

# Hybrid Information Extraction Systems for Open Data

## Class 2: Entity Linking

Pablo Ariel Duboue, PhD

Curso de Posgrado Cs. de la Computacion  
FaMAF-UNC

# What are Named Entities?

- ▶ At its core, proper names
- ▶ Nowadays generalized to nouns and multi-word expressions within a semantic class
  - ▶ 200 categories including “color” which contain common nouns
- ▶ Useful outside IE
  - ▶ Reduce the vocabulary space, instead of every name of every person have a token “NAME\_OF\_PERSON”

## Some Example NEs

[Fred Flintstone]<sup>person</sup> was named [CTO]<sup>position</sup> of [Time Bank Inc.]<sup>organization</sup> in [2031]<sup>date</sup>. The [next year] [he] got married and became [CEO]<sup>position</sup> of [Dinosaur Savings & Loan]<sup>organization</sup>.

from Grishman (2012)

# NE in the Context of IE

- ▶ NEs are the entries in the DB
- ▶ Events are the DB schema

# Named Entity Recognition Techniques

- ▶ Three techniques
  - ▶ Regular Expressions
  - ▶ Lists of names (gazetteers)
  - ▶ Machine learning
    - ▶ Word Sense Disambiguation
    - ▶ Semi-supervised

# Regular Expressions

- ▶ Regular Expressions are a succinct way to encode an automaton that accepts a regular language
- ▶ Constructs:
  - ▶ literal (e.g. /RÉSOLU/)
  - ▶ character class (e.g., /[0-9]/)
  - ▶ quantifiers (e.g., /,? /)
  - ▶ groups (e.g., /([0-9][0-9][0-9])/)

# RE: Literal

- ▶ Literals are characters or sequences of characters that need to be matched verbatim
  - ▶ In perl code from the baseline: `/ (concernant) /`
- ▶ Characters that have a meaning in the RE language (e.g., `' ? '`) need to be escaped (e.g., `' \ ? '`)
  - ▶ In Java you will need to double escape (e.g., `" \\ ? "`)
- ▶ Special quotation to escape unknown sequences: `\E ... \Q`
  - ▶ In perl code from the baseline: `/ \Q $amount \E \spour /`

## RE: Character Classes

- ▶ Succinct way to describe a set of characters
  - ▶ Either by listing all members (e.g. `/[xyz]/`)
  - ▶ Or by using a range (e.g., `/[0-9]/`)
- ▶ There are also patterns for most common sets
  - ▶ `/\d/` for digits
  - ▶ `/\s/` for white space
  - ▶ special class `/./` that matches any character



## RE: Quantifiers

- ▶ Extend the smaller regular recursively
  - ? one or nothing
  - \* nothing, one or more
  - + one or more
  - {n,m} at least n, at most m
- ▶ From perl baseline `/.*RÉSOLU À L'UNANIMITÉ:/`

# RE: Groups

- ▶ Concatenate other regular expressions
  - ▶ `/concernant?\spour/` means the `t` is optional!
  - ▶ `/((concernant)?\spour/` means `concernant` is optional
- ▶ The regular expressions inside a group could be of any complexity, including groups
  - ▶ `/((concernant\s)?pour/`
- ▶ By default groups are capturing which means the match is returned by the system
  - ▶ Non-capturing groups are indicated with `?:`
    - ▶ `/(?:concernant\s)?pour/`

## RE: Alternatives

- ▶ Indicates two or more regular expressions could be matched
  - ▶ `/((du\scontrat\sde)|(requis\spour)|(concernant)/`
  - ▶ Character classes are a succinct way to representing many alternatives
- ▶ Watch out the need for grouping
  - ▶ `/du\scontrat\sde|requis\spour/` means `(du\scontrat\sde)[er](equis\spour)` which is not what you want

