

# **The Importance of High-Quality Input for Word Sense Disambiguation: An Application-Oriented Evaluation of Part-of-Speech Taggers**

Tanja Gaustad  
Humanities Computing  
University of Groningen, The Netherlands  
tanja@let.rug.nl  
<http://www.let.rug.nl/~tanja>

# Overview

- Introduction
- Presentation of Part-of-Speech (PoS) Taggers and Stand-Alone Results
- Word Sense Disambiguation (WSD) System for Dutch
  - \* Maximum Entropy classification
  - \* Corpus, Corpus Preparation, and System Settings
- Results of PoS Taggers in WSD System
- Evaluation

# Introduction

- Certain NLP tools used as subcomponent or pre-processor of complex system, e.g. PoS taggers
- Subcomponents of complex systems influence final results  
⇒ Application-oriented evaluation needed
- PoS taggers as a subcomponent of a WSD system for Dutch
- Hypothesis: more accurate subcomponents give better results in complex system

## Comparison of PoS Taggers

- Comparison of 3 PoS taggers:
  - \* Hidden Markov Model (HMM) Tagger
  - \* Memory-Based (MBT) Tagger
  - \* Transformation-Based (TBL) Tagger
- Dutch Eindhoven Corpus (760,000 words) used for training and stand-alone evaluation
- Limited WOTAN tag set with 48 tags used (original WOTAN tag set: 233 tags)
- In WSD application, only 12 tags considered (main PoS categories)

## HMM Tagger

- Trigram HMM tagger: each state = previous 2 PoS tags
- Two relevant probabilities:  $P(t_i|t_{i-2}t_{i-1})$  and  $P(w_i|t_i)$
- Training with forward-backward algorithm for each tag:

$$P(t_i = t) = \alpha_i(t)\beta_i(t)$$

$\alpha_i(t)$  = total probability of all paths through model ending at tag  $t$  at position  $i$

$\beta_i(t)$  = total probability of all paths starting at tag  $t$  in position  $i$  continuing to the end

## HMM Tagger II

- **Smoothing:** variant of linear interpolation
  - \* take into account lower order models
  - \* assign weights to each model to capture relative importance
- **Unknown words**
  - \* Heuristic rule for recognizing names (capitalized words = N)
  - \* Set of FS automata find possible tags based on suffixes

# MBT Tagger

- Based on Memory-Based Learning (extension of  $k$ -Nearest Neighbor approach)
- Two components
  - \* memory-based learner
  - \* similarity-based classification
- Extraction of 3 data structures from annotated corpus
  - \* lexicon
  - \* known words case base
  - \* unknown words case base

## MBT Tagger II

- Lexicon lookup
  - \* determine context
  - \* if found, get lexical representation
  - \* if not found, compute lexical representation based on form
- Classification
  - \* compute similarity test examples – examples in memory
  - \* extrapolate category of test example based on most similar example(s)

## MBT Tagger III

- Information used
  - \* **Known words:** preceding two tags and words, ambiguous tag and word to the right  
classification: IGTREE algorithm with one nearest neighbour
  - \* **Unknown words:** preceding tag, ambiguous tag to right, first and last three letters of ambiguous word  
classification: IB1 algorithm with 9 nearest neighbours

# TBL Tagger

- Main components
  - \* specification of admissible transformations
  - \* learning algorithm
- Initial step: assign a tag to each word independent of context
  - \* **Known words:** most likely tag determined by maximum likelihood estimation from training corpus
  - \* **Unknown word:** tag first N, then adapted via lexical rules learned during training based on local properties of unknown word (e.g. suffix)

## TBL Tagger II

- Second step: context rules adapt initial tags (where necessary)
- Contextual transformation rules and order of application selected by learning algorithm during training
- Dutch TBL tagger: 250 lexical rules, 300 contextual rules

## Stand-Alone Results

PoS Tagger	Accuracy
TBL	94.20
HMM	95.93
MBT	<b>96.21</b>

- Evaluated on Eindhoven corpus (80-10-10 split)
- MBT performs best, closely followed by HMM, TBL least accurate
- If hypothesis correct, ranking of PoS taggers should be the same when integrated into the WSD system
- Expectation might be falsified by possible corpus dependency of PoS taggers (capacity to generalize)

