Machine Learning in GATE

Angus Roberts, Horacio Saggion, Genevieve Gorrell
Recap

• Previous two days looked at knowledge engineered IE
• This session looks at machine learned IE
• Supervised learning
• Effort is shifted from language engineers to annotators
Outline

• Machine Learning and IE
• Support Vector Machines
• GATE's learning API and PR
• Learning entities – hands on
• Learning relations – demo
• (classifying sentences and documents)
Machine learning for information extraction
Machine Learning

• We have data items comprising labels and features
  • E.g. an instance of “cat” has features “whiskers=1”, “fur=1”. A “stone” has “whiskers=0” and “fur=0”

• Machine learning algorithm learns a relationship between the features and the labels
  • E.g. “if whiskers=1 then cat”

• This is used to label new data
  • We have a new instance with features “whiskers=1” and “fur=1”--is it a cat or not???
Types of ML

- Classification
  - Training instances pre-labelled with classes
  - ML algorithm learns to classify unseen data according to attributes
- Clustering
  - Unlabelled training data
  - Clusters are determined automatically from the data
- Derive representation using ML algorithm
- Automate decision-making in the future
**ML in Information Extraction**

- We have annotations (classes)
- We have features (words, context, word features etc.)
- Can we learn how features match classes using ML?
- Once obtained, the ML representation can do our annotation for us based on features in the text
  - Pre-annotation
  - Automated systems
- Possibly good alternative to knowledge engineering approaches
  - No need to write the rules
  - However, need to prepare training data
ML in Information Extraction

• Central to ML work is evaluation
  • Need to try different methods, different parameters, to obtain good result
• Precision: How many of the annotations we identified are correct?
• Recall: How many of the annotations we should have identified did we?
• F-Score:
  \[ F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \]
• Testing requires an unseen test set
  • Hold out a test set
    • Simple approach but data may be scarce
  • Cross-validation
    • split training data into e.g. 10 sections
    • Take turns to use each “fold” as a test set
    • Average score across the 10
ML Algorithms

- Vector space models
  - Data have attributes (word features, context etc.)
  - Each attribute is a dimension
  - Data positioned in space
  - Methods involve splitting the space
  - Having learned the split, apply to new data
  - Support vector machines, K-Nearest Neighbours etc.
- Finite state models, decision trees, Bayesian classification and more …
- We will focus on support vector machines today
Support vector machines
Support Vector Machines

• Attempt to find a hyperplane that separates data
• Goal: maximize margin separating two classes
• Wider margin = greater generalisation
Support Vector Machines

- Points near decision boundary: support vectors (removing them would change boundary)
- Points far from boundary not important for decision
- What if data doesn't split?
  - Soft boundary methods exist for imperfect solutions
  - However linear separator may be completely unsuitable
Support Vector Machines

• What if there is no separating hyperplane?
• See example:
• Or class may be a globule

They do not work!
Kernel Trick

- Map data into different dimensionality
- Now the points are separable!
- E.g. features alone may not make class linearly separable but combining features may
- Generate many new features and let algorithm decide which to use
Support Vector Machines

- SVMs combined with kernel trick provide a powerful technique
- Multiclass methods simple extension to two class technique (one vs. another, one vs. others)
- Widely used with great success across a range of linguistic tasks
GATE's learning API and PR
API and PRs

• User Guide 9.24
  ▪ Machine Learning PR

• Chapter 11
  ▪ Machine Learning API
    • Support for 3 types of learning
    • Produce features from annotations
    • Abstracts away from ML algorithms
  ▪ Batch Learning PR
    • A GATE language analyser
Instances, attributes, classes

California Governor Arnold Schwarzenegger proposes deep cuts.

Instances: Any annotation
Tokens are often convenient

Attributes: Any annotation feature relative to instances
Token.String
Token.category (POS)
Sentence.length

Class: The thing we want to learn
A feature on an annotation

Entity.type = Location
Entity.type = Person
Surround mode

- This learned class covers more than one instance....
- Begin / End boundary learning
- Dealt with by API - surround mode
- Transparent to the user
California Governor Arnold Schwarzenegger proposes deep cuts.

