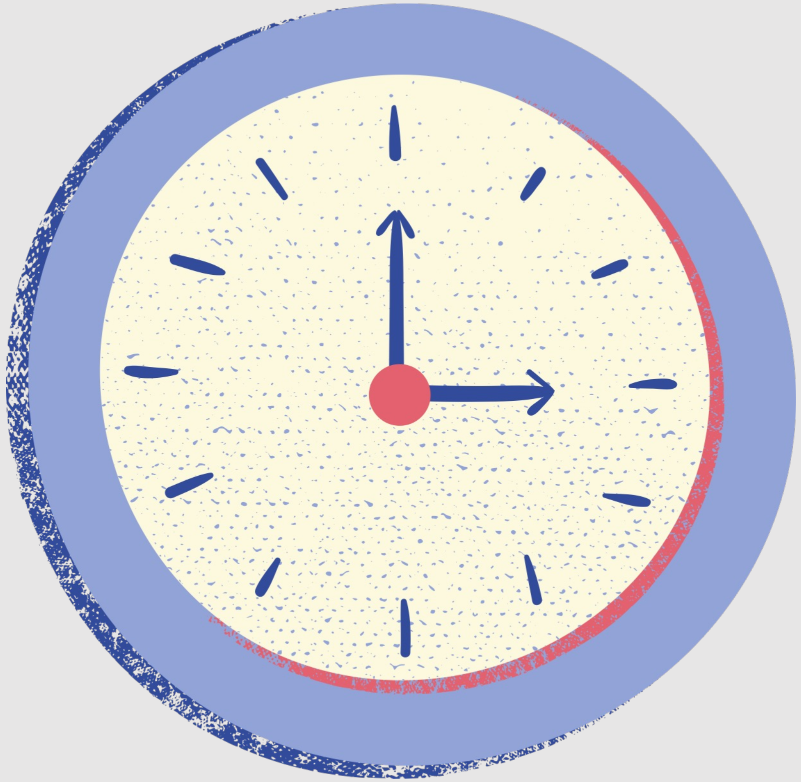
Tuesday

Thursday

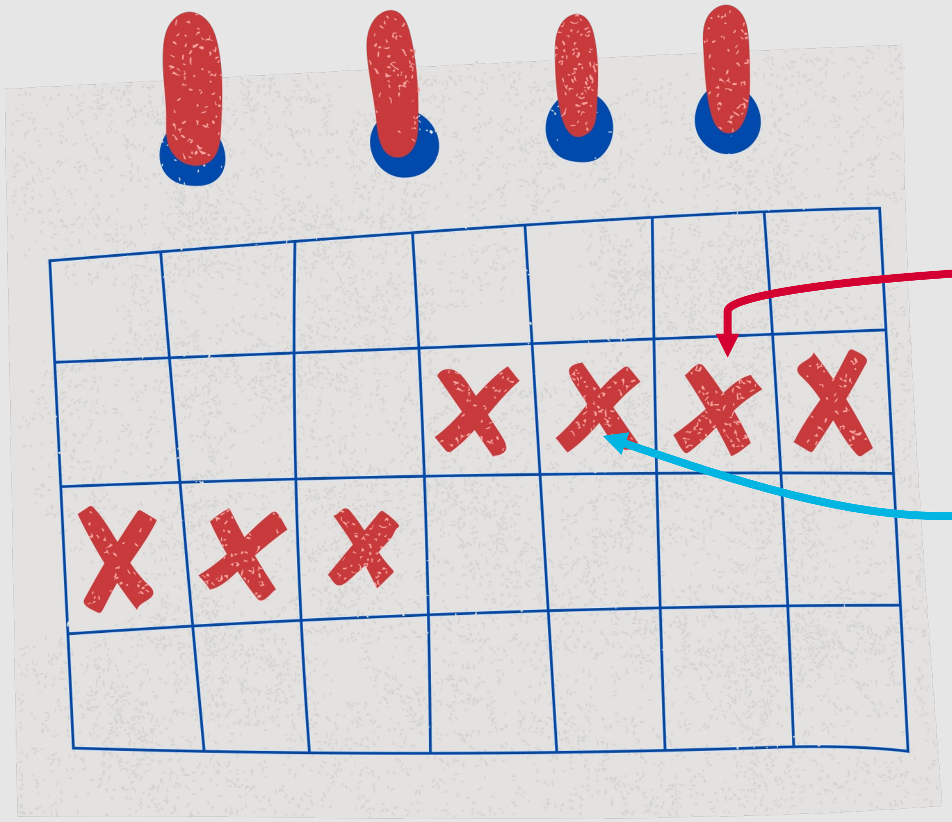
Event Extraction
Temporal Logic
Temporal Analysis

Time and Temporality

- Events are situated within time, and they can also relate to one another temporally
  - Events happen at particular dates and times
  - Events can occur before, after, or simultaneously with one another