# Amount RE

- ▶  
`/(\d?\d?(?:\s?|\.?)\d{3}(?:\s?|,?)(?:\d{3})?(?:\,\d{2})?\s?\$)/`
- ▶ `(\d?\d? // two digits (optional)`
  - ▶ `(?:\s?|\.?) // a space or a period (optional)`
  - ▶ `\d{3} // three digits (required)`
  - ▶ `(?:\s?|,?) // a space or a period (optional)`
  - ▶ `(?:\d{3})? // three digits (optional)`
  - ▶ `(?:\,\d{2})? // an optional comma followed by two digits`
- ▶ `\s?\$) // an optional space with a required dollar sign`

# What is Machine Learning?

- ▶ A new way of programming
- ▶ Magic!
- ▶ Leaving part of the behavior of your program to be specified by calculating unknown numbers from "data"
  - ▶ Two phases of execution: "training" and "application"

# The ultimate TDD

- ▶ If you're using a library, you almost do no coding, just test!
- ▶ But every time you test, your data becomes more and more obsolete
  - ▶ No peeking!
- ▶ Have met people who didn't have any tests and considered
  - ▶ Bugs in the code same are the same as model issues
  - ▶ My experience has been quite the opposite, the code you write on top of machine learning algorithms has to be double and triple checked

# Taxonomy of Machine Learning Approaches

- ▶ **Supervised learning**

*Monkey see, monkey do*

- ▶ Classification

- ▶ **Unsupervised learning**

*Do I look fat?*

- ▶ Clustering

- ▶ **Others**

- ▶ Reinforcement learning: learning from past successes and mistakes (good for game AIs and politicians)
  - ▶ Active learning: asking what you don't know (needs less data)
  - ▶ Semi-supervised: annotated + raw data

# Concepts

- ▶ Trying to learn a function  $f(x_1, \dots, x_n) \rightarrow y$ 
  - ▶  $x_i$  are the **input** features.
  - ▶  $y$  is the **target** class.
- ▶ The key here is *extrapolation*, that is, we want our learned function to **generalize** to unseen inputs.
  - ▶ Linear interpolation is on itself a type of supervised learning.



# Data

- ▶ Collecting the data
  - ▶ Data collection hooks
  - ▶ Annotating data
    - ▶ Annotation guidelines
    - ▶ Cross and self agreement
- ▶ Representing the data (as **features**, more on this later)
- ▶ Understanding how well the system operates over the data
  - ▶ Testing on **unseen** data
- ▶ A DB is a rather poor ML algorithm
  - ▶ Make sure your system is not just memorizing the data
  - ▶ “Freedom” of the model

# Evaluating

- ▶ Held out data
  - ▶ Make sure the held out is representative of the problem and the overall population of instances you want to apply the classifier
- ▶ Repeated experiments
  - ▶ Every time you run something on eval data, it changes you!
- ▶ Cross-validation
  - ▶ Training and testing on the same data but not quite
  - ▶  $\text{data} = \{A, B, C\}$ 
    - ▶ train in A, B, test in C
    - ▶ train in A, C, test in B
    - ▶ train in B, C, test in A

# Metrics

- ▶ Measuring how many times a classifier outputs the right answer (“accuracy”) is not enough
  - ▶ Many interesting problems are very biased towards a background class
  - ▶ If 95% of the time something doesn’t happen, saying it’ll never happen (not a very useful classifier!) will make you only 5% wrong
- ▶ Metrics:

$$\text{precision} = \frac{|\text{correctly tagged}|}{|\text{tagged}|} = \frac{tp}{tp + fp}$$

$$\text{recall} = \frac{|\text{correctly tagged}|}{|\text{should be tagged}|} = \frac{tp}{tp + fn}$$

$$F = 2 \cdot \frac{P \cdot R}{P + R}$$

# Naive Bayes

- ▶ Count and multiply
- ▶ How spam filters work
- ▶ Very easy to implement
- ▶ Works relatively well but it can seldom solve the problem completely
  - ▶ If you add the target class as a feature, it will still has a high error rate
  - ▶ It never “trusts” anything too much

