CSC 5930/9010: Text Mining

Information Extraction Overview

Dr. Paula Matuszek
Paula.Matuszek@villanova.edu
Paula.Matuszek@gmail.com
(610) 647-9789
Information Extraction Overview

- Given a body of text: extract from it some well-defined set of information
- Typically draws heavily on NLP
- Three main components:
  - Domain knowledge base
  - Knowledge model
  - Extraction Engine
Information Extraction Domain Knowledge Base

- **Terms**: enumerated list of strings which are all members of some class.
  - “January”, “February”
  - “Smith”, “Wong”, “Martinez”, “Matuszek”
  - “lysine”, “alanine”, “cysteine”

- **Classes**: general categories of terms
  - Month names, Last Names, Amino acids
  - Capitalized nouns
  - Verb Phrases

- **Often organized into ontologies**
Domain Knowledge Base

- Rules: LHS, RHS, salience
- Left Hand Side (LHS): a pattern to be matched, written as relationships among terms and classes
- Right Hand Side (RHS): an action to be taken when the pattern is found
- Salience: priority of this rule (weight, strength, confidence)
Some Rule Examples:

- `<Monthname> <Year> => <Date>`
- `<Date> <Name> => print “Birthdate”, <Name>, <Date>`
- `<Name> <Address> => create address database record`
- `<daynumber> “/” <monthnumber> “/” <year> => create date database record (50)`
- `<monthnumber> “/” <daynumber> “/” <year> => create date database record (60)`
- `<capitalized noun> <single letter> “.” <capitalized noun> => <Name>`
- `<noun phrase> <to be verb> <noun phrase> => create “relationship” database record`
Generic KB

- Generic KB: KB likely to be useful in many domains
  - names
  - dates
  - places
  - organizations
- Almost all Information Extraction systems have one
- Limited by cost of development: it takes about 200 rules to define dates reasonably well, for instance.
Domain-specific Knowledge Base

- We mostly can’t afford to build a KB for the entire world.
- However, most applications are fairly domain-specific.
- Therefore we build domain-specific KBs which identify the kind of information we are interested in.
  - Protein-protein interactions
  - airline flights
  - terrorist activities
Domain-specific KBs

- Typically start with the generic KBs
- Add terminology
- Figure out what kinds of information you want to extract
- Add rules to identify it
- Test against documents which have been human-scored to determine precision and recall for individual items.
Knowledge Model

• We aren’t looking for documents, we are looking for information. What information?

• A knowledge model or schema identifies the information components we want and their relationship

• Typically looks very much like a DB schema or object definition
Knowledge Model Examples

- Personal records
  - Name
    - First name
    - Middle Initial
    - Last Name
  - Birthdate
    - Month
    - Day
    - Year
  - Address
    - House number
    - Street
    - City
    - State
    - Zip
Knowledge Model Examples

• Protein Inhibitors
  • Protein name (class?)
  • Compound name (class?)
  • Pointer to source
  • Cache of text
  • Offset into text
Knowledge Model Examples

- **Airline Flight Record**
  - **Airline**
  - **Flight**
    - **Number**
    - **Origin**
    - **Destination**
    - **Date**
    - **Status**
    - **departure time**
    - **arrival time**
Extraction Engine

- Tool which applies rules to text and extracts matches
  - Tokenizer
  - Part of Speech (POS) Tagger
  - Term and class tagger
  - Rule engine: match LHS, execute RHS
- Rule engine is iterative
- May include an interactive component which is essentially a query engine against already extracted information
Extraction Example: Birthdates

- Problem: create a database of birthdays from text with birth information

- Sample sentences:
  - George Washington was born in 1725.
  - Washington was born on Feb. 12, 1725.
  - Feb. 12 is Washington's birthday.
  - Washington's birth date is Feb. 12, 1725.
  - George Washington was born in America.
  - Washington's standard was born by his troops in 1778.
Birthdates: Knowledge Model

- **Simple birthdate model:**
  - Name
  - Birthdate

- **Complex birthdate model:**
  - Name
    - First Name
    - Middle Name
    - Last Name
  - Date
    - Day
    - Month
    - Year
Birthdates Knowledge Base

- Generic KB: Name, Date
- Domain specific KB: Rules
  1. `<Name> "was born" {"in"|"on"} <Date>`
     
     => Insert (Name, Date) into database
  2. `<Date> "is" <Name, possessive> "birthday"`
     
     => Insert (Name, Date) into database
  3. `<Name,possessive> "birth" "date" "is" <Date>`
     
     => Insert (Name, Date) into database
Birthdays: Extraction Process

- Washington was born in 1725

- Tokenize:
  - "Washington"
  - "was"
  - "born"
  - "in"
  - "1725"
  - "."
Extraction, POS Tagging