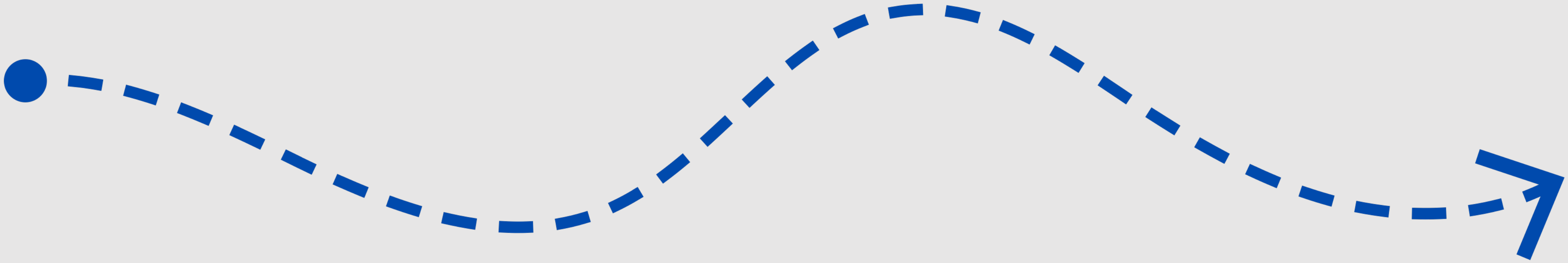
Temporal Expressions



- Temporal expressions allow us to understand how events are situated within time
- These expressions may be explicit statements of date or time:
  - Project Part 2 is due on Friday at 12 p.m.
- Or they may be in reference to other expressions or events:
  - 26 hours from now
- In an ideal case, we would be able to build a timeline of events from a segment of text by normalizing those events to calendar dates and times and placing them on a timeline …but this is very challenging!

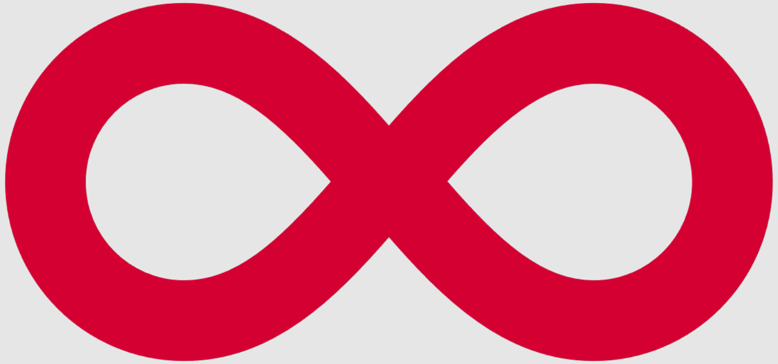
Representing Time

- Although there are numerous philosophical theories of time, the most straightforward theory holds that:
  - **Time flows forward**
  - **Events are associated with points or intervals in time**
- Assuming this to be true, we can order events by situating them on a timeline, with one event preceding another if time flows from the first event to the second
- If we situate the current moment in time along this timeline, we build notions of past, present, and future



Temporal Logic

- **Temporal logic** is a formal way to represent temporal information
- Although many schemes can be used to represent temporal logic, a straightforward way to do so involves staying within the first-order logic framework



How do we adapt first-order logic to represent temporality?

- Simple first-order logic representations focus on meaning irrespective of time

$$\exists e\ \mathrm{Arriving}(e) \wedge \mathrm{Arriver}(e, \mathrm{Speaker}) \wedge \mathrm{Destination}(e, \mathrm{Chicago})$$

I arrived in Chicago.

I am arriving in Chicago.

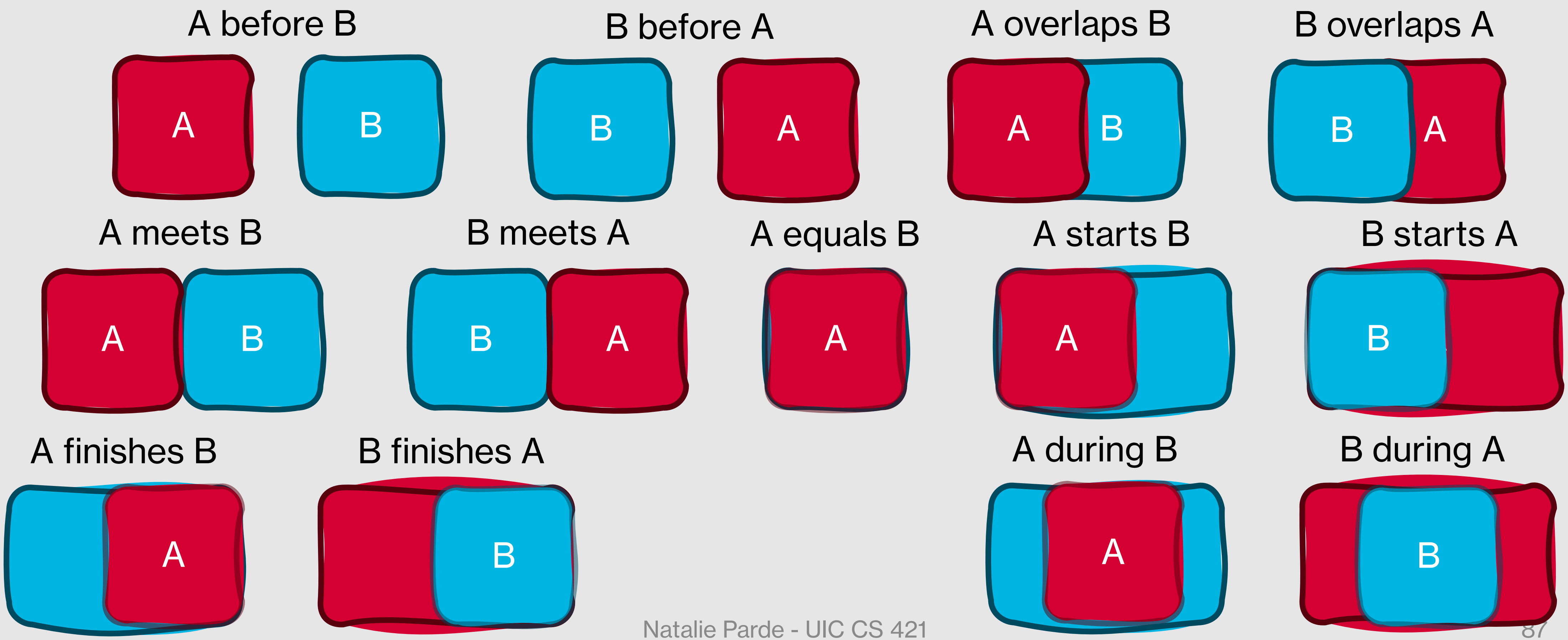
I will arrive in Chicago.

