# NERC

Named Entity Recognition and Classification

# Outline

- State of the art
- Tools
- References
- QA

# Team

- Bizdadea Dan
- Bratiloveanu Florentina
- Ciobanu Catalin
- Sorostinean Mihaela

# State of the art

- Maximum Entropy Models (ME) (Catalin)
- Support Vector Machines (SVM) (Dan)
- Neural Networks (NN) (Flori)
- Decision Trees (Mihaela)

# Maximum entropy models

- What is a Maximum Entropy model ?
- Named Entity Recognition models:
    - O. Bender, F. J. Ochm and H. Ney [1]
    - J. R. Curran and S. Clark [2]
    - Y.-F. Lin, T.-H. Tsai, W.-C. Chou [3]

# What is a maximum entropy model ?

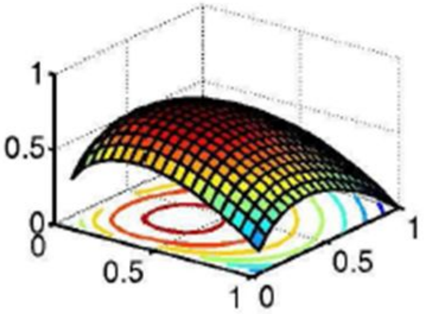
- Goal : estimate p
- Choose p with maximum entropy (“uncertainty”) subject to the constraints (“evidence”)
- Collect (a, b) pairs where:
    - a is the thing to be predicted (e. g. tag sequence)
    - b is the context (e. g. word sequence)

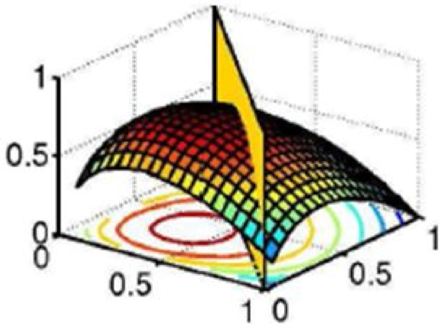
$$H(p) = -\sum_{x \in A \times B} p(x) \log p(x)$$

$$x = (a, b), \quad where \quad a \in A \wedge b \in B$$

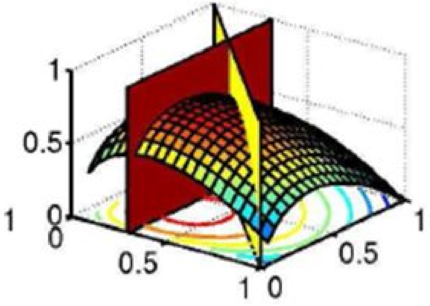
# Example ME Model

- Toss a coin: p(H) = p1, p(T) = p2
- Constraint: p1 + p2 = 1
- p = (p1, p2) ?
- Maximize H(p) by choosing p




p1 + p2 = 1




p1+p2 =1 p1= 0.3

# Features

- A feature function is the set of possible classes:

$$f_j : \varepsilon \rightarrow \{0,1\}, \quad \varepsilon = A \times B$$

- A is the set of possible classes (e. g. tags)
- B space of context (e. g. neighbouring words)

$$f_j(a,b) = \begin{cases} 1 & \mathit{if} \;\; a = DET \;\&\; curWord(b) = \text{"}that\text{"} \\ 0 & o.w. \end{cases}$$

# Restating the problem

- The task: find p* such that: $p^* = \arg\max_{p \in P} H(p)$ where:

$$P = \{p \mid E_p f_j = E_{\tilde{p}} f_j, j = \{1, \ldots, k\}\}$$

- Constraint:

$$\{E_p f_j = E_{\tilde{p}} f_j = d_j, j = \{1, \ldots, k\}\}$$

- Observed probability of x $\tilde{p}(x)$
- Model probability of x $p(x)$
- Model expectation $E_p f_j = \sum_{x \in \varepsilon} p(x) f_j(x)$