- Three classes, including null
- Many algorithms are binary classifiers
- One against all (One against others)
  - LOC vs PERS+NULL / PERS vs LOC+NULL / NULL vs LOC+PERS
- One against one (One against another one)
  - LOC vs PERS / LOC vs NULL / PERS vs NULL
- Dealt with by API - `multClassification2Binary`
- Transparent to the user
ML applications in GATE

- Batch Learning PR
  - Evaluation
  - Training
  - Application
- Runs after all other PRs – must be last PR
- Configured via xml file
- A single directory holds generated features, models, and config file
The configuration file

```xml
<?xml version="1.0"?>
<ML-CONFIG>
  <VERBOSITY level="1"/>
  <SURROUND value="true"/>
  <FILTERING ratio="0.0" dis="near"/>
</ML-CONFIG>
```

- **Verbosity**: 0, 1, 2
- **Surround mode**: set true for entities, false for relations
- **Filtering**: e.g. remove instances distant from the hyperplane
Thresholds

- Control selection of boundaries and classes in post processing
- The defaults we give will work
- Experiment
- See the documentation

<PARAMETER
    name="thresholdProbabilityEntity" value="0.3"/>
<PARAMETER
    name="thresholdProbabilityBoundary" value="0.5"/>
<PARAMETER
    name="thresholdProbabilityClassification" value="0.5"/>
Multiclass and evaluation

• Multi-class
  ▪ one-vs-others
  ▪ One-vs-another

• Evaluation
  ▪ Kfold – runs gives number of folds
  ▪ holdout – ratio gives training/test
The learning Engine

- SVM: Java implementation of LibSVM
  - Uneven margins set with -tau

- NaiveBayesWeka

- C4.5Weka

- Learning algorithm and implementation specific
  SVM: Java implementation of LibSVM
    - Uneven margins set with -tau
The dataset

<DATASET>

• Defines
  ▪ Instance annotation
  ▪ Class
  ▪ Annotation feature to instance attribute mapping

</DATASET>
Learning entities

*Hands on*
The Problem

- Information extraction consists on the identification of pre-specified facts in running texts.
- One important component of any information extraction system is a named entity identification component.
- Two main approaches exist for the identification of entities in text:
  - Hand-crafted rules: you’ve seen the ANNIE system.
  - Machine learning approaches: we will explore one possibility in this session using a classification system.
- Manually developed rules use different sources of information: identity of tokens, parts of speech, orthography of the tokens, dictionary information (e.g., Lookup process), etc.
- ML components also rely on those sources of information and features have to be carefully selected by the ML developer.
The Problem
Features for learning
Consider the string “Alcan, Inc.” in the text what we want the ML component to do is to annotate this whole string as a company name. Note that the ML component will treat this problem as classification: it will transform this into the problem of classifying individual tokens in text (e.g. “Alcan” is the beginning of a company name and “.” (after Inc) is the end of the company name.

There are several “features” one could use to recognize the string as the name of a company: the first token is a NNP (proper noun), the last token is a company designator, the first token after the string is the verb “to engage”, etc.

We are going to consider features which can be extracted from the linguistic and semantic analysis of the text: tokenisation, parts of speech tagging, morphological analysis, gazetteer lookup, and entity recognition.

Additionally one may use information computed by a parser, dependency relations, or syntactic information.

In some cases extra processes will be required in order to transform the result of the analysis into features the ML component can use.
Exercise I

- Implement a ML component based on SVM to identify the following concepts in company profiles:
  - company name
  - address
  - fax
  - phone
  - website
  - industry type
  - creation date
  - industry sector
  - main products
  - market locations
  - number of employees
  - stock exchange listings

- Materials (under directory hand-on-resources/ml/entity-learning)
  - training data: a set of 5 company profiles annotated with the target concepts (corpus/annotated) - each document contains an annotation Mention with a feature class representing the target concept
  - Test documents (without target concepts): a set of company profiles from the same source as the training data (corpus/testing)
  - SVM configuration file learn-company.xml (experiments/company-profile-learning)
Exercise I