- "Washington", noun, proper noun, subject
- "was": auxiliary verb, past tense, third person singular (3PS)
- "born": verb, past tense, 3PS
- "was born": verb phrase, passive
- "in": preposition
- "1725": prepositional object
- "in 1725" prepositional phrase
Extraction, Class Tagging

- "Washington": Last Name
- "was": nothing additional
- "born": nothing additional
- "in": nothing additional
- "1725": Year
Extraction: Rules

- **Name Rules:**
  - "Washington": Name

- **Date Rules:**
  - "1725": Date

- **Birthday Rule # 1:**
  - Insert (Washington, 1725) into database
Other Approaches

- Classifiers trained using machine learning approaches.
- Hybrid systems using both approaches
Tools

• Tool we will cover:
  • GATE. Open source product (http://gate.ac.uk/) with extensive text processing capabilities.

• Some other well-known tools
  • I2E. Commercial product from Linguamatics (http://www.linguamatics.com/) with strong linguistics background and sophisticated interfaces.
  • NLTK has some limited entity and relation extraction tools
  • Calais. Commercial product from Reuters, based on ClearForest. Limited free functionality available (http://www.opencalais.com/)
  • ReVerb. (formerly TextRunner). U of Washington program for extracting binary relationships (http://reverb.cs.washington.edu/)
  • AeroText. Commercial product originally developed by Lockheed, now owned by Rocket Software. (http://www.rocketsoftware.com/products/rocket-aerotext)

• Find more at http://www.kdnuggets.com/software/text.html
More Tools

- In addition to tools developed as IE tools initially, there are IE capabilities in many other packages
  - Oracle: [http://docs.oracle.com/cd/B28359_01/datamine.111/b28129/text.htm](http://docs.oracle.com/cd/B28359_01/datamine.111/b28129/text.htm)
What is GATE?

• Stands for General Architecture for Text Engineering.
• Developed at the University of Sheffield
• Component-based architecture with data separated from applications, many discrete capabilities included as plugins.
Who Uses GATE?

- Scientists performing experiments that involve processing human language
- Developers developing applications with language processing components
- Teachers and students of courses about language and language computation
- Us :-)
How GATE can Help?

• Specify an architecture, or organizational structure, for language processing software

• Provide a framework that implements the architecture and can be used to embed language processing capabilities in applications

• Provide a development environment built on top of the framework made up of convenient tools for developing components (plugins)
Really?

• Yeah, really.

• It’s been under development for 18 years and is still under very active development

• Open-source, with dozens of developers, some of whom have been involved since the beginning

• Active community that provides good support
  • Mailing list: lists.sourceforge.net/lists/listinfo/gate-users
  • twitter: twitter.com/#!/GateAcUk
  • LinkedIn: http://www.linkedin.com/groups/GATE-2230077

• Many other text mining capabilities have been integrated with it.

• An almost overwhelming amount of documentation
GATE Architecture Overview

http://gate.ac.uk/overview.html
GATE Product Family

- GATE Developer: IDE for language processing, with information extraction and other plugins.
- GATE Embedded: object library which can be included in applications
- GATE Teamware: collaborative annotation environment
- GATE Mimir: a “multiparadigm index” which supports semantic indexing and search
- GATE Wiki: “controllable wiki” based on Grails and Subversion
- GATE Cloud: GATE embedded running on supercomputer hardware
GATE Core Concepts

- The basic business of GATE is annotating documents
- Core concepts are
  - the *documents* to be annotated
  - *corpora* comprising sets of documents, grouping documents for the purpose of running uniform processes across them
  - *annotations* that are created on documents
  - *annotation types* such as ‘Name’ or ‘Date
  - *annotation sets* comprising groups of annotations
  - *processing resources* that manipulate and create annotations on documents
  - *applications*, comprising sequences of processing resources, that can be applied to a document or corpus
GATE Components

- We will deal primarily with GATE Developer:
- It has four components:
  - Applications: groups of processes to be run on a document or corpus.
  - LanguageResources (LRs): entities such as lexicons, documents, corpora, annotation schemas, ontologies.
  - ProcessingResources (PRs): tools that operate on unstructured text, such as parsers and tokenizers. These are mostly plugins.
  - DataStores: saved processed documents and resources.
Summary

- Text mining below the document level
- NOT typically interactive, because it’s slow
- Typically builds up a DB of information which can then be queried
- Uses a combination of term- and rule-driven analysis and Natural Language Processing parsing.
- We will be working with GATE
  - A rich collection of resources
  - Which operate on unstructured text
  - To produce and display structure