# Why Naive?

- ▶ Bayes Rule

$$p(C | F_1, \dots, F_n) = \frac{p(C)p(F_1, \dots, F_n | C)}{p(F_1, \dots, F_n)}$$

$$posterior = \frac{prior \times likelihood}{evidence}$$

- ▶ Naive Part

- ▶ Independence assumption of the  $F_x$ , that is

$$p(F_i | C, F_j) = p(F_i | C)$$

$$p(C | F_1, \dots, F_n) \propto p(C)p(F_1 | C) \dots p(F_n | C)$$

# Maximum Entropy

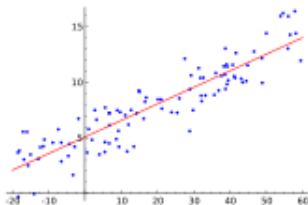
- ▶ Tons and tons of (binary) features
- ▶ Very popular at beginning of 2000's
  - ▶ CRF has taken some of its glamour
    - ▶ Mature code
- ▶ OpenNLP MaxEnt uses strings to represent its input data

*previous=succeeds current=Terrence next=D.  
currentWordsCapitalized*
- ▶ Training with `trainModel(dataIndexer, iterations)` and using it with `double[] eval(String[] context)`

# Maximum Entropy Details

- ▶ A maximum entropy classifier is a multi-class generalization of a logistic regression, so we will discuss them instead
  - ▶ Multi-class can be obtained by a number of techniques, e.g., using a pivot class
- ▶ Related to Naive Bayes if we think of it as not estimating the probability of the features given the class, but the probability of the class given the features directly
  - ▶ So we don't need to make an independence assumption between the features anymore

# Linear Regression



from Wikipedia

$$y_i = \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^T \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, \dots, n$$



# Least-squares Estimation

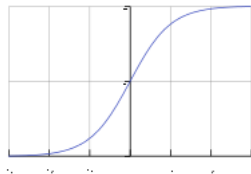
Representing the problem as  $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$  where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

a closed solution for  $\boldsymbol{\beta}$  is given by:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = (\sum_i \mathbf{x}_i \mathbf{x}_i^T)^{-1} (\sum_i \mathbf{x}_i y_i)$$

# Logistic Function



from Wikipedia

$$F(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

# Logistic Regression: Intuition

- ▶ We move from the target function space to the probability space for the target function being on a certain class
- ▶ Minimize the error for a linear combination of features approximating the logit of the said probability
- ▶ No close solution, use numerical methods to find an approximate solution

# Logistic Regression, estimating $p_i$

$$Y_i \mid x_{1,i}, \dots, x_{m,i} \sim \text{Bernoulli}(p_i) \quad (1)$$

$$\mathbb{E}[Y_i \mid x_{1,i}, \dots, x_{m,i}] = p_i \quad (2)$$

$$\Pr(Y_i = y_i \mid x_{1,i}, \dots, x_{m,i}) = \begin{cases} p_i & \text{if } y_i = 1 \\ 1 - p_i & \text{if } y_i = 0 \end{cases} \quad (3)$$

$$\Pr(Y_i = y_i \mid x_{1,i}, \dots, x_{m,i}) = p_i^{y_i} (1 - p_i)^{(1-y_i)} \quad (4)$$

# Estimating $p_i$ and regression coefficients

- ▶ The  $p_i$  and the linear combination coefficients are all unknown so we resort to a search process that minimizes the error on training

$$\text{logit}(p_i) = \ln \left( \frac{p_i}{1 - p_i} \right) = \beta_0 + \beta_1 x_{1,i} + \dots + \beta_m x_{m,i}$$

- ▶ Need some regularization process to avoid trivial or overly complex solutions

# How to Come Up with Features

1. Throw everything (and the kitchen sink) at it
2. Stop and think
  - 2.1 What information would **you** us to solve that problem?
  - 2.2 Look for published work
    - ▶ Papers: <http://aclweb.org/anthology-new/>
    - ▶ Blog postings
    - ▶ Open source projects
3. Add computable features
  - ▶ Learning to sum takes an incredible amount of training!

