Coreference Resolution

Machine Learning Approaches

What is Coreference Resolution?

Example

$[\text{Barack Obama}]_1^1$ nominated $[\text{Hillary Clinton}]_2^2$ as $[[\text{his}]_3^1$ secretary of state$]_4^3$ on $[\text{Monday}]_5^4$. $[\text{He}]_6^1$

- Superscript: ID of an entity
- Subscript: ID of a mention

Coreference as Clustering

- Bell number:

Number of Mentions	Bell number
10	115975
20	51724158236496
30	846749014529889671069667
50	1.8572414972124E+47
100	$2.3 \cdot 10^{117}$

Coreference as Clustering

- Conditional Random Fields for Coreference:

Features for the Clustering y

$$p(y|x) = \frac{\exp(\sum_{features} \lambda_f f(x,y))}{\sum_{\hat{y} \in clusterings} \exp(\sum_{features} \lambda_f f(x,\hat{y}))}$$

Arbitrary Clustering

Input text

➔ Infeasible to calculate, so we need to go back to (crude) approximations!

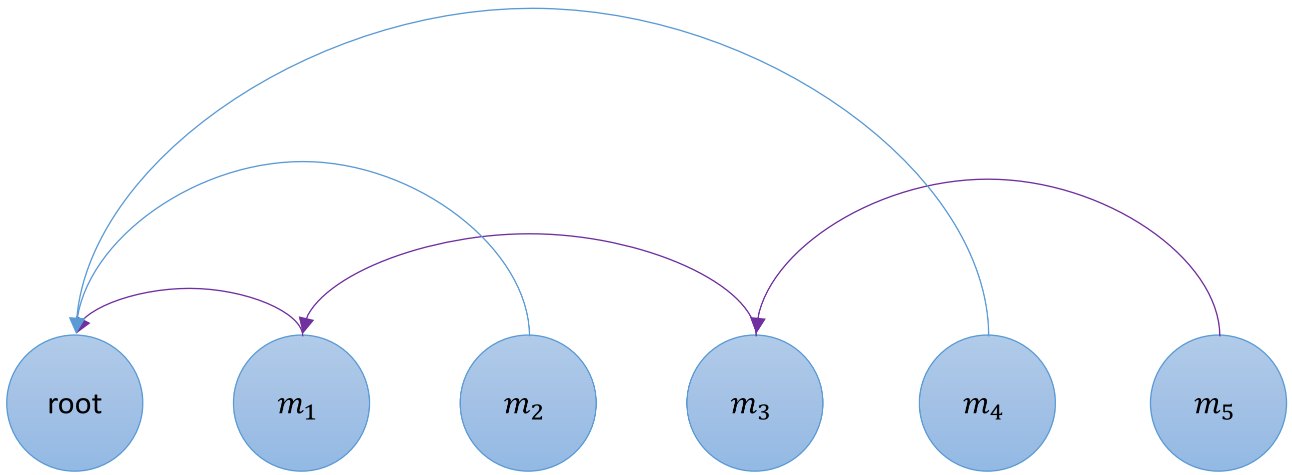
Coreference as Clustering

- Global methods are infeasible, since there is no dynamic programming for the scoring of the clusters

➔ We are looking for approaches, that search for local solutions, which we can combine into a clustering

Coreference as Dependency Parsing

- Let us compare Coreference Resolution to Dependency Parsing



- Instead of tokens, which are looking for a head, we are searching a head for every mention
- Newly introduced mentions will be resolved to the artificial *root node*