## WSD System for Dutch

- Semantic lexical ambiguity major problem in NLP (e.g. MT, IR)
- WSD = attribute correct sense(s) to words in context
- WSD system used here
  - \* Supervised, corpus-based
  - \* Combination of statistical classification with linguistic information
  - \* Intuition: (high quality) linguistic information beneficial for WSD
- Dutch data needs morpho-syntactic and semantic disambiguation

## Maximum Entropy classification

- Maximum entropy = general technique to estimate probability distributions
- Use Features extracted from labeled training data to derive constraints for model
- Constraints characterize class-specific expectations for distribution
- Distribution should maximize entropy **and** model should satisfy constraints imposed by training data

## Maximum Entropy classification II

$$p(c|x) = \frac{1}{Z} \exp \left( \sum_i \lambda_i f_i(x, c) \right)$$

$f_i(x, c)$  = # of times feature  $i$  used to find class  $c$  for event  $x$   
 $\lambda_i$  = maximize likelihood of training data and entropy of  $p$

- Training: weight  $\lambda_i$  for each feature  $i$  present in the training data computed and stored
- Testing: sum of weights  $\lambda_i$  of all features  $i$  found in the test instances computed for each class  $c$  and class with highest score chosen

## Maximum Entropy classification III

Main advantages:

- Property functions take into account any information which might be useful for disambiguation
- Dissimilar types of information can be combined into single model for WSD
- No independence assumptions (as in e.g. a Naive Bayes algorithm) necessary

## Corpus and Corpus Preparation

- Training section of Dutch SENSEVAL-2 corpus (120,000 tokens)
- Procedure to build classifiers
  - \* Lemmatize and PoS tag corpus
  - \* Extract all instances for each ambiguous wordform and lemma
  - \* Transform instances into feature vector, e.g.  
aarde N gat in de , zodat het aarde\_grond
  - \* Build classifier for each ambiguous wordform or lemma

# Classifiers

- Grouping of ambiguous words based on either same lemma or same wordform
- Comparison of classifiers (based on different feature sets)
- Basic classifiers
  - \* Wordforms: lemma and context
  - \* Lemmas: wordform and context
- Classifiers including PoS
  - \* Wordforms: lemma, PoS, and context
  - \* Lemmas: wordform, PoS, and context

# System Settings

- Settings
  - \* Context  $\pm 3$  words
  - \* Only context within same sentence
  - \* Frequency threshold of 10
  - \* Context = bag of words (independent of position relative to ambiguous wordform/lemma)
- 1,364 ambiguous lemmas  $\Rightarrow$  622 classifiers  
952 ambiguous wordforms  $\Rightarrow$  486 classifiers

## Results of PoS Taggers in WSD System

<b>Base: Wordforms</b>			
Feature set	Accuracy		
baseline	76.70		
lemma, con. words (basic)	80.81		
lemma, con. lemmas (basic)	80.52		
	TBL	HMM	MBT
lemma, pos, con. words	81.67	81.67	<b>81.89</b>
lemma, pos, con. lemmas	81.42	81.36	<b>81.67</b>

<b>Base: Lemmas</b>			
baseline	73.41		
word, con. words (basic)	82.52		
word, con. lemmas (basic)	82.25		
	TBL	HMM	MBT
word, pos, con. words	83.32	83.34	<b>83.46</b>
word, pos, con. lemmas	83.06	83.05	<b>83.30</b>

## Results of PoS Taggers in WSD System II

- Leave-one-out approach: every data item used as test item once, classifier trained on remaining items
- Basic classifiers perform better than frequency baseline
- Adding more information improves results (basic intuition behind WSD system)
- MBT PoS tags working best
- Surprise: HMM and TBL (almost) equal performance

## Evaluation

- In stand-alone results and PoS tags in WSD data, MBT and HMM closer together, TBL really different
- Integrated in to WSD system, HMM and TBL close together
- Possible explanation: difference training corpus – WSD data  
⇒ HMM tagger no longer performing better on WSD data than TBL tagger
- Heuristics for unknown words?

## Conclusion and Future Work

- More accurate PoS input yields better results
- But: need not always be the case  
⇒ Possible corpus dependency
- Future work
  - \* include PoS of context
  - \* optimize settings of PoS taggers