# Improving the Classifier

- ▶ More data
- ▶ Better features
- ▶ Solve a different problem
- ▶ Shop around for a different classifier / parametrization
  - ▶ Procedural overfitting
- ▶ Add unlabelled data
- ▶ Drop ML and program it by hand

# NE Detection

- ▶ Dictionary-based
- ▶ WSD
- ▶ Semi-supervised



# Dictionary-based (Gazetteers)

- ▶ Gazetteers and their problems
  - ▶ Spurious matches
  - ▶ In Octroy, “La Firme” was in some moment the name of a company
    - ▶ Same with “CE”
- ▶ Need annotated corpus for evaluation, otherwise more rules hurt performance

# Simple Rules

- ▶ All capitalized sequences that end in “Inc.” are companies
  - ▶ Work well as a starting point
- ▶ Need annotated corpus for evaluation, otherwise more rules hurt performance

# WSD-based

- ▶ The context around an occurrence determines its function
- ▶ If we can segment the text around likely NEs, we can then use the context to determine its type

# Biology: Problem

- ▶ “Disambiguating proteins, genes, and RNA in text: a machine learning approach,” Hatzivassiloglou, Duboue, Rzhetsky (2001)
- ▶ The same term refers to genes, proteins and mRNA:
  - ▶ “By UV cross-linking and immunoprecipitation, we show that **SBP2** specifically *binds* selenoprotein *mRNAs* both in vitro and in vivo.”
  - ▶ “The **SBP2** *clone* used in this study generates a 3173 nt transcript (2541 nt of coding sequence plus a 632 nt 3' UTR truncated at the polyadenylation site).”
- ▶ This ambiguity is so pervasive that in many cases the author of the text inserts the word “gene”, “protein” or “mRNA” to disambiguate it itself
  - ▶ That happens in only 2.65% of the cases though

## Biology: Features

- ▶ Take a context around the term, use the occurrence of words before or after the term as features.
- ▶ Keep a tally of the number of times each word has appear with which target class:

term	gene	protein	mRNA
PRIORS	0.44	0.42	0.14
D-PHE-PRO-VAL-ORN-LEU		1.0	
NOVAGEN	0.46	0.46	0.08
GLCNAC-MAN	1.0		
REV-RESPONSIVE	0.5	0.5	
EPICENTRE		1.0	
GENEROUSLY	0.33	0.67	

# Biology: Methods

- Instead of multiplying, operate on logs

```
float[] predict = (float[]) priors.clone();  
// ... for each word in context ...  
if (wordfreqs.containsKey(word)) {  
    float[] logfreqs = wordfreqs.get(word);  
    for (int i = 0; i < predict.length; i++)  
        predict[i] += logfreqs[i];  
}
```

# Biology: Results

- ▶ Used a number of variations on the features
  - ▶ Removed capitalization, stemming, filtered part-of-speech, added positional information
  - ▶ Changed the problem from three-way to two-way classification
- ▶ Results of Tree-learning and Naive Bayes were comparable (76% two-way and 67% three-way).
- ▶ Distilled some interesting rules from the decision trees:
  - ▶ after ENCODES is present  
before ENCODES is NOT present  
⇒ class gene [96.5%]

# Sequence Tagging (IOB)

- ▶ Given  $n$ -tags, create  $2n + 1$  classes:
  - ▶ B-tag: this word starts a tag
  - ▶ I-tag: this word is inside a tag
  - ▶ O: this word is outside all tags (background model)
- ▶ Learn a classifier that goes from features around a word to these classes



# IOB Example

## ► From WNUT NER competition

- tonite O
- " O
- running B-tvshow
- wit I-tvshow
- mjd I-tvshow
- " O
- live O
- from O
- 7- O
- 9pm O
- eastern O
- sirius B-company
- 211 O
- . O