Coreference as Dependency Parsing

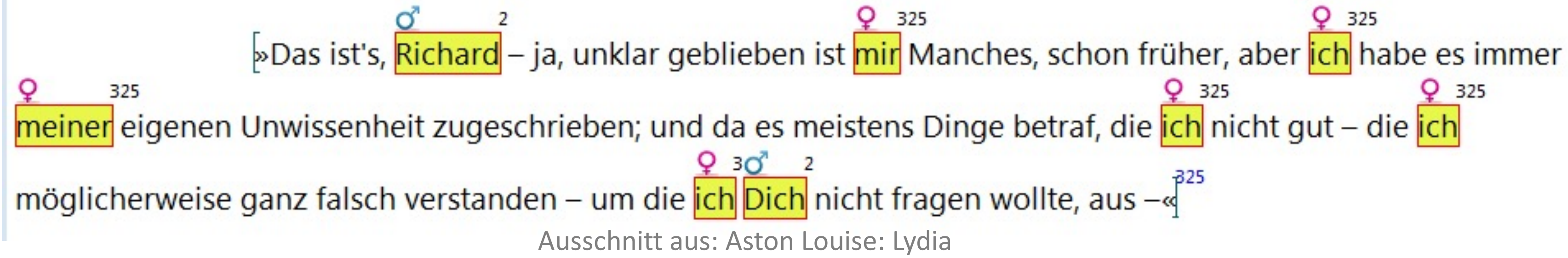
- The problem has now become much simpler:
  - For every mention, we have to find **any** antecedent
  - First appearances of an entity will be mapped to root

➔Instead of predicting an entire cluster at once, we will now only score **links between mentions**

- This results in at least the following approaches:
  1. Mention-Pair Modell
  2. Mention-Ranking Modell
  3. Entity-Mention Modell
  4. Cluster-Ranking Modell

Machine Learning: Mention-Pair

- Given a labelled corpus:



»Das ist's, Richard – ja, unklar geblieben ist mir Manches, schon früher, aber ich habe es immer meiner eigenen Unwissenheit zugeschrieben; und da es meistens Dinge betraf, die ich nicht gut – die ich möglicherweise ganz falsch verstanden – um die ich Dich nicht fragen wollte, aus –«

Ausschnitt aus: Aston Louise: Lydia

- Let us now model this into a learnable problem

Machine Learning: Mention-Pair

- We first extract all instances (x, y) as follows:
  - Every x is now a pair of mentions (m_i, m_j)
  - Every y is a label of $\{\text{coref}, \text{non}-\text{coref}\}$

➔ Simple binary, unstructured classification

- And on top, we get a quadratic amount of instances
  - E.g. for the labelled „Aston Louise- Lydia“ corpus: $\frac{643 \cdot 642}{2} = 206403$!!

Machine Learning: Mention-Pair

- Is getting so many training examples a good thing?
  - Laws et al: of 1,7 million instances, 98.5% were non-coref!
    (**Data Skewness**)

➔Either very robust classifier are required or a strategy to „sample“ the instances which carry most information:

1. Sampling
2. Active-Learning

Data Skewness

- Why is **Data Skewness** even a problem?

- Example with Maximum-Entropy:
  - Usually a good feature will receive a high weight λ during training
  - Being a good feature means, that is appears more often with class A, than with class B
  - But lets assume the calculation of our features is noisy (e.g. through different preprocessing or a noisy data set)

  ➔ If the feature is misleading just a single time out of 98 instances, then its entire expressiveness is already gone, because of the imbalance in our data set!

➔ Getting a more or less equal amount of instances for each class is a desired property!

Data Skewness

- Why is **Data Skewness** even a problem?

- Example with Perceptron:
  - You remember the bias $\boldsymbol{b}$
  - If there is such a large skewness in your data, you can think of your classification process as follows: Each instance has to persuade the classifier not to be of the most frequent class! (otherwise all the score will be negated by adding the bias!)

➔ Getting a more or less equal amount of instances for each class is a desired property!

Machine Learning: Mention-Pair

- Method of Soon et al. (2001):
  - Instead of building all pairs (m_i, m_j) of the training data, just take:

For each mention m_i:

Iterate backwards in the text, $(m_j\ , j < i)$

Insert the pair (m_i, m_j) into the train data

If (m_i, m_j) co-refer,

Break;

  - In words:
    For every mention m_i only add pairs with other mentions until you find the first mention that co-refers.

