Classifyer for Morphology

Uwe Quasthoff

Universität Leipzig
Institut für Informatik
quasthoff@informatik.uni-leipzig.de

General Task

Decompose words into known parts (here: stems, affixes, other words).

We can use

- Similarities at the end of words (for lemmatization and compound splitting) and
- Similarities at the beginning of words (for compound splitting).

Problem: Irregularities.

Lemmatization

In the typical case (i.e. for regular words with POS N,A,V) there is a main form from which the others are derived.

In GERMAN:

For N and A we just drop a suffix. For V, we additionally add *en*.

Examples:

N: Schulen $\rightarrow$ Schule, Teppichen $\rightarrow$ Teppich

A: schönster $\rightarrow$ schön, grünen $\rightarrow$ grün

V: lernte $\rightarrow$ lernen

Irregularities in German Lemmatization

Frequent irregularities:

- N: Umlauts: Häuser → Haus
- N: irregular inflection: Extrema → Extremum, Brochitiden → Bronchitis
- A: Umlauts: jünger → jung
- V: irregular inflection: singen, sang, gesungen
- V: *ge-* and *zu-*: gelaufen → laufen,
  fortgelaufen → fortlaufen, fortzulaufen → fortlaufen, zugelaufen → ????
  gegessen → essen
- More difficult: zugelaufen → ????, entgegengelaufen → ????

Singular words:

- N: Säle → Saal, Köge → Koog, Feuerwehrleute → Feuerwehrmann

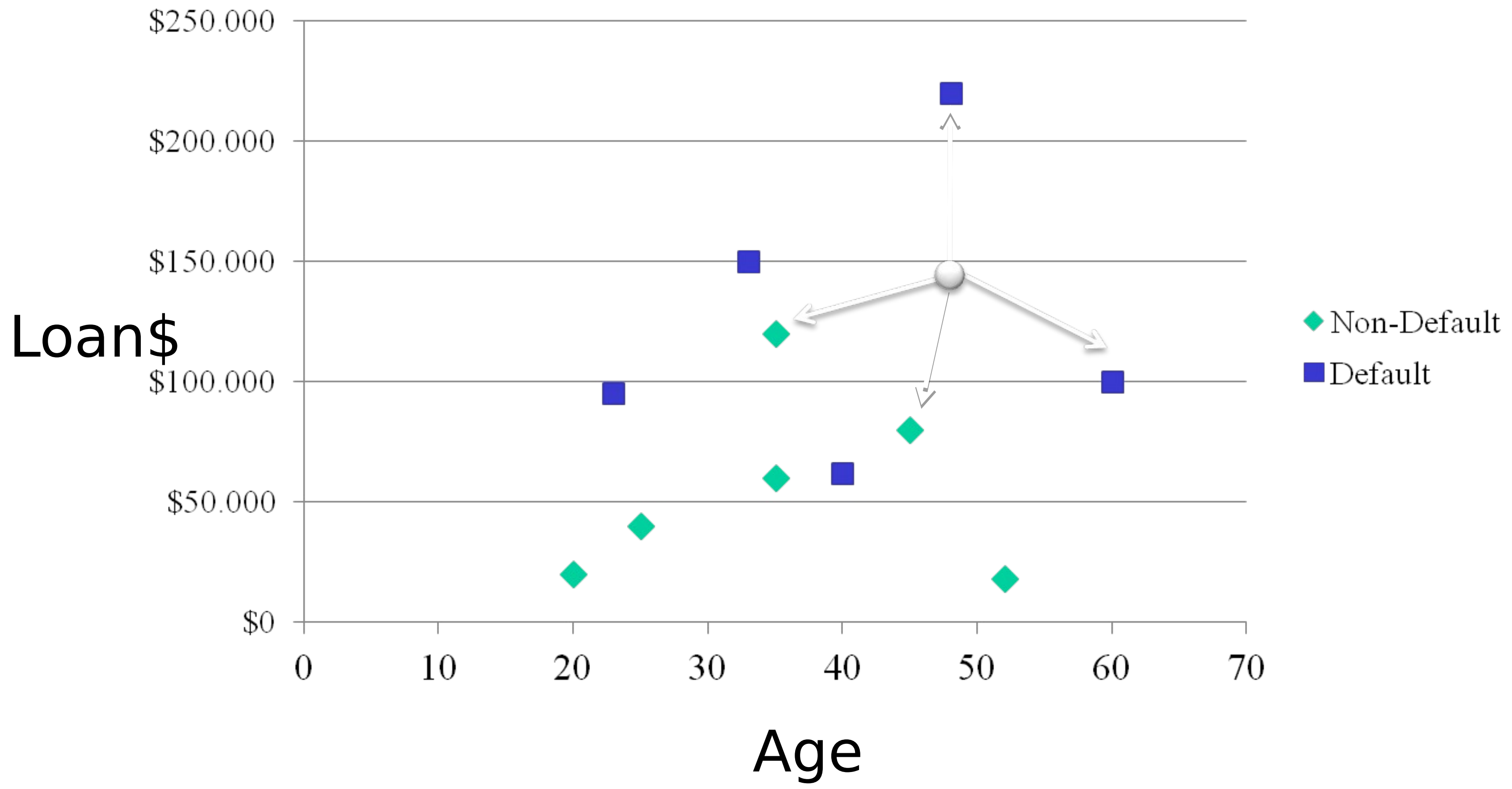
General Idea of Instance-based Learning

- Learning: store all the data instances
- Performance:
  - when a new query instance is encountered
    - retrieve a similar set of related instances from memory
    - use to classify the new query