1. Run an experiment with the training data to check the performance of the learning component
   • Create a corpus and populate it with the training data
   • Create a Learning PR using the provided configuration file
   • Create a corpus pipeline containing the Learning PR: set the Learning PR to “evaluation” mode
   • Run the pipeline over the corpus and examine the results

2. Run an experiment with the test data and check the results of the annotation process on unseen documents
   • Create a corpus and populate it with the training data
   • Create a Learning PR using the provided configuration file
   • Create a corpus pipeline containing the Learning PR: set the Learning PR to “training” mode
Exercise I

1. Run an experiment with the test data and check the results of the annotation process on unseen documents (cont)
   • Create a corpus with the test documents
   • Annotate the documents in the corpus with ANNIE + grammar to create Entity (grammars/create_entity.jape)
   • Train the learning system using the training documents (training mode)
   • Apply the learning system (application mode) to the test documents – use your own annotation set as output
   • Examine the result of the annotation process

2. Run an experiment with the training data to check the performance of the learning component by modifying some of the parameters (follow the steps in 1.) - create a working directory, copy the configuration file, modify it, and test the learning component with the modified configuration file (change for example the tau parameter from 1 to 0.5, etc.)
Exercise II

• Implement a ML component based on SVM to learn ANNIE, e.g. To learn to identify the following concepts or named entities: Location, Address, Date, Person, Organization

• Materials (under directory hand-on-resources/ml/entity-learning)
  • We will use the testing data provided in Exercise I

• Create a corpus with the test data and prepare it for learning and testing
  • Annotate the corpus with ANNIE + the Entity grammar

• Inspired by the previous exercise create a configuration file that will learn the concept Entity and its type (you can not use Entity as a feature for learning!)

• Run a ML experiment using your configuration file, use the “evaluation” mode over the corpus and analyse the results
Exercise II

• As a variation, separate a few documents for testing, train the learner without the separated documents, and run it in application mode over the test documents.

• You may want to use the annotationDiff tool to verify in each document how the learner performed.
Learning relations

Demonstration
The CLEF project
More sophisticated indexing and querying
Why was a drug given?
What were the results of an exam?
Supervised system architecture

GATE training pipeline
- Linguistic processing
- Model learning

Human annotated gold standard

GATE application pipeline
- Termino term recognition
- Other linguistic processing
- Model application

Application texts

Termino database

External knowledge

<xml>
<de-id’d text>
<entities>
<ontology links>
<relations>
</xml>
Previous work

- Clinical relations have usually been extracted as part of a larger clinical IE system.
- Extraction has usually involved syntactic parses, domain-specific grammars and knowledge bases, often hand crafted.
- In other areas of biomedicine, statistical machine learning has come to predominate.
- We apply statistical techniques to clinical relations.
## Entity types

<table>
<thead>
<tr>
<th>Entity type</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition</td>
<td>Symptom, diagnosis, complication, etc.</td>
</tr>
<tr>
<td>Drug or device</td>
<td>Drug or some other prescribed item</td>
</tr>
<tr>
<td>Intervention</td>
<td>Action performed by a clinician</td>
</tr>
<tr>
<td>Investigation</td>
<td>Tests, measurements and studies</td>
</tr>
<tr>
<td>Locus</td>
<td>Anatomical location, body substance</td>
</tr>
</tbody>
</table>
## Relation types