Machine Learning: Mention-Pair

- Method of Soon et al. (2001):

Example

$[\text{Barack Obama}]_1^1$ nominated $[\text{Hillary Clinton}]_2^2$ as $[[\text{his}]_3^1$ secretary of state$]_4^3$ on $[\text{Monday}]_5^4$. $[\text{He}]_6^1$

- This method produces the following pairs:
  - $(\text{Hillary Clinton, Barack Obama}): \text{Non}-\text{Coref}$
  - $(\text{his, Hillary Clinton}): \text{Non}-\text{Coref}$
  - $(\text{his, Barack Obama}): \text{Coref}$
  - $(\text{secretary of state}, X): \text{Non}-\text{Coref}$ (3 Instances)
  - $(\text{monday}, X): \text{Non}-\text{Coref}$ (4 Instances)
  - $(\text{He, Monday}): \text{Non}-\text{Coref}$
  - $(\text{He, secretary of state}): \text{Non}-\text{Coref}$
  - $(\text{He, his}): \text{Coref}$

13 instances
(11 vs. 2)

Machine Learning: Mention-Pair

- Features for our classifier:
  - E.g. for the pair (Barack Obama, his)
  - [Barack Obama]$_1^1$ nominated [Hillary Clinton]$_2^2$ as [[his]$_3^1$ secretary of state]$_4^3$ on [Monday]$_5^4$. [He]$_6^1$

- Brainstorming: …
  (you could think about this yourself, this is the key issue as of why this task is so hard!)

Machine Learning: Mention-Pair

- Features of Soon et al.

Type	Description	Value
Distance	Distance between m_i and m_j in terms of the number of sentences	Integer
String-matching	Is m_j an alias of m_i?	Boolean
	Do m_i and m_j match after stripping of articles and demonstrative pronouns?	Boolean
Grammatical	Does m_j start with an definite article?	Boolean
	Does m_j start with a demonstrative pronoun?	Boolean
	Is m_i pronominal?	Boolean
	Is m_j pronominal?	Boolean
	Do m_i and m_j agree in number?	Boolean
	Do m_i and m_j agree in gender?	Boolean
	Do m_i and m_j both contain a proper name?	Boolean
Syntactic	Is m_j an apposition?	Boolean
Semantic	Do m_i and m_j agree in semantic class	Boolean

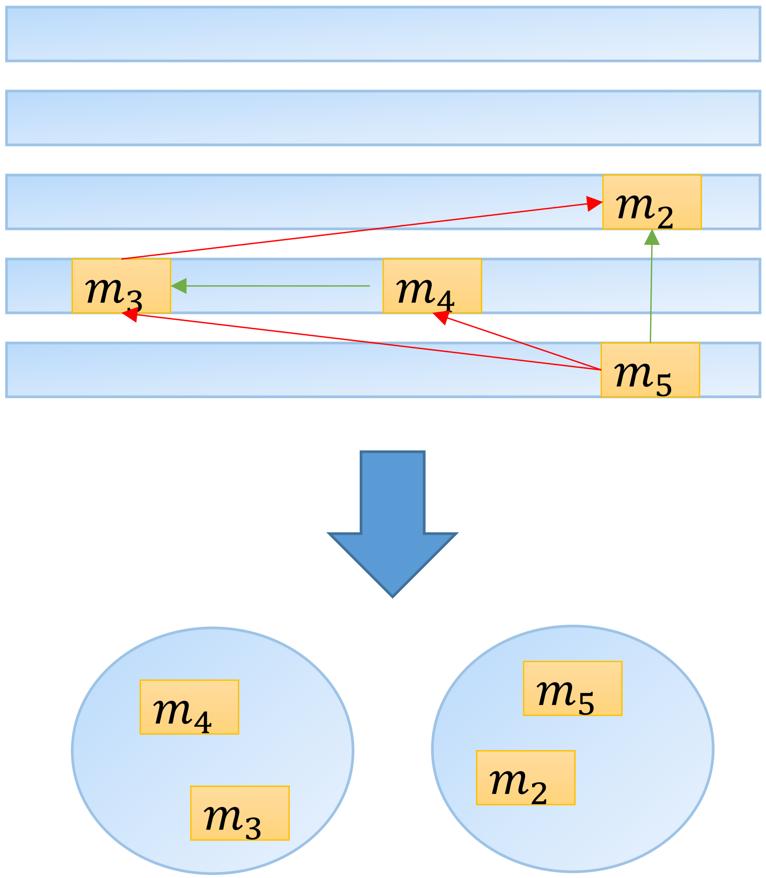
Machine Learning: Mention-Pair

- Application of the approach (according to Soon):