KNN - Definition

KNN (K-Nearest Neighbors) is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure

KNN Classification



KNN – Algorithmus

- If K=1, select the nearest neighbor
- If K>1,
  - For classification select the most frequent neighbor.
  - For regression calculate the average of K neighbors.

Pros and Cons of Instance Based Learning

- Pros
  - Can construct a different approximation to the target function for each distinct query instance to be classified
  - Can use more complex, symbolic representations
- Cons
  - Cost of classification can be high
  - Uses all attributes (do not learn which are most important)

k-nearest neighbor (knn) learning

- Most basic type of instance learning
- Assumes all instances are points in n-dimensional space
- A distance measure is needed to determine the “closeness” of instances
- Classify an instance by finding its nearest neighbors and picking the most popular class among the neighbors

Important Decisions

- Distance measure
- Value of k (usually odd)
- Voting mechanism
- Memory indexing

High-Dimensional Near Neighbors

- **Nearest Neighbors Data Structure**
  - Given – N points $P=\{p_1, \ldots, p_N\}$ in metric space (M,D)
  - Queries – “Which point $p \in P$ is closest to point q?”
  - Complexity – Tradeoff preprocessing space with query time
- **Applications**
  - vector quantization
  - multimedia databases
  - data mining
  - machine learning
  - …

The *More Data* effect

- *There's no data like more data* (speech recognition motto)
- Banko and Brill (2001): confusibles
  - Differences between algorithms flip or disappear
  - Differences between representations disappear
  - Growth of curve seems log-linear (constant improvement with exponentially more data)
  - Explanation sought in "Zipf's tail"

Banko and Brill (2001)

- Demonstrated on {to,two,too} using 1M to 1G examples:
  - Initial range between 3 classifiers at
    - 1M: 83-85%
    - 1G: 96-97%
  - Extremely simple memory-based classifier (one word left, one word right):
    - 86% at 1M, 93% at 1G
    - apparent constant improvement on log-growth

Could words replace PoS?

Simple intuition:

- PoS disambiguate *explicitly*

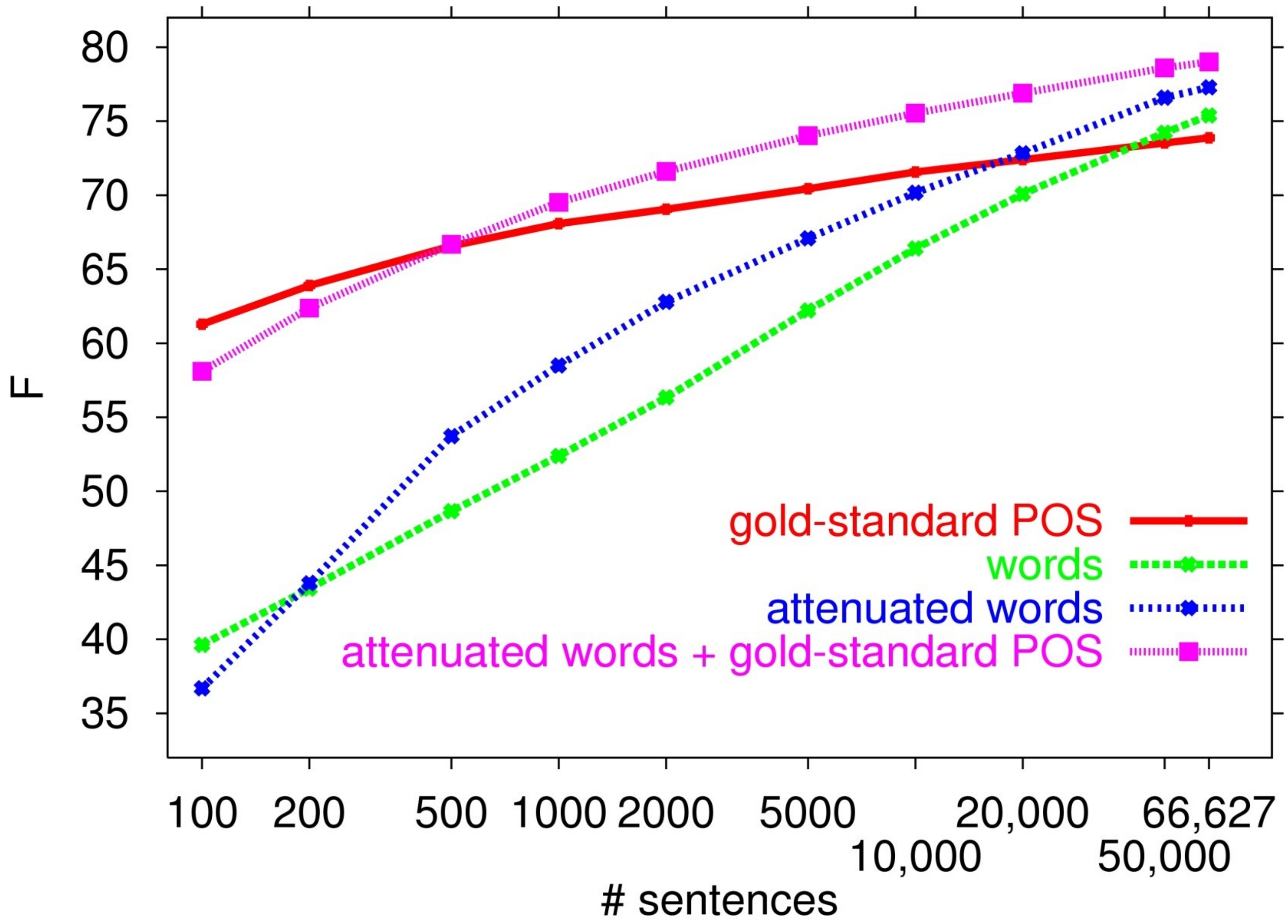
  suspect-N vs suspect-V

- words disambiguate *implicitly*

  … the suspect …

  … we suspect …

Results: learning curves



attenuated words: Seltene Wörter werden ersetzt durch ein Label MORPH- und die letzten beiden Buchstaben, also z.B. developments => MORPH-ts

Outline

NLP Methods, Resources and Applications

- Text Segmentation
- Part of Speech Tagging
- Stemming
- Lemmatization
- Syntactic Parsing
- Named Entity Recognition
- Term Extraction and Terminology Data Management Tools
- Language Identification
- Statistical Language Modelling Toolkits

Generalizing lexicon

- A memory-based morphological analyzer is
  - A lexicon: 100% accurate reconstruction of all examples in training material
  - At the same time, capable of processing **unseen** words
- In essence, unseen words are the only problem remaining

EBL: Summary

- EBL methods base decisions on similarity to specific past instances rather than constructing abstractions
- Exemplar-based methods abandon the goal of maintaining concept “simplicity”
- Consequently, they trade decreased learning time for increased classification time.
- They store all examples and, thus, all exceptions too. This seems to be very important for NLP tasks. “***Forgetting Exceptions is Harmful in NL Learning***” (Daelemans et al. 99)

EBL: Summary

- Important issues are:
  - Defining appropriate distance metrics
  - Representing appropriately example attributes
  - Efficient indexing of training cases
  - Handling irrelevant features

Memory-based learning and classification

- Learning:
  - Store instances in memory
- Classification:
  - Given new test instance X,
  - Compare it to all memory instances
    - Compute a distance between X and memory instance Y
    - Update the top *k* of closest instances (nearest neighbors)
  - When done, take the majority class of the *k* nearest neighbors as the class of X

TIMBL

- TiMBL: Tilburg Memory Based Learner (Daelemans et. al. 1998-2012), https://languagemachines.github.io/timbl/
- Available for research and education
- Lazy learning, extending k-NN and IB1
- Especially fro NLP tasks
- Can deal with millions of instances
- With several thousand attributes

Current practice

- Default TiMBL settings:
  - k=1, Overlap, Gain Ratio, no distance weighting
  - Work well for some morpho-phonological tasks
- Rules of thumb:
  - Combine MVDM with bigger k
  - Combine distance weighting with bigger k
  - Very good bet: higher k, MVDM, Gain Ratio, distance weighting
  - Especially for sentence and text level tasks

TIMBL - Optionen

Command line:

```
      timbl -f data-file {-t test-file} [options]

      -a n          : algorithm
        0 or IB1    : IB1  (default)  knn, k=1
        1 or IG     : IGTree                  decision tree, IG
        2 or TRIBL  : TRIBL                   first decision tree, later knn
        3 or IB2    : IB2                     decision tree wuth Pruning
        4 or TRIBL2: TRIBL2                   first IB2, later knn
```

TIMBL – data format

Standard format for training and testing: Attributes in columns (separated ba blanks), last column is class.

Typical example:

```
= = = T i s c h e s -2
K o n v e r t e r n -1
= = = = S c h u l e -1
= = = = B e z ü g e -3ug
```

Output: Additional column with result class

```
= = = T i s c h e s -2 -2
K o n v e r t e r n -1 6ort
= = = = S c h u l e -1 -1
= = = = B e z ü g e -3ug -3ug
```

Classification tasks for words using their letter sequence

Goal: One class per word (What to do with ambiguities?)

Task:

- Lemmatization: base form for