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Argument 1</th>
<th>Argument 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>has_target</td>
<td>Investigation</td>
<td>Locus</td>
</tr>
<tr>
<td></td>
<td>Intervention</td>
<td>Locus</td>
</tr>
<tr>
<td>has_finding</td>
<td>Investigation</td>
<td>Condition</td>
</tr>
<tr>
<td></td>
<td>Investigation</td>
<td>Result</td>
</tr>
<tr>
<td>has_indication</td>
<td>Drug or device</td>
<td>Condition</td>
</tr>
<tr>
<td></td>
<td>Intervention</td>
<td>Condition</td>
</tr>
<tr>
<td></td>
<td>Investigation</td>
<td>Condition</td>
</tr>
<tr>
<td>has_location</td>
<td>Condition</td>
<td>Locus</td>
</tr>
<tr>
<td>negation_modifies</td>
<td>Negation modifier</td>
<td>Condition</td>
</tr>
<tr>
<td>laterality_modifies</td>
<td>Laterality modifier</td>
<td>Intervention</td>
</tr>
<tr>
<td></td>
<td>Laterality modifier</td>
<td>Locus</td>
</tr>
<tr>
<td>sub-location_modifies</td>
<td>Sub-location modifier</td>
<td>Locus</td>
</tr>
</tbody>
</table>
System architecture

GATE pipeline

- Pre-process
- Pair entities
- Generate relation model
  - learning
  - and application

SVM models

- Training and test texts
- Relation annotations

GATE
Learning relations

• Learn relations between pairs of entities
• Create all possible pairings of entities across \( n \) sentences in the gold standard, constrained by legal entity types
  - \( n \): e.g. the same, or adjacent
• Generate features describing the characteristics of these pairs
• Build SVM models from these features
Configuring in GATE

<DATASET>
  <INSTANCE-TYPE>theInstanceAnnotation</INSTANCE-TYPE>
  <INSTANCE-ARG1>featureForIdOfArg1</INSTANCE-ARG1>
  <INSTANCE-ARG2>featureForIdOfArg2</INSTANCE-ARG2>

  <FEATURES-ARG1>...</FEATURES-ARG1>

  <FEATURES-ARG2>...</FEATURES-ARG2>

  <ATTRIBUTE_REL>...</ATTRIBUTE_REL>

  <ATTRIBUTE_REL>...</ATTRIBUTE_REL>

  ...

</DATASET>
Creating entity pairings

- Entity pairings provide instances
- They will therefore provide features
- A “pairing and features” PR or JAPE needs to be run before the Learning
- Entities and features are problem specific
- We do not have a generic “pairing and features” PR
- You currently need to write your own
### Feature examples

<table>
<thead>
<tr>
<th>Features set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tokens(6)</td>
<td>Surface string and POS for window of 6</td>
</tr>
<tr>
<td>type</td>
<td>Concatenated type of arguments</td>
</tr>
<tr>
<td>direction</td>
<td>Linear text order of arguments</td>
</tr>
<tr>
<td>distance</td>
<td>Sentence and paragraph boundaries</td>
</tr>
<tr>
<td>string</td>
<td>Surface string features of context</td>
</tr>
<tr>
<td>POS</td>
<td>POS features of context</td>
</tr>
<tr>
<td>intervening entities</td>
<td>Numbers and types of intervening entities</td>
</tr>
<tr>
<td>events</td>
<td>Intervening interventions &amp; investigations</td>
</tr>
</tbody>
</table>
# Performance by feature set

<table>
<thead>
<tr>
<th>Feature set</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>tokens(6) + type</td>
<td>33</td>
<td>22</td>
<td>26</td>
</tr>
<tr>
<td>+ direction</td>
<td>38</td>
<td>36</td>
<td>37</td>
</tr>
<tr>
<td>+ distance</td>
<td>50</td>
<td>70</td>
<td>58</td>
</tr>
<tr>
<td>+ string</td>
<td>63</td>
<td>74</td>
<td>68</td>
</tr>
<tr>
<td>+ POS</td>
<td>62</td>
<td>73</td>
<td>67</td>
</tr>
<tr>
<td>+ intervening entities</td>
<td>64</td>
<td>75</td>
<td>69</td>
</tr>
<tr>
<td>+ events</td>
<td>65</td>
<td>75</td>
<td>69</td>
</tr>
<tr>
<td>IAA</td>
<td></td>
<td></td>
<td>47</td>
</tr>
<tr>
<td>CIAA</td>
<td></td>
<td></td>
<td>75</td>
</tr>
</tbody>
</table>