- Create instances „on-the-fly“ and predict them
  - Only backwards in the text

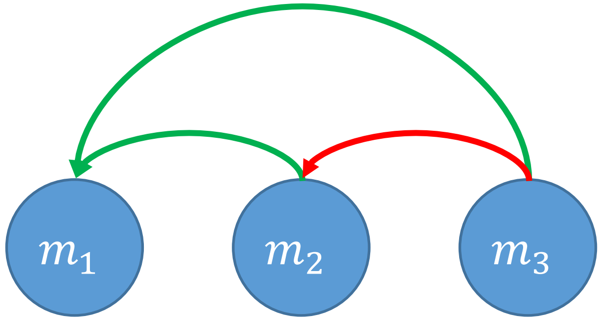
- Stop as soon as you get the first **Coref** prediction. Add the edge to the solution

➔ Build the clustering from the positive „edges“

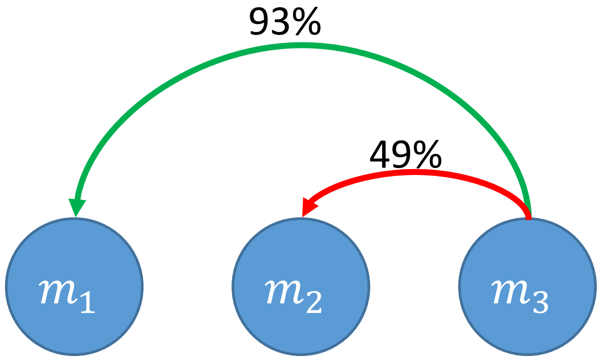


Machine Learning: Mention-Pair

- Problems:
  1. Mention-Pair produces inconsistencies
     - $(m_3, m_2) \Rightarrow \text{not coref}$
     - $(m_2, m_1) \Rightarrow \text{coref}$
     - $(m_3, m_1) \Rightarrow \text{coref}$



  2. Mentions are not competing (instances are independent)



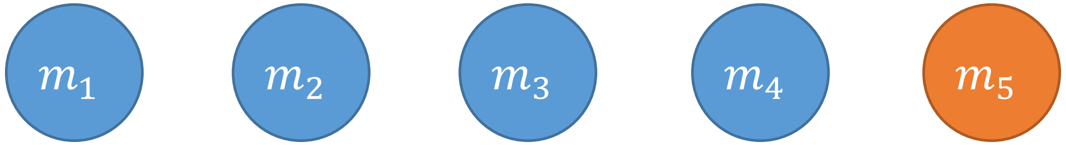
  3. Local features: They can only take two mentions into account!

Machine Learning: Mention-Ranking

- For every mention, „rank“ all candidates against each other and finally take the best one

Machine Learning: Mention-Ranking

- Ranking instead of classification: But how?
- Idea: No complete ranking is required, all we need is a correct first position!
- If for example we are looking for the best candidate for m_5
  - We have 4 mentions competing: m_4 up to m_1



Mention-Ranking: Tournament Model

- The tournament model is a determines a ranking with a simple binary classifier
  - And all we need to adjust is the way we model our instances
  - So far our instances have been of the form: $\left(x = \left(m_i, m_j\right), y \in \{\text{Coref}, \text{Non}-\text{Coref}\}\right)$
  - The tournament model changes this into: $\left(x = \left(m_i, m_j\right), \left(m_i, m_k\right), y \in \{\text{First–pair}, \text{Second–pair}\}\right)$
  - So the classifier gets two pairs and decides which pair is the better one
  - Classifier is still binary in nature
    - First-pair ➔ We prefer the first pairing $\left(\left(m_5, m_4\right); \left(m_5, m_3\right)\right)$
    - Second-pair ➔ We prefer the second pairing $\left(\left(m_5, m_4\right); \left(m_5, m_3\right)\right)$

➔ It is just a smart way to apply a binary classifier!

Mention-Ranking: Tournament Model

- How to apply it:
  - We always have two pairings with one „defeating“ the other
  - Once we have reached the first mention, there is a clear winner
  - E.g. (we want to find the the antecedent for m_5):