# WNUT NER 2016 baseline python system

- ▶ Workshop on Noisy User-generated Text (W-NUT) NER:

- ▶ [https://github.com/aritter/twitter\\_nlp/tree/master/data/annotated/wnut16](https://github.com/aritter/twitter_nlp/tree/master/data/annotated/wnut16)

- ▶ Training data (annotated tweets) → train.feats:

**O** w[-1]=*planning* w[0]=*the* w[-1]|w[0]=*planning|the* w[0]|w[1]=*the|next*  
DICT=tv.tv\_program DICT=people.person word=*the* word\_lower=*the*  
**O** w[-1]=*the* w[0]=*next* w[-1]|w[0]=*the|next* w[0]|w[1]=*next|Disney*  
DICT=tv.tv\_program DICT=people.person word=*next*  
word\_lower=*next* prefix=*n* prefix=*ne* prefix=*nex* suffix=*t* suffix=*xt*  
**B-facility** w[-1]=*next* w[0]=*Disney* w[-1]|w[0]=*next|Disney*  
w[0]|w[1]=*Disney|World* DICT=tv.tv\_program DICT=people.person  
word=*Disney* word\_lower=*disney* prefix=*d* prefix=*di* prefix=*dis* suffix=*y*  
suffix=*ey* suffix=*ney* INITCAP INITCAP\_AND\_GOODCAP  
**I-facility** w[-1]=*Disney* w[0]=*World* w[-1]|w[0]=*Disney|World*  
w[0]|w[1]=*World|trip* DICT=tv.tv\_program DICT=people.person  
word=*World* word\_lower=*world* prefix=*w* prefix=*wo* prefix=*wor* suffix=*d*

# Bootstrapping (Semisupervised)

- ▶ Use a seed set of entities to collect contexts around them on a corpus
- ▶ Apply the trained system to the corpus, collect more entities
- ▶ Repeat until no more entities are found
- ▶ As described, never seen working in practice
  - ▶ Need to clean the list of extracted entities by hand after each iteration or it ends up annotating everything
- ▶ Can be improved using co-training: using an alternative view of the data as a separate classifier (different features or different corpus or different approach)

# Annotating Entities

- ▶ Interesting entities are rare, if you annotate a random sample you will need to annotate a very large number of documents to achieve coverage
- ▶ Alternative: Active Learning, let the trained system pick new things to annotate
  - ▶ Problem is that your system will be biased towards rare cases, need to weight the training samples

# Evaluation

- ▶ Guidelines are usually straightforward to follow
- ▶ Problems:
  - ▶ Systematic polysemy (GPE vs location)
  - ▶ Corpus characterization (training on certain data, applying on different one)
  - ▶ Tags missing / spurious / partial match

# Entity Linking

- ▶ Determine which occurrences refer to which entity
- ▶ Need a corpus of disambiguated references or at least text related to each entity
- ▶ `https://github.com/ejmeij/  
entity-linking-and-retrieval-tutorial`

# In-Document Linking

- ▶ In-document coreference

- ▶ Pronouns detection is good (80-90%)

[Fred Flintstone] was named CTO of Time Bank Inc. in 2031.  
The next year [he] ...

- ▶ Nouns coreference is not that good

# Open Data Taxonomies

- ▶ DBpedia
- ▶ Extracted from Wikipedia Info Boxes
- ▶ RDF triples



# DBpedia Spotlight



Confidence:  0.5 Language: French ▾

☐ n-best candidates SELECT TYPES... ANNOTATE

---

CE-2012/389

CRÉDITS ADDITIONNELS – GREFFE

RÉSOLU À L'UNANIMITÉ:  
d'approuver des crédits additionnels au montant de 10 000 \$, plus  
les taxes applicables, pour défrayer les honoraires juridiques dans  
la cause Elie Chakieh et Hellen Christodoulou contre Ville de  
[Laval](#) (re: requête introductive d'instance, Cour supérieure,  
district de [Laval](#), 540-17-004312-103)  
(C/T: 1230955)  
(D/Greffe: 15-2010-MB-0594)  
(Réf: 2-5)