How do we adapt first-order logic to represent temporality?

- Event variables to the rescue!
  - We can use specify temporal information based on the event's order with respect to other events using attributes associated with the event variable
- **Interval algebra** is one framework developed by James F. Allen that is used to discuss temporal ordering relationships
  - All events and time expressions are modeled as intervals (no time points!), which can be long or very short

Allen Relations

- There are 13 relations that can hold between intervals A and B in interval algebra



How do we include interval algebra in our first-order logic model?

1

Add a temporal variable representing the interval corresponding to the event

2

Add a very small interval corresponding to the current time (“Now”)

3

Add temporal predicates relating the event to the current time as indicated by event verb tense

Interval Algebra in Practice

Original First-Order Logic

$$\exists e\ \mathrm{Arriving}(e) \wedge \mathrm{Arriver}(e, \mathrm{Speaker}) \wedge \mathrm{Destination}(e, \mathrm{Chicago})$$

I arrived in Chicago.

I am arriving in Chicago.

I will arrive in Chicago.

Interval Algebra in Practice

Updated First-Order Logic

Original First-Order Logic

$\exists e \text{ Arriving}(e) \wedge \text{Arriver}(e, \text{Speaker}) \wedge \text{Destination}(e, \text{Chicago})$

I arrived in Chicago.

I am arriving in Chicago.

I will arrive in Chicago.

$\exists e, i \text{ Arriving}(e) \wedge \text{Arriver}(e, \text{Speaker}) \wedge \text{Destination}(e, \text{Chicago}) \wedge \text{IntervalOf}(e, i) \wedge \text{Before}(i, \text{Now})$

Interval Algebra in Practice

Updated First-Order Logic

Original First-Order Logic

$\exists e$ Arriving(e) ∧ Arriver$(e,$ Speaker$)$ ∧ Destination$(e,$ Chicago$)$

I arrived in Chicago.

I am arriving in Chicago.

I will arrive in Chicago.

$\exists e, i$ Arriving(e) ∧ Arriver$(e,$ Speaker$)$ ∧ Destination$(e,$ Chicago$)$ ∧ IntervalOf(e, i) ∧ Before$(i,$ Now$)$

$\exists e, i$ Arriving(e) ∧ Arriver$(e,$ Speaker$)$ ∧ Destination$(e,$ Chicago$)$ ∧ IntervalOf(e, i) ∧ During$(i,$ Now$)$

Interval Algebra in Practice

Updated First-Order Logic

Original First-Order Logic

$\exists e\ \mathrm{Arriving}(e) \wedge \mathrm{Arriver}(e, \mathrm{Speaker}) \wedge \mathrm{Destination}(e, \mathrm{Chicago})$

I arrived in Chicago.

I am arriving in Chicago.

I will arrive in Chicago.

$\exists e, i\ \mathrm{Arriving}(e) \wedge \mathrm{Arriver}(e, \mathrm{Speaker}) \wedge \mathrm{Destination}(e, \mathrm{Chicago}) \wedge \mathrm{IntervalOf}(e, i) \wedge \mathrm{Before}(i, \mathrm{Now})$

$\exists e, i\ \mathrm{Arriving}(e) \wedge \mathrm{Arriver}(e, \mathrm{Speaker}) \wedge \mathrm{Destination}(e, \mathrm{Chicago}) \wedge \mathrm{IntervalOf}(e, i) \wedge \mathrm{During}(i, \mathrm{Now})$

$\exists e, i\ \mathrm{Arriving}(e) \wedge \mathrm{Arriver}(e, \mathrm{Speaker}) \wedge \mathrm{Destination}(e, \mathrm{Chicago}) \wedge \mathrm{IntervalOf}(e, i) \wedge \mathrm{After}(i, \mathrm{Now})$

Temporal Ambiguity

- Although relating verb tenses with points in time may initially seem simple, there are many opportunities for ambiguity to arise
  - *Okay, we fly from Chicago to Burlington at noon.*
    - Present tense verb indicating future event
  - *Flight 1902 arrived late.*
    - Past tense verb.
  - *Flight 1902 had arrived late.*
    - Past tense verb with respect to some unnamed event.

Reichenbach's Reference Point

- To address this, Hans Reichenbach introduced the notion of reference points that are separate from the utterance time and event time
  - Reichenbach, Hans (1947). *Elements of Symbolic Logic*. New York: Macmillan & Co.
- This means that the current moment in time is not necessarily equated with the time of the utterance and/or used as a reference point for the event occurrence

Reichenbach's Reference Point

When I got off the blue line, I taught CS 421.

Happened before the utterance

When I got off the blue line, I had taught CS 421.

Reichenbach's Reference Point

Reference point

When I got off the blue line, I taught CS 421.

Happened before the utterance

Reference point

When I got off the blue line, I had taught CS 421.

Reichenbach's Reference Point

Reference point

When I got off the blue line, I taught CS 421.

Happened before the utterance

Happened *after* the speaker got off the blue line

Reference point

When I got off the blue line, I had taught CS 421.

Happened *prior to* when the speaker got off the blue line

Reichenbach's Reference Point

Reference point

When I got off the blue line, I taught CS 421.