# O. Bender, F. J. Ochm and H. Ney ME model

- Lexical features
- Word features
  - Capitalization
  - Digits and numbers
  - Prefixes and suffixes
- Dictionary features
- Feature selection: Given a threshold k, include those features that have been observed on the training data at least k times.
- Search: Viterbi search to compute the highest probability sequence

$$
\begin{aligned}
Pr(c_1^N | w_1^N) &= \prod_{n=1}^{N} Pr(c_n | c_1^{n-1}, w_1^N) \\
&\underset{model}{=} \prod_{n=1}^{N} p(c_n | c_{n-2}^{n-1}, w_{n-2}^{n+2}) \, .
\end{aligned}
$$

# J.R. Curran and S. Clark ME model

- Language Independent NER

| Condition | Contextual predicate |
|---|---|
| $freq(w_i) < 5$ | $w_i$ contains period<br>$w_i$ contains punctuation<br>$w_i$ is only digits<br>$w_i$ is a number<br>$w_i$ is {upper,lower,title,mixed} case<br>$w_i$ is alphanumeric<br>length of $w_i$<br>$w_i$ has only Roman numerals<br>$w_i$ is an initial (X.)<br>$w_i$ is an acronym (ABC, A.B.C.) |
| $\forall w_i$ | memory NE tag for $w_i$<br>unigram tag of $w_{i+1}$<br>unigram tag of $w_{i+2}$ |
| $\forall w_i$ | $w_i$ in a gazetteer<br>$w_{i-1}$ in a gazetteer<br>$w_{i+1}$ in a gazetteer |
| $\forall w_i$ | $w_i$ not lowercase and $f_{lc} > f_{uc}$ |
| $\forall w_i$ | unigrams of word type<br>bigrams of word types<br>trigrams of word types |

# Y.-F Lin, T.-H. Tsai, W.-C. Chou ME Model

- Lists derived from training data
  - Frequent Word List (FWL)
  - Useful Unigrams (UNI)
  - Useful Bigrams (UBI) (e.g. “CITY OF”, “ARRIVES IN”)
  - Useful Word Suffixes (SUF)
  - Useful Name Class Suffixes (NCS)
  - Function Words (FUN)
- Global features
  - Acronyms
  - Unigrams
  - Bigrams

# Support Vector Machines - Introduction

- SVMs introduced in COLT-92 by Boser, Guyon, Vapnik
- Kernel Machines: large class of learning algorithms, SVMs a particular instance
- SVM is a linear learning system that builds two-class classifiers :
- {(x1,y1),(x2,y2),...,(xn,yn)}
- Xi = (xi1,xi2,...,xir) is a r-dimensional input vector
- Yi =  is its class label and yi E {1,-1}
- SVM finds a linear function of the form
- f(x) = <w.x> + b

# SVM - Introduction

In essence, SVM finds a hyperplane

<w.x> + b = 0

That separates positive and negative training examples.

This hyperplane is called a decision boundary or decision surface

Geometrically , the hyperplane <w.x> + b = 0 divides the input space into two half spaces : one half for positive examples and other half for negative examples

# Linear SVM - Separable case

In <w.x> + b = 0, w defines a direction perpendicular to the hyperplane. W is called the normal vector of the hyperplane.

Without changing the normal vector, varying b moves the plane parallel to itself

We define two parallel planes H+ and H0 that pass through x+ and x-.

H+ : <w . x+> + b = 1 and H- : <w . x-> + b = -1

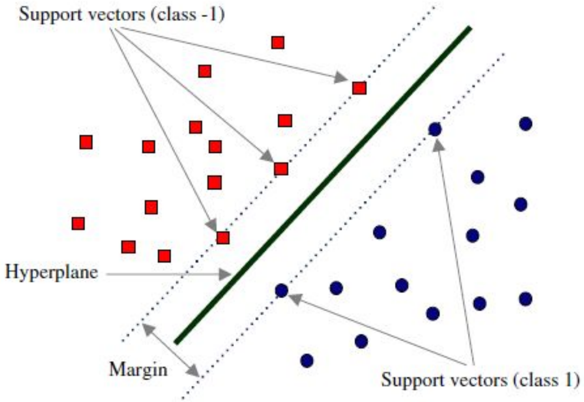
Such that: <w . xi> + b >= 1 if yi = 1 AND <w . xi> + b <= 1 if yi = -1

Distance from xi to hyperplane =  |<w.xi> + b| / ||w||

D+  = distance from a point xs on hyperplane to the support vector (class 1):