BACK TO TEXT

Only showing the types: DBpedia:Place

from <http://dbpedia-spotlight.github.io/demo/>

# DBpedia Spotlight Demo Server

- ▶ Accessing the demo server:

```
curl http://model.dbpedia-spotlight.org/en/annotate  
--data-urlencode "text=President Obama called  
Wednesday on Congress ..."  
--data "confidence=0.35"
```

- ▶ Result:

```
{ "Resources": [ { "@URI": "http://dbpedia.org/resource/Barack_Obama",  
"@support": "14695", "@types": "DBpedia:Agent, Schema:Person,  
Http://xmlns.com/foaf/0.1/Person, DBpedia:Person, DBpedia:OfficeHolder",  
"@surfaceForm": "President Obama", "@offset": "0",  
"@similarityScore": "0.993", "@percentageOfSecondRank": "0.006" },  
{ "@URI": "http://dbpedia.org/resource/United_States_Congress",  
"@support": "19906", "@types": "DBpedia:Agent,  
Schema:Organization, DBpedia:Organisation, DBpedia:Legislature",
```

# DBpedia Spotlight Running your Own

- ▶ `wget http://model.dbpedia-spotlight.org/dbpedia-spotlight-latest.jar`
- ▶ `wget http://downloads.dbpedia-spotlight.org/en/2016-04/model/en.tar.gz`
- ▶ `tar xzf en.tar.gz`
- ▶ `java -jar dbpedia-spotlight-latest.jar en http://localhost:2222/rest`
- ▶ Available trained models for:
  - ▶ da, de, en, **es**, **fr**, hu, it, nl, pt, ru, sv, tr
- ▶ <https://github.com/dbpedia-spotlight/dbpedia-spotlight/tree/master/uima>

# DBpedia Spotlight (2013)

- ▶ Phrase Spotting:
  - ▶ First step: identify candidates for possible annotations
    - ▶ Language independent: FSA with all possible Wikipedia titles and rewordings
    - ▶ Language dependent: Capitalized Tokens + OpenNLP models
  - ▶ Second step: remove overlaps using preferences among generation methods
- ▶ Disambiguation:
  - ▶ Generative probabilistic model based on contexts
  - ▶ Counts for the model are kept on indices using Kyro

## NER in the End-to-End Case Study

- ▶ Continuing with the Octroy Pipeline, we will analyze code in branch class2 of
  - ▶ <https://github.com/IE4OpenData/Octroy>
- ▶ Dependency management and build cycle is usually managed with the Apache Maven tool:
  - ▶ mvn clean
    - ▶ mvn compile
    - ▶ mvn test
    - ▶ mvn appassembler:assemble
    - ▶ pom.xml (Project Object Model)

# Running the Pipeline

- ▶ `./target/appassembler/bin/run-pipeline-tsv` or `run-pipeline-xmi`
- ▶ Focus on Company annotations
- ▶ OpenNLP pipeline:
  - ▶ `./target/appassembler/bin/run-pipeline-xmi`  
`org/ie4opendata/octroy/OctroyEngineOpenNLP.xml`  
`./docs/dev32 /tmp/dev32/`
- ▶ ConceptMapper pipeline:
  - ▶ `./target/appassembler/bin/run-pipeline-xmi`  
`org/ie4opendata/octroy/OctroyEngineCM.xml` `./docs/dev32`  
`/tmp/dev32`

# Evaluating the Results

- ▶ [https://github.com/IE4OpenData/ruta\\_testing\\_standalone](https://github.com/IE4OpenData/ruta_testing_standalone)
  - ▶ `mvn package appassembler:assemble`
  - ▶ `./target/appassembler/bin/ruta-evaluate`
- ▶ Evaluating XML annotated with different pipelines (output on `/tmp/dev32`)

```
$ /path/to/ruta_testing_standalone/target/appassembler/bin/ruta-evaluate \  
--gold data/gold32 --eval /tmp/dev32 \  
--include org.ie4opendata.octroy.Company \  
--typesystem ./src/main/resources/org/ie4opendata/octroy/octroy_eval_ts.xml
```