Happened before the utterance

Happened *after* the speaker got off the blue line

- Reference point precedes the teaching event

Reference point

When I got off the blue line, I had taught CS 421.

Happened *prior to* when the speaker got off the blue line

- Teaching event precedes the reference point

Temporal Information Beyond Tense

- Although Reichenbach's approach is clearly illustrated using English verb tenses, there are also many other ways that languages can convey temporal information
  - CS 421 is held *in the morning*
  - Assignments are due at *noon* and the weekend begins *afterwards*!

Time and Metaphor

- Many languages (including English) frequently rely on **metaphors** to express temporality
- Most frequent in English: <TIME> is <SPACE>
  - *In* the morning
  - *Around* noon
  - Midnight is *near*
- This facilitates understanding of a complex topic for humans, but may create additional complexities when processing temporal expressions computationally

Aspect

- **Aspect** defines categories of events or states based on their temporal structure
- Events may be:
  - Ongoing or complete
  - At a specific point or over an interval of time

Aspectual Distinctions

- **Events** involve change, whereas **states** do not
- **Stative expressions** indicate the state or property of an event participant at a given point in time

I **like** CS 421.

I **need** to finish my research project.

I **want** to take CS 521.

- Event participant is experiencing something at a specific point in time
- Event doesn't involve internal change over time

Aspectual Distinctions

Activity, **achievement**, and **accomplishment** expressions all involve some form of change

Activity expressions describe events undertaken by a participant that occur over a span of time and have no particular end point

Accomplishment expressions describe events that take place over time but have a natural end point and result in a particular state

Achievement expressions describe events that happen in an instant and result in a particular state, without conceptualizing the process or activity leading up to that state

Activity, Accomplishment, and Achievement Expressions

Activity	Accomplishment	Achievement
Mohammad lives in Chicago.	Ankit ordered some ice cream.	Eli found the missing assignment.
Natalie works at UIC.	Usman traveled to New Orleans.	Pardis finished the paper.

Activity

- Event participant is or has engaged in the activity for some period of time
- No specification that the activity might have stopped

Accomplishment

- Event occurs over some period of time
- Event ends when the intended state is accomplished

Achievement

- Event occurs in an instant
- Event results in a new state

Event expressions can easily be shifted to other aspectual classes!

- Surrounding context guides the interpretation of the event

Mina sailed.

Activity

Mina sailed to Paris.

Accomplishment

This Week's Topics

Tuesday

Named Entity Recognition

Relation Extraction

Thursday

Event Extraction

Temporal Logic

Temporal Analysis

There are numerous temporal analysis tasks that we may want to perform.

- We can train systems to perform these tasks using temporally annotated datasets
- Popular dataset for this domain: **TimeBank**
  - American English text annotated using **TimeML**, a markup language based on interval algebra
  - TimeML includes three types of objects:
    - **Events** (representing events and states)
    - **Times** (representing temporal expressions like dates)
    - **Links** (representing relationships between events and times)
      - **TLinks** describe Allen relations
      - **ALinks** describe aspectual relationships
      - **SLinks** describe subordination relationships involving modality, evidentiality, and factuality
  - More details: https://timeml.github.io/site/timebank/documentation-1.2.html

TimeBank

Delta Air Lines earnings soared 33% to a record in the fiscal first quarter, bucking the industry trend toward declining profits.

```
<TIMEX3 tid=“t57” type=“DATE” value=“1989-10-26” functionInDocument=“CREATION_TIME”>
10/26/89 </TIMEX3>

Delta Air Lines earnings <EVENT eid=“e1” class=“OCCURRENCE”> soared </EVENT> 33% to a
record in <TIMEX tid==“t58” type=“DATE” value=“1989-Q1” anchorTimeID=“t57”> the fiscal
first quarter </TIMEX3>, <EVENT eid=“e3” class=“OCCURRENCE”>bucking</EVENT> the
industry trend toward <EVENT eid=“e4” class=“OCCURRENCE”>declining</EVENT> profits.
```

TimeBank

Delta Air Lines earnings soared 33% to a record in the fiscal first quarter, bucking the industry trend toward declining profits.

```
<TIMEX3 tid="t57" type="DATE" value="1989-10-26" functionInDocument="CREATION_TIME">
10/26/89 </TIMEX3>

Delta Air Lines earnings <EVENT eid="e1" class="OCCURRENCE"> soared </EVENT> 33% to a
record in <TIMEX tid="t58" type="DATE" value="1989-Q1" anchorTimeID="t57"> the fiscal
first quarter </TIMEX3>, <EVENT eid="e3" class="OCCURRENCE">bucking</EVENT> the
industry trend toward <EVENT eid="e4" class="OCCURRENCE">declining</EVENT> profits.
```

- Three events

TimeBank

Delta Air Lines earnings soared 33% to a record in the fiscal first quarter, bucking the industry trend toward declining profits.

- Three events
- Two temporal expressions

```
<TIMEX3 tid="t57" type="DATE" value="1989-10-26" functionInDocument="CREATION_TIME">
10/26/89 </TIMEX3>

Delta Air Lines earnings <EVENT eid="e1" class="OCCURRENCE"> soared </EVENT> 33% to a
record in <TIMEX tid="t58" type="DATE" value="1989-Q1" anchorTimeID="t57"> the fiscal
first quarter </TIMEX3>, <EVENT eid="e3" class="OCCURRENCE">bucking</EVENT> the
industry trend toward <EVENT eid="e4" class="OCCURRENCE">declining</EVENT> profits.
```

TimeBank

Delta Air Lines earnings soared 33% to a record in the fiscal first quarter, bucking the industry trend toward declining profits.