D+ = |<w.xs> + b -1| / ||w|| = 1 / || w|| , because <w.xs> + b = 0

Same goes for D- = 1 / ||w|| => margin = (d+) + (d-) = 2 / ||w||



# Linear SVM - Separable case

Maximizing the margin = minimizing ||w||^2 / 2 = <w . w> / 2

We can define linear SVM as follows :

Given D = {(x1,y1), (x2,y2),...,(xn,yn)}, we need to solve the following constrained minimization problem :

Minimize : <w . w> / 2 , subject to : yi (<w . xi> + b) >= 1, i=1,2,...n

The full description of the solution requires significant amount of optimisation theory and time so I will not include it here.

Since the objective function is quadratic and convex and the constraints are linear in the parameters w and b, we can use the standard Lagrange multiplier to solve it.

# Linear SVM - Separable case - Testing

Our final decision boundary is :

$\langle w . x \rangle + b = \mathrm{Sum}(i\ E\ sv)\ \ y_i . a_i . \langle x_i . x \rangle + b = 0$, where sv is the set of indices of the support vectors in the training data

Testing :

Given a test instance z, we classify it using the following :

$$\mathrm{sign}(\langle w.z \rangle + b) = \mathrm{sign}(\mathrm{Sum}(i\ E\ sv)\ y_i . a_i . \langle x_i . z \rangle + b)$$

If it returns 1 then the test instance z is classified as positive, otherwise it is classified as negative.

# Linear SVM - Non separable case

In practice data is always noisy so the linear separable SVM will not find a solution because the constraints cannot be satisfied.

To allow errors in data we can relax the margin constraints by introducing slack variables psi(>=0) as follows :

<w . xi> + b >= 1 - psi for yi = 1

<w . xi> + b <= -1 + psi for yi = -1

# SVM limitations

- It works only in real valued space. For a categorical attribute we need to convert its categorical values to numeric values

- It allows only two classes. For multiple class classification problems several strategies can be applied : one-against-rest, one-against-one.

# Feature selection

- Internal features : are the ones provided from within the sequence of words that constitute the entity
- External features : are the ones obtained by the context in which entities appear.

Examples :

Context: previous m and next n words.

Suf : Suffix string of length N will be 1 if it contains punctuation or any other special symbol

Pre : Preffix string of length N will be 1 if it contains punctuation or any other special symbol or number

First word : 1 if it is the first word of a sentence

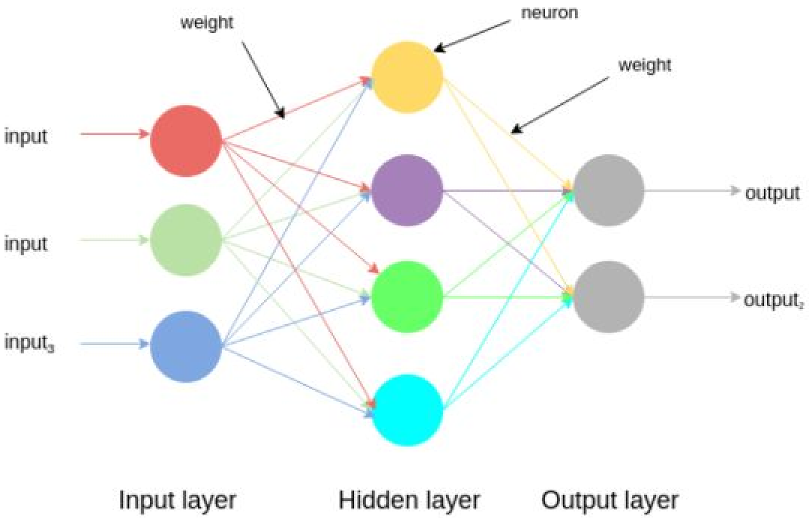
Last word : 1 if it is the last word of a sentence

CntDigit : 1 if the word contains digit.