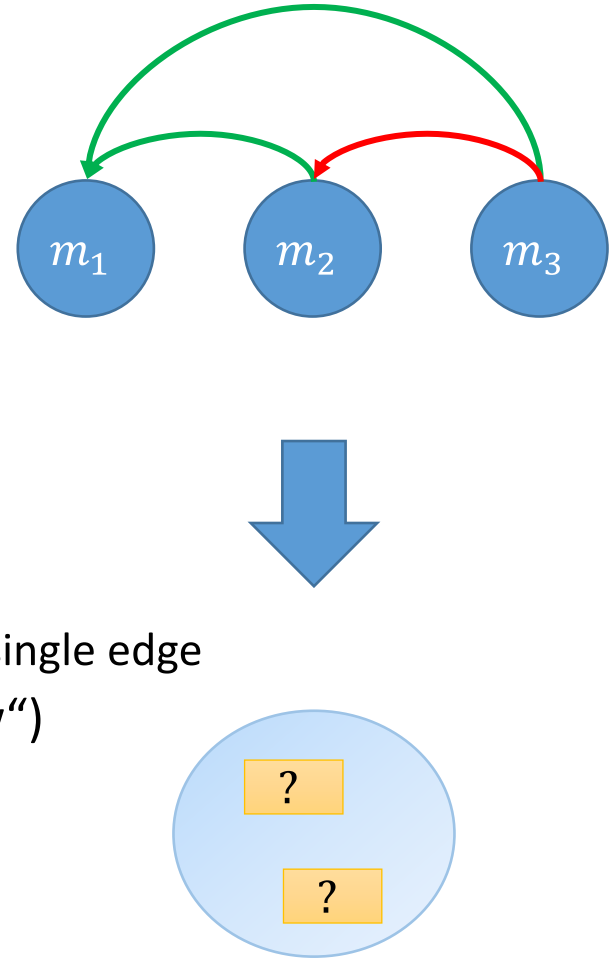
m_1 m_2 m_3 m_4 m_5

$$(m_5, m_4)\ vs.\ (m_5, m_3) \rightarrow (m_5, m_4)$$
$$(m_5, m_4)\ vs.\ (m_5, m_2) \rightarrow (m_5, m_2)$$
$$(m_5, m_2)\ vs.\ (m_5, m_1) \rightarrow (\mathrm{m}_5, m_2)$$

→ m_5 is resolved to m_2

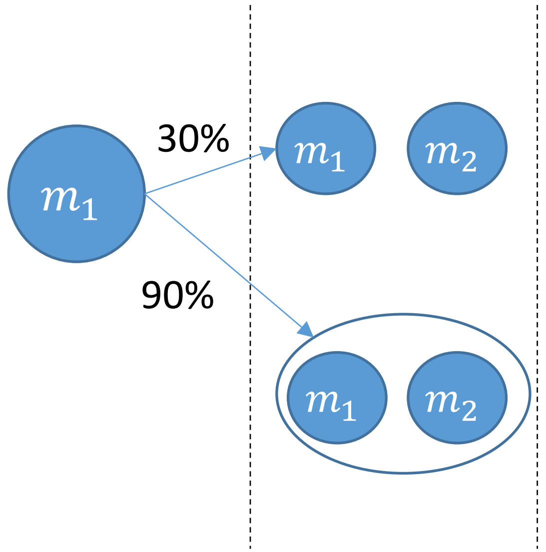
Mention-Ranking: Tournament Model

- Remaining problems:
  1. Mention-Ranking produces inconsistencies
     - $(m_3, m_2) \Rightarrow \text{Not–Coref}$
     - $(m_2, m_1) \Rightarrow \text{Coref}$
     - $(m_3, m_1) \Rightarrow \text{Coref}$
  2. ~~Mentions compete against each other~~
     - ➔This is solved at least partially!
  3. Local Features
     - Features can yet again only be calculated on the basis of a single edge
  4. How to recognize the first appearances? („discourse-new“)
     - There is always one pairing that remains …



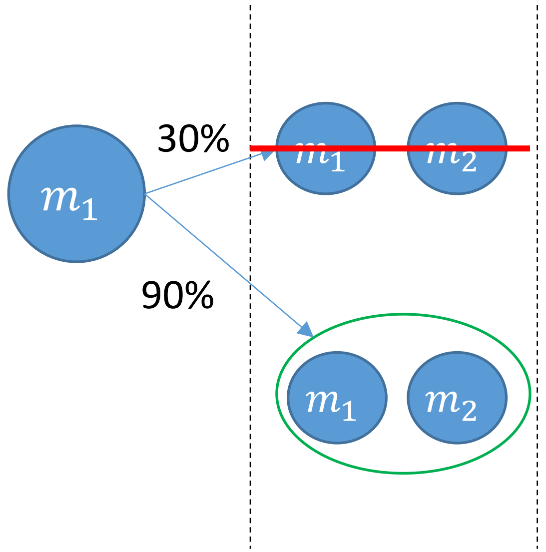
Machine Learning: Entity-Mention

- Let us improve on the problem of local features, by:
  - Creating the clusters in an iterative fashion
  - E.g. Luo 2006 using the Bell-Tree:



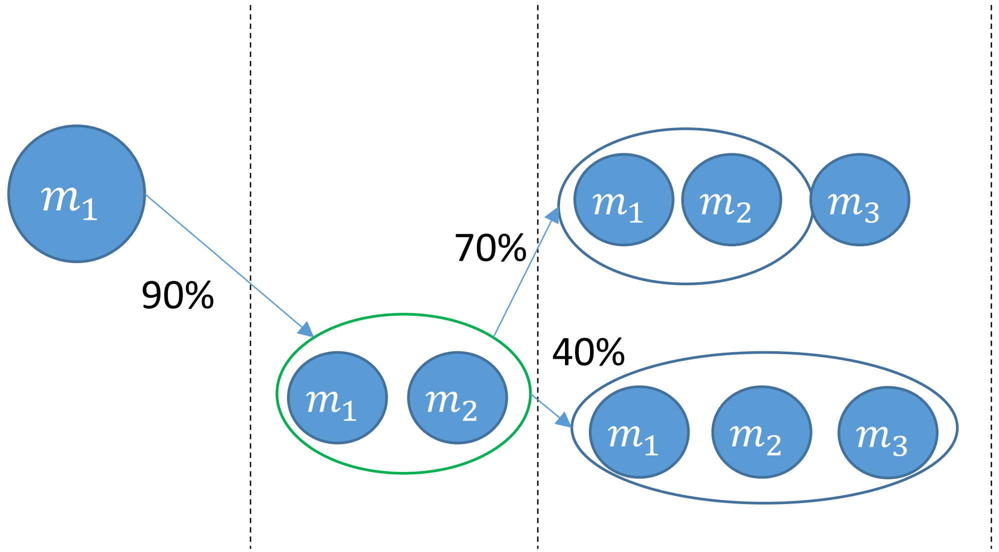
Machine Learning: Entity-Mention

- Let us improve on the problem of local features, by:
  - Creating the clusters in an iterative fashion
  - E.g. Luo 2006 using the Bell-Tree:



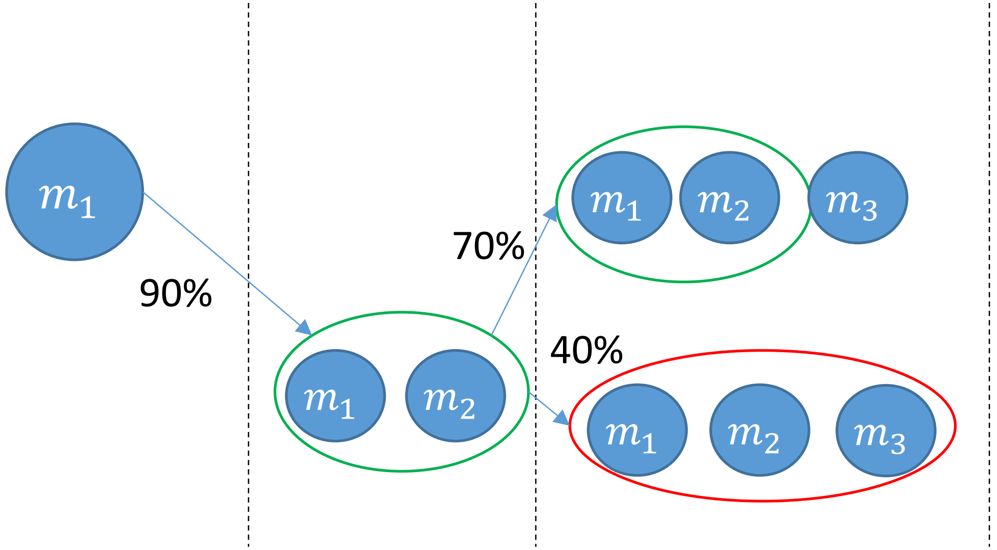
Machine Learning: Entity-Mention

- Let us improve on the problem of local features, by:
  - Creating the clusters in an iterative fashion
  - E.g. Luo 2006 using the Bell-Tree:



Machine Learning: Entity-Mention

- Let us improve on the problem of local features, by:
  - Creating the clusters in an iterative fashion
  - E.g. Luo 2006 using the Bell-Tree:



Machine Learning: Entity-Mention

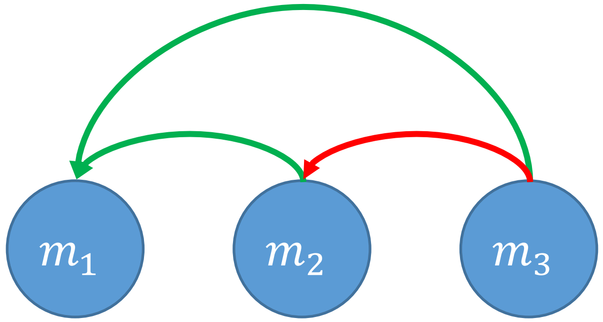
- Clusters are now built in an iterative fashion, mentions are resolved from the beginning of a text up to the end of a text (similar as to how a human would do it)

- We will always only keep the best solution
  ➔ State-space shrinks to a manageable size!

- Instances will be $(cluster_i, m_k)$
  - So we can now access features of the (partial) cluster and a mention!

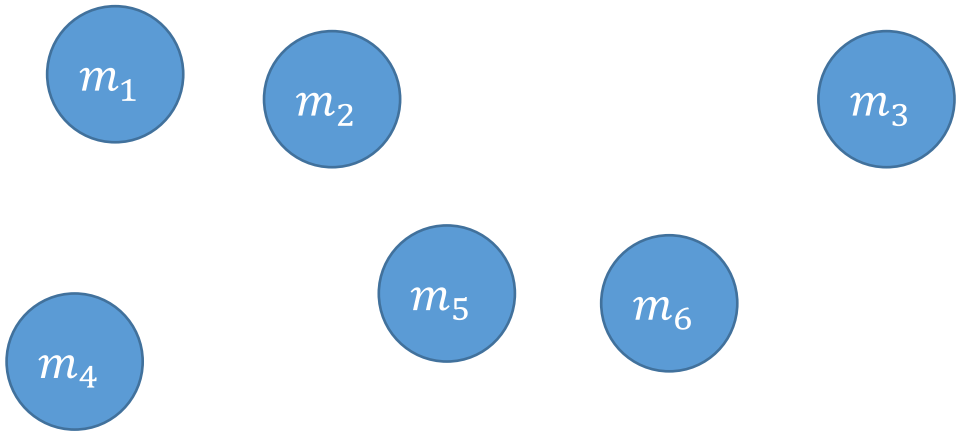
Machine Learning: Entity-Mention

- Remaining problems:
  1. ~~Mention-Ranking produces inconsistencies~~
     - $(m_3, m_2) \Rightarrow \text{Not–Coref}$
     - $(m_2, m_1) \Rightarrow \text{Coref}$
     - $(m_3, m_1) \Rightarrow \text{Coref}$
  2. Mentions do not compete against each other!
  3. ~~Local Features~~
  4. How to recognize the first appearances? („Discourse-new“)
     - There is always one pairing that remains …