# The Entities

- ▶ Amount
- ▶ Company
- ▶ Two approaches:
  - ▶ Dictionary using UIMA Concept Mapper (with linking)
  - ▶ Machine Learning MEMM using OpenNLP NameFinder



# Linking Approach

- ▶ We use ConceptMapper with the NEQ code as a field in the dictionary
- ▶ [http://duboue.net/download/neq\\_dict.xml.gz](http://duboue.net/download/neq_dict.xml.gz)
  - ▶ 289Mb decompressed
  - ▶ ~2 million canonical entries over 3.6 million variants

# The Types

- ▶ DocumentAnnotation
- ▶ Token
- ▶ Sentence
- ▶ Amount
- ▶ Company

# The AEs

- ▶ ContractClassifier
- ▶ ContractFlowController
- ▶ AmountAnnotator
- ▶ NeqConceptMapper

## Some Results

- ▶ ContractClassifier works well
- ▶ ConceptMapper produces too many spurious matches
  - ▶ Add ML
  - ▶ Filter dictionary through general purpose corpus

# Some Results (cont.)

CE-2009/219

DÉBUT DES TRAVAUX - SOUMISSION «9380»  
RÈGLEMENT L-11330-U - ASPHALTE DESJARDINS INC.  
RÉSOLU À L'UNANIMITÉ:  
que la Direction générale soit et, par la présente, est  
autorisée à  
faire débiter par la compagnie Asphalte Desjardins inc.  
les  
travaux prévus au règlement numéro L-11330-U.  
soumission  
«9380», et ce, sur réception du cautionnement;  
il est également résolu d'autoriser la firme CIMA+ à  
effectuer la  
surveillance desdits travaux, les honoraires étant calculés  
conformément aux dispositions de l'offre de service «OS-  
9244».  
(C/T: 1096419)  
(Réf: 26-96)

Click In Text to See Annotation Detail

Annotations

- Company
  - Company ("ASPHALTE DESJARDINS INC.")
    - begin = 128
    - end = 152
    - registrationNumber = 1143605
    - officialName = CARRIÈRES LAUR

Legend

☒ Com... ☐ Doc... ☐ Doc... ☐ Sent... ☐ Token

Select All Deselect All Hide Unselected

## Some Results (cont.)

CE-2009/219

DÉBUT DES TRAVAUX - SOUMISSION «9380»  
RÈGLEMENT L-11330-U - ASPHALTE DESJARDINS INC.  
RÉSOLU À L'UNANIMITÉ:  
que la Direction générale soit et, par la présente, est  
autorisée à  
faire débiter par la compagnie Asphalte Desjardins inc.  
les  
travaux prévus au règlement numéro L-11330-U,  
soumission  
«9380», et ce, sur réception du cautionnement;  
il est également résolu d'autoriser la firme CIMA+ à  
effectuer la  
surveillance desdits travaux, les honoraires étant calculés  
conformément aux dispositions de l'offre de service «OS-  
9244»,  
(C/T: 1096419)  
(Réf: 26-96)

Click In Text to See Annotation Detail

Annotations

- Company
  - Company ("CE")
    - begin = 55
    - end = 57
    - registrationNumber = 3363487
    - officialName = SPAZZME DES
  - Company ("CE")
    - begin = 55
    - end = 57
    - registrationNumber = 1167964
    - officialName = CE