- Three events
- Two temporal expressions
- Four temporal links capturing Allen relations

```
<TIMEX3 tid="t57" type="DATE" value="1989-10-26" functionInDocument="CREATION_TIME">
10/26/89 </TIMEX3>

Delta Air Lines earnings <EVENT eid="e1" class="OCCURRENCE"> soared </EVENT> 33% to a
record in <TIMEX3 tid="t58" type="DATE" value="1989-Q1" anchorTimeID="t57"> the fiscal
first quarter </TIMEX3>, <EVENT eid="e3" class="OCCURRENCE">bucking</EVENT> the industry
trend toward <EVENT eid="e4" class="OCCURRENCE">declining</EVENT> profits.

<TLINK lid="l1" relType="IS_INCLUDED" eventInstanceID="e1" relatedToTime="t58" />
<TLINK lid="l2" relType="BEFORE" eventInstanceID="e1" relatedToTime="t57" />
<TLINK lid="l3" relType="SIMULTANEOUS" eventInstanceID="e1" relatedToEventInstance="e3" />
<TLINK lid="l4" relType="IS_INCLUDED" eventInstanceID="e1" relatedToEventInstance="e4" />
```

How can we automatically recognize and interpret the types of information present in TimeBank?

- Automated **temporal analysis** involves three common steps:
  - **Extracting temporal expressions**
  - **Normalizing these expressions** by converting them to a standard format
  - **Linking events to times** and **extracting time graphs and timelines** from the text

Extracting Temporal Expressions

- Temporal expressions may refer to absolute points in time, relative times, durations, or combinations thereof
- **Absolute temporal expressions** can be mapped directly to calendar dates, times of day, or both
- **Relative temporal expressions** can be mapped to particular times through some other reference point
  - *A week from last Tuesday*
- **Durations** can be mapped to spans of time at varying levels of granularity
  - Seconds, minutes, days, weeks, years, etc.

Example Temporal Expressions

Absolute	Relative	Duration
October 26, 2023	Yesterday	Two hours
The summer of 2023	Next semester	Three days
9:30 a.m.	Two weeks from yesterday	Four years
The fall semester in 2023	Last year	Two semesters

How do we recognize temporal expressions?

- Temporal expressions are grammatical constructions that often have temporal **lexical triggers** as their heads
- We can build **rule-based methods** to search for these lexical triggers and subsequently expand them to recognize larger and larger projections of them
- We can also use BIO tags to extract temporal expressions, framing this process as a **supervised span labeling task**

```
/(\d+)[-\s]($TEUnits)(s)?([-\s]old)?/
```

→ Duration since reference

Challenges in Extracting Temporal Expressions

- In general, approaches for extracting temporal expressions are heavily lexicalized
- This reliance on lexical triggers may also lead to false positives being tagged as temporal expressions
  - I’m reading *1984* by George Orwell
  - “*Wednesday Morning, 3 A.M.*” was Simon & Garfunkel’s first album
- To avoid this issue, it is important to consider broader context as well

Temporal Normalization

- Once we recognize temporal expressions, we can **normalize** them by mapping them to specific durations or points in time
- Normalized times are represented using the **ISO 8601** standard for encoding temporal values
  - **Date:** YYYY-MM-DD
  - **Weeks:** YYYY-W*nn*, with weeks numbered from 01-53 in the year (W001 has the first Thursday of the year)
    - Note: ISO weeks begin on Monday
  - **Durations:** P*nx*, where *n* denotes the length as an integer and *x* represents the temporal unit of measurement (e.g., P3Y="three years" or P2D="two days")

Sample ISO Patterns for Times and Durations

Unit	Pattern	Sample Value
Fully specified dates	YYYY-MM-DD	2023-10-26
Weeks	YYYY-Wnn	2023-W10
Weekends	PnWE	P1WE
24-hour clock times	HH:MM:SS	09:30:00
Dates and times	YYYY-MM-DDTHH:MM:SS	2023-10-26T09:30:00

Additional Examples: https://en.wikipedia.org/wiki/ISO_8601

For our previous example….

Delta Air Lines earnings soared 33% to a record in the fiscal first quarter, bucking the industry trend toward declining profits.

```
<TIMEX3 tid="t57" type="DATE" value="1989-10-26" functionInDocument="CREATION_TIME">
10/26/89 </TIMEX3>

Delta Air Lines earnings <EVENT eid="e1" class="OCCURRENCE"> soared </EVENT> 33% to a
record in <TIMEX3 tid="t58" type="DATE" value="1989-Q1" anchorTimeID="t57"> the fiscal
first quarter </TIMEX3>, <EVENT eid="e3" class="OCCURRENCE">bucking</EVENT> the industry
trend toward <EVENT eid="e4" class="OCCURRENCE">declining</EVENT> profits.

<TLINK lid="l1" relType="IS_INCLUDED" eventInstanceID="e1" relatedToTime="t58" />
<TLINK lid="l2" relType="BEFORE" eventInstanceID="e1" relatedToTime="t57" />
<TLINK lid="l3" relType="SIMULTANEOUS" eventInstanceID="e1" relatedToEventInstance="e3" />
<TLINK lid="l4" relType="IS_INCLUDED" eventInstanceID="e1" relatedToEventInstance="e4" />
```

How can we perform temporal normalization?