# Neural Networks

- find models capable of generalization
- extract from low-level(word vectors) to high-level(combination of rudimentary features) features
- reduce programming burden
- applicability: cancer classification, text to speech and vice versa, object

# Neural Networks - visual representation



# Neural Networks - preprocessing (I)

- Word2vec
  - given text corpus for the distribution
  - output: set of vectors; basically, it turns text into a numerical form such that neural networks can understand
  - it can make highly accurate guesses about the words' meaning, and clusters words by meaning
  - it was recently shown that the word vectors capture many linguistic regularities, for example vector operations vector('Paris') – vector('France') + vector('Italy') results in a vector that is very close to vector ('Rome')
  - and vector('king') – vector('man') + vector('woman') is close to vector('queen')

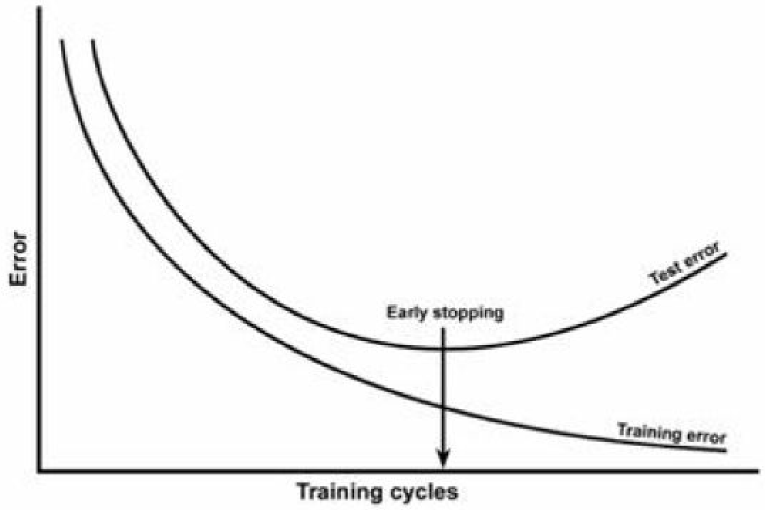
# Neural Networks - preprocessing (II)

- how many layers/features, what type of layers
- gradient descent versus stochastic gradient descent
- data normalization
- loss function: classification(binary/multiclass) or regression? Log-likelihood?

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[ -y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

# Neural Networks - train and test

- split dataset for training and testing
- when to stop training?



# NERC Systems based on Decision Trees

- in various works evaluated the behavior of C4.5 alg. on the task of learning decision trees to recognise and classify named entities in text
- the decision trees are used as NERC grammars
- among the advantages reported we can mention:
  - the recognisers built by C4,5 are quite simple and can be translated into a small number of comprehensible rules
  - the results obtained show that tree induced can outperform a manually constructed grammar

# NERC Systems based on Decision Trees

- C4.5 - supervised learning algorithm => induction of decision trees

  - search based on recursive partitioning of training data

  - at each stage a feature that discriminates best between examples is selected => increasingly purer subsets (many examples of one class)

  - overtraining of the decision tree -> prevented by introducing a pruning method

  - captures the most important classification patterns

  - can handle both symbolic and numerical data

# References

[1] Bender, Oliver, Franz Josef Och, and Hermann Ney. "Maximum entropy models for named entity recognition." *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, 2003.
[2] Curran, James R., and Stephen Clark. "Language independent NER using a maximum entropy tagger." *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, 2003.
[3] Chieu, Hai Leong, and Hwee Tou Ng. "Named entity recognition: a maximum entropy approach using global information." *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 2002.
[4] Liu, 2011 "*Web Data Mining*" p.109 - p.120 Ch 3.8 : Support Vector Machines
[5] Bishop,2006 "*Pattern recognition and machine learning*" p.325 - p.357, Ch. 7 : Sparse Kernel Machines
[6] Asif Ekbal and Sivaji Bandyopadhyay - "*Named Entity Recognition using Support Vector Machine: A Language Independent Approach*"
[7] Fredrick Edward Kitoogo and Venansius Baryamureeba - "*A Methodology for Feature Selection in Named Entity Recognition*"
[8] Joel Mickelin - "*Named Entity Recognition with Support Vector Machines*", Master of Science Thesis Stockholm, Sweden 2013
[9] Tomas Mikolov et al. - "Distributed Representations of Words and Phrases and their Compositionality"
[10] Xiang Zhang, Yann LeCun - "Text Understanding from Scratch"

# Questions ?

Thank you

for your attention!