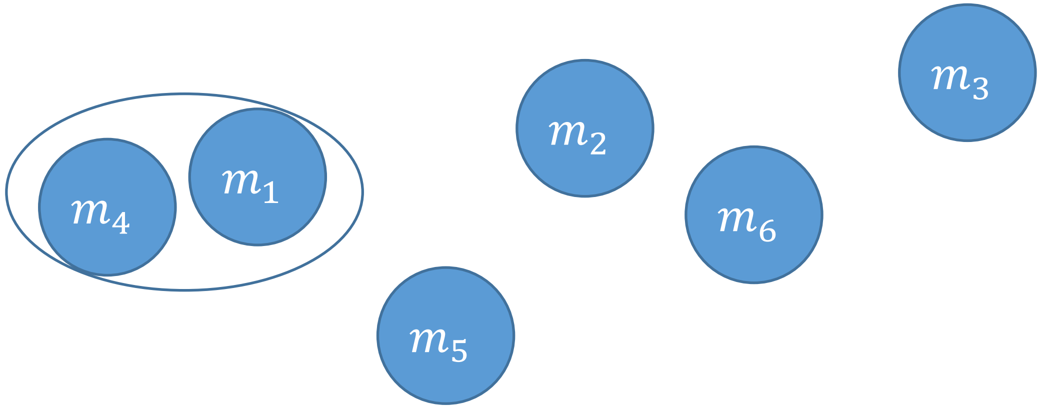
Machine Learning: Cluster-Ranking

- Combine the strengths of the **Mention ranking** model with the strengths of the **Entity-mention** model
- For example by using a hierarchical clustering (e.g. HAC)
- Start with all mentions being their own „cluster“



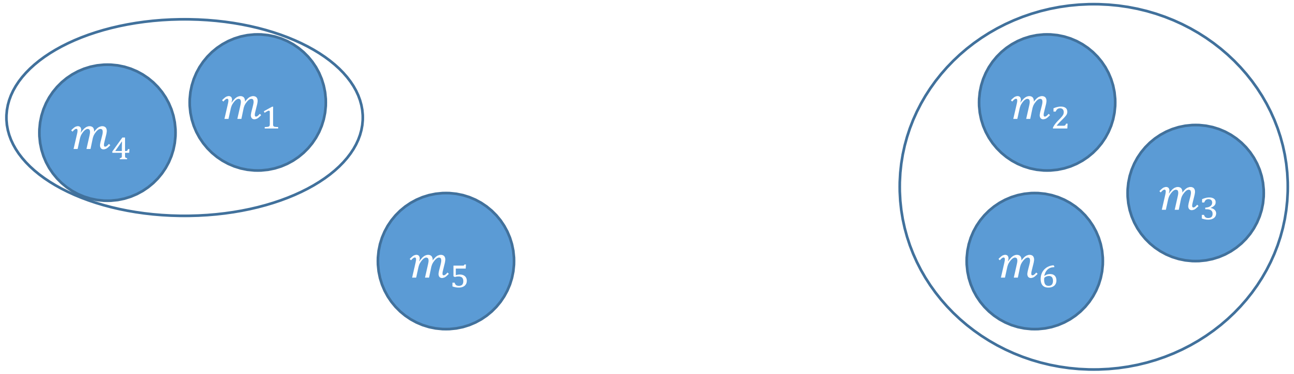
Machine Learning: Cluster-Ranking

- Start with all mentions being their own „cluster“
- Use the tournament model to get the best pairing for each mention
  - ➔ But only apply the one which is the best according to some criteria



Machine Learning: Cluster-Ranking

- Start with all mentions being their own „cluster“
- Use the tournament model to get the best pairing for each mention
  - ➔ But only apply the one which is the best according to some criteria
- Repeat until there is no more combination that exceeds a certain threshold



Machine Learning: Cluster-Ranking

- Remaining Problems:
  1. ~~We produce inconsistencies~~
  2. ~~Mentions compete against each other~~
  3. ~~Local Features~~

     ➔ We can now calculate features using pairs of clusters!
  4. How to recognize the first appearances? („Discourse-new“)
     - There is always one pairing that remains …

There is one issue remaining!

Machine Learning: Anaphoricity

- The decision, whether a mention has any antecedent in the text is called „Anaphoricity“-Problem

- All ranking approaches need this kind of information!

- Two typical ways to solve this:
  1. Create a dummy mention: if the dummy wins the tournament, then we do not assign any antecedent! (**Joint-Approach J**)
  2. A separate classifier decides, whether a mention is resolved at all! (**Pipeline-Approach P**)

Recap: Machine Learning for Coreference

- In this lecture we presented different approaches to keep the problem of Coreference Resolution tractable
  - We did this by finding an analogy to the task of Dependency Parsing

- This resulted in four models, that are independent of the classifier that is used:
  - Mention-Pair
  - Mention-Ranking
  - Entity-Pair
  - Cluster-Ranking

Even though the more complicated models tend to outperform the more basic models, there is no solution to Coreference Resolution as of now!

Evaluation

- Coreference Resolution produces a „clustering“
- For evaluating clusterings, measures such as the MUC metric are addressed in detail in the chapter "Evaluation"