- Temporal normalization is often handled using rule-based approaches that match patterns associated with different types of temporal expressions

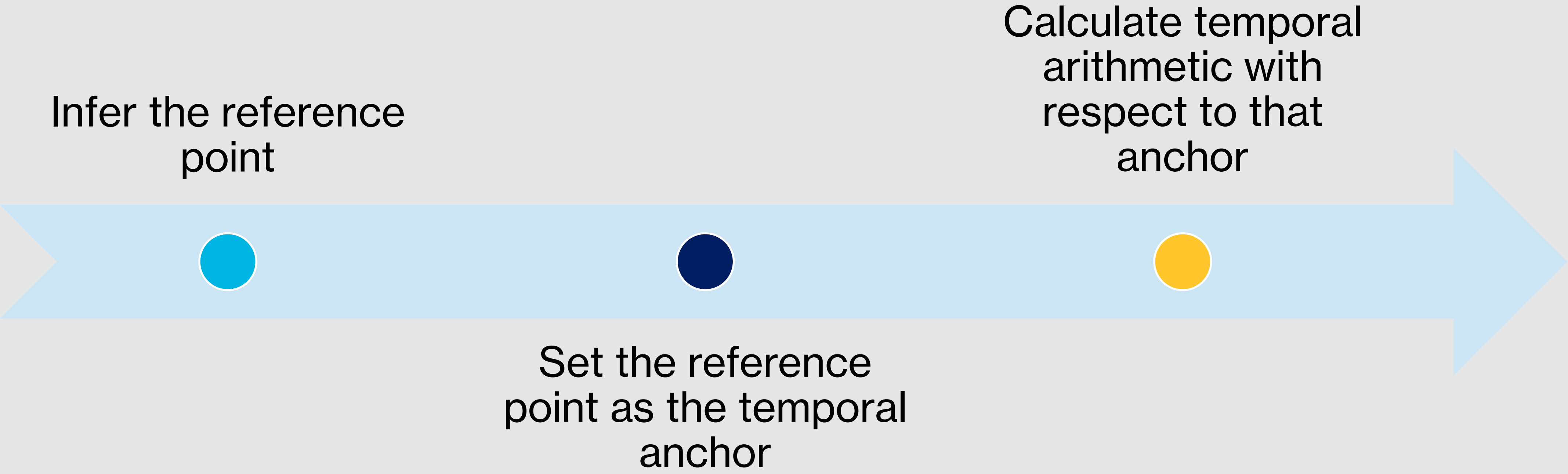
```
Pattern: /(\d+)[-\s]($TEUnits)(s)?([-\s]old)?/
Result:  Duration($1, $2)
```

- This is challenging because:
  - Fully qualified temporal expressions tend to be rare
  - Most expressions in text instead do not explicitly state a **temporal anchor**

Temporal Anchors

- Temporal anchors serve as the base point across which temporal expressions are normalized
  - "Today" = temporal anchor
  - "Yesterday" = temporal anchor – 1
  - "Tomorrow" = temporal anchor + 1
- **Without an explicit temporal anchor, we must infer the temporal anchor implicitly** (generally using domain-specific heuristics)
  - News articles: The temporal anchor can generally assumed to be the dateline for the news article

Relative Temporal Expressions

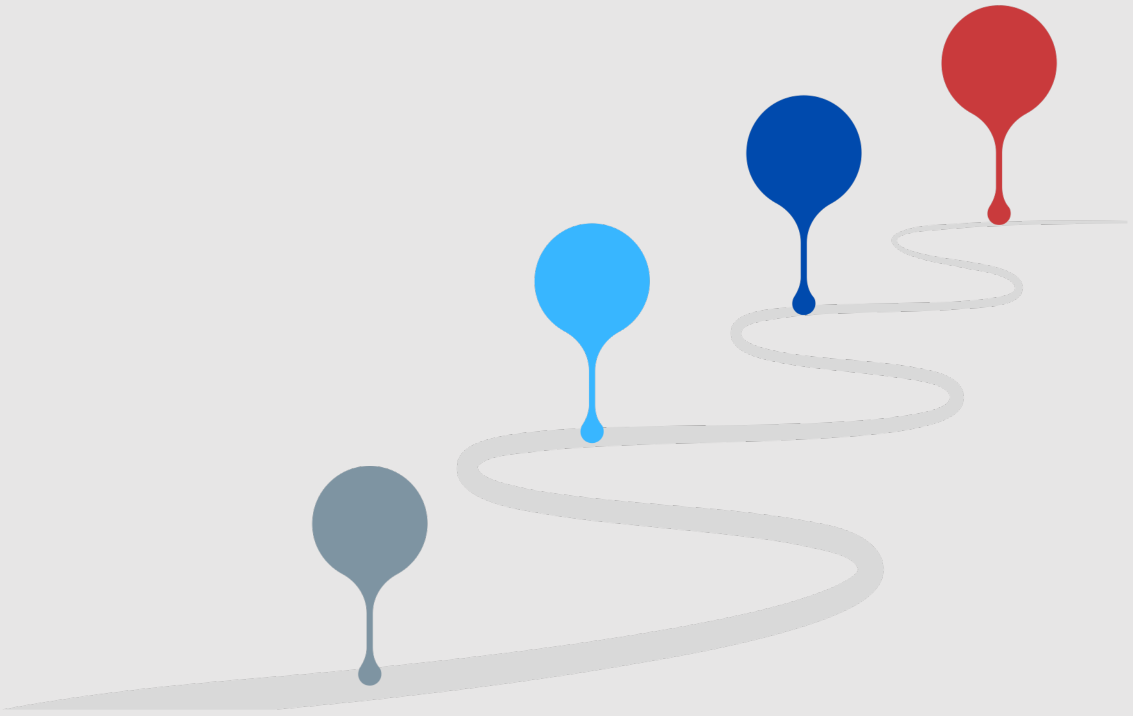


Normalization Ambiguities

- Language is noisy …temporal ambiguities will arise!
- Common temporal ambiguities:
  - Next
  - Last
- In general, ambiguities are handled using commonsense and domain-specific heuristics
  - "Next" → Tends to skip the nearest unit as that unit grows closer (e.g., "next Friday" may mean the nearest future Friday when it is stated on Saturday, but is likelier to mean the following future Friday when it is stated on Thursday)