Legend

☒ Com... ☐ Doc... ☐ Doc... ☐ Sent... ☐ Token

Select All Deselect All Hide Unselected

# Some Results (cont.)

CE-2009/219

DÉBUT DES TRAVAUX - SOUMISSION «9380»  
RÈGLEMENT L-11330-U - ASPHALTE DESJARDINS INC.  
RÉSOLU À L'UNANIMITÉ:  
que la Direction générale soit et, par la présente, est  
autorisée à  
faire débiter par la compagnie Asphalte Desjardins inc.  
les  
travaux prévus au règlement numéro L-11330-U.  
soumission  
«9380», et ce, sur réception du cautionnement;  
il est également résolu d'autoriser la firme CIMA+ à  
effectuer la  
surveillance desdits travaux, les honoraires étant calculés  
conformément aux dispositions de l'offre de service «OS-  
9244».  
(C/T: 1096419)  
(Réf: 26-96)

**Click In Text to See Annotation Detail**

Annotations

- Company
  - Company ("la firme")
    - begin = 443
    - end = 451
    - registrationNumber = 1163122
    - officialName = MORIS, ALLIANCE
  - Company ("la firme")
    - begin = 443
    - end = 451
    - registrationNumber = 1170863
    - officialName = La Firme - Comp
  - Company ("la firme")
  - Company ("la firme")
  - Company ("la firme")
  - Company ("la firme")
  - Company ("la firme")

**Legend**

☒ Com... ☐ Doc... ☐ Doc... ☐ Sent... ☐ Token

Select All Deselect All Hide Unselected

# Evaluation

- ▶ `JAVA_OPTS=-Xmx6G`  
`./target/appassembler/bin/run-pipeline-xmi`  
`org/ie4opendata/octroy/OctroyEngineCM.xml ./docs/dev32`  
`./output/dev32-cm`
- ▶ `../ruta_testing_standalone/target/appassembler/bin/ruta-`  
`evaluate --gold data/gold32 --eval ./output/dev32-cm`  
`--typesystem ./src/main/re-`  
`sources/org/ie4opendata/octroy/octroy_eval_ts.xml --include`  
`org.ie4opendata.octroy.Company`
- ▶ FP 522 / FN 8 / TP 11
- ▶ Prec 0.021 (very, very low!)
- ▶ Rec 0.579 (good)
- ▶ F1 0.04 (very, very low)
- ▶ Overgenerates



# MEMM Approach

- ▶ Annotate the named entities
- ▶ Train a Maximum Entity model
- ▶ `opennlp.tools.namefind.NameFinderME`

```
private static AdaptiveFeatureGenerator createFeatureGenerator() {  
    return new CachedFeatureGenerator(  
        new AdaptiveFeatureGenerator[] {  
            new WindowFeatureGenerator(new TokenFeatureGenerator(), 2, 2),  
            new WindowFeatureGenerator(new TokenClassFeatureGenerator(true), 2,  
2),  
            new OutcomePriorFeatureGenerator(),  
            new PreviousMapFeatureGenerator(),  
            new BigramNameFeatureGenerator(),  
            new SentenceFeatureGenerator(true, false)  
        }  
    );  
};
```

# Generate Training File

- ▶ OpenNLP has a standalone model generating capabilities, based on “semi-structured text files”
  - ▶ This approach encodes tokens as white-space separated terms
  - ▶ And sentences as full lines
  - ▶ NER tokens are marked with pseudo SGML tags as  
`<START:person> <END>`
- ▶ This format clearly does not use stand-off annotation but we will look into it to build up towards using ClearTk
  - ▶ Annotating training data with this format will render the annotations useless if the tokenizer / sentence boundary detector is changed!
- ▶ You can generate the training file (empty file, the annotations have to be added by hand using a text editor) using
  - ▶ `./target/appassembler/bin/opennlp-trainer-extractor <folder with txt files> <output file>`
- ▶ An annotated file is in `data/company.training36`
  - ▶ Note the sentence boundaries are very poor

# Available Models

- ▶ Trained MEMM models for multiple tasks (NER but also tokenization and sentence boundary detection) are made available by the OpenNLP project only at
  - ▶ <http://opennlp.sourceforge.net/models-1.5/>