Temporal Ordering of Events

- The broader goal of extracting and normalizing temporal expressions is to be able to situate them along a timeline
- One step toward achieving this goal is to perform **temporal ordering**, such that extracted events are ordered based on the resolved temporal expressions within the text
- This task can be framed similarly to relation classification



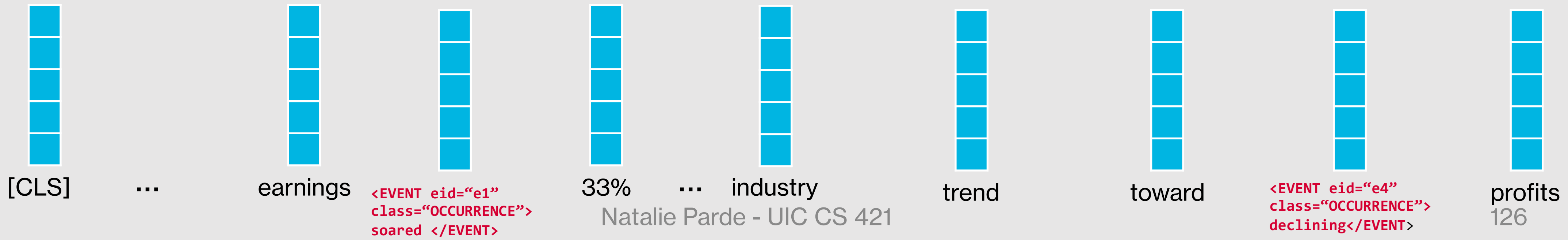
How do we perform temporal ordering?

Detect all events and temporal expressions from the text

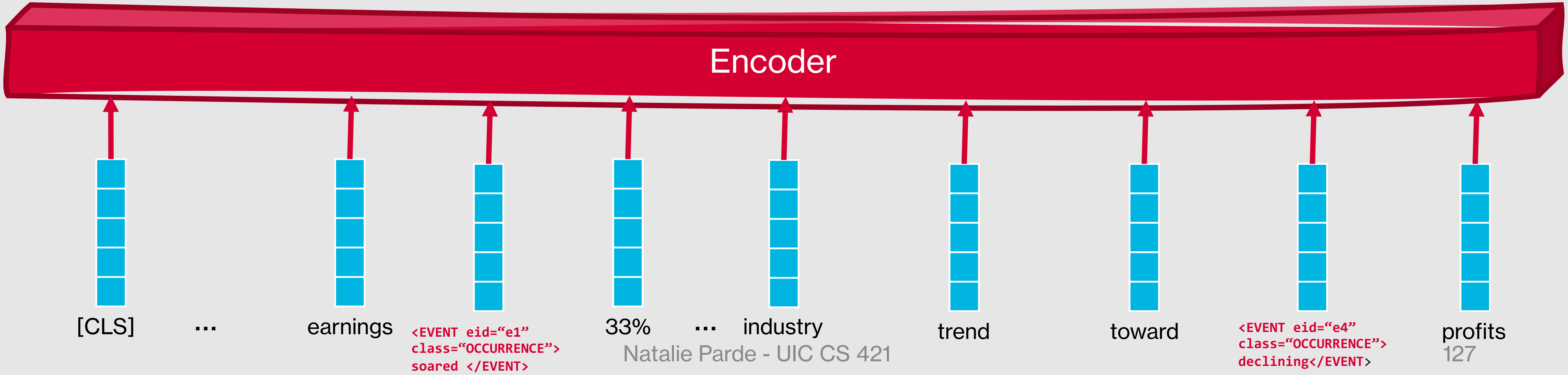
For all possible event-event, event-time, and time-time pairs, assert links

- Train temporal relation classifiers to predict TLinks

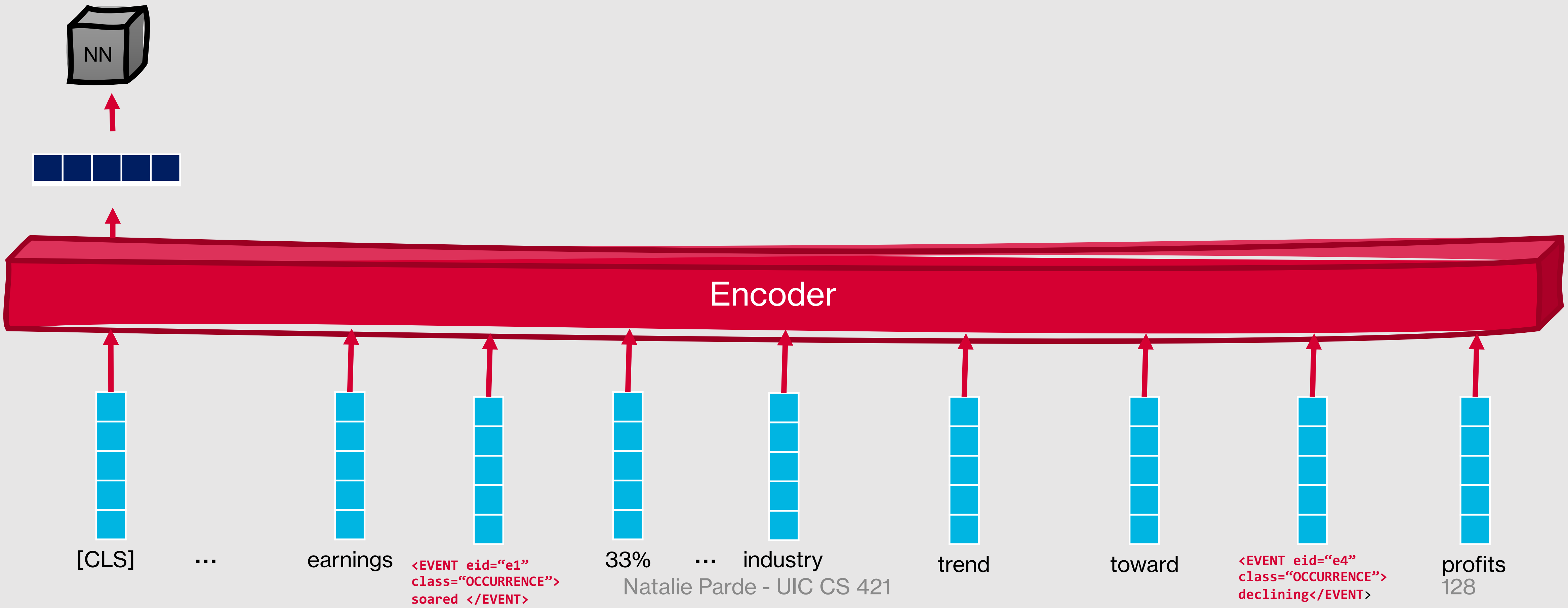
Example Neural Temporal Ordering



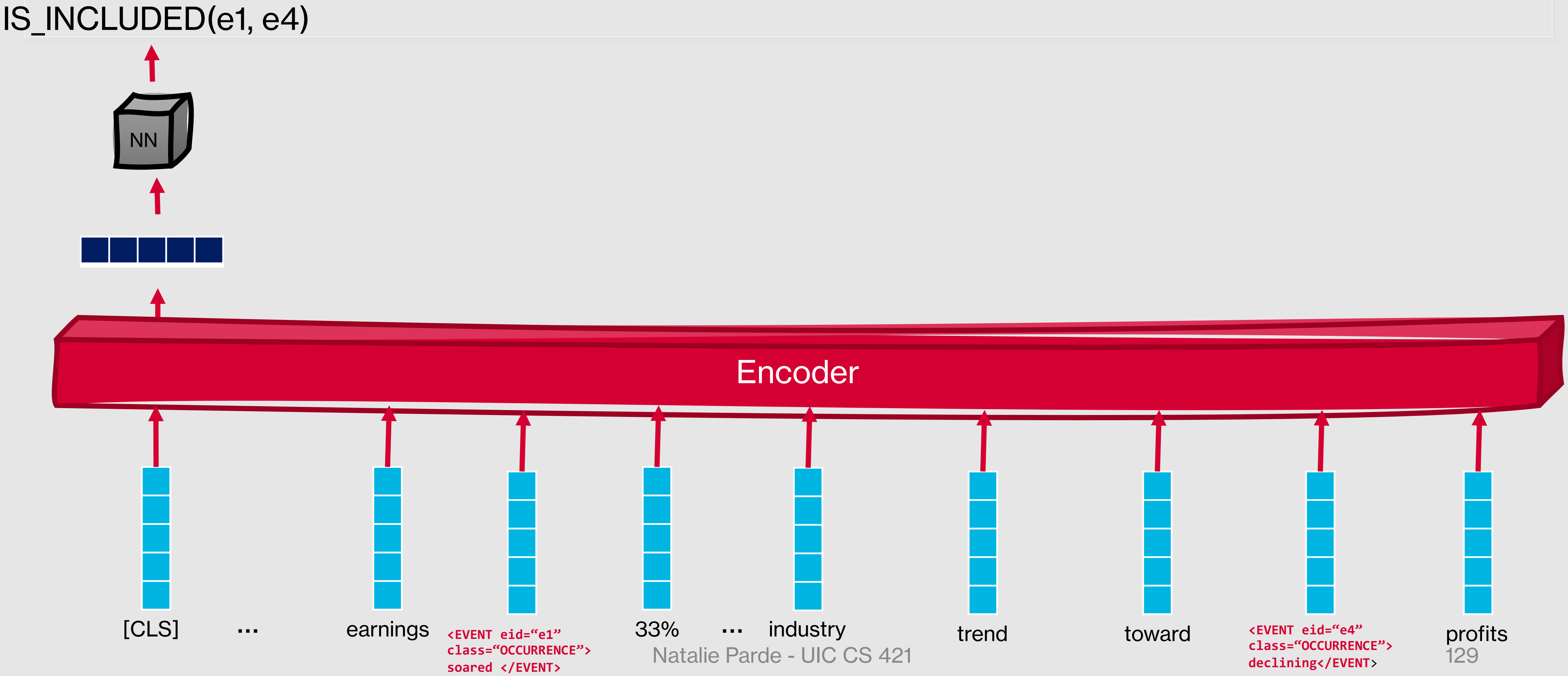
Example Neural Temporal Ordering



Example Neural Temporal Ordering



Example Neural Temporal Ordering



Summary: Event Extraction and Temporal Reasoning

- **Events** are actions or states that can be assigned to a particular point or interval in time
- **Template filling** approaches recognize specific **scripts** or events as **templates** and assign segments from the text to roles represented by a fixed set of slots participating in those templates
- **Temporal expressions** allow us to understand how events are situated within time
- We can represent **temporal logic** using the first-order logic framework and incorporating **interval algebra**
- **Aspect** defines categories of events or states based on their temporal structure
- Approaches for **temporal analysis** focus on extracting temporal expressions, normalizing those expressions, and using the extracted and normalized temporal information to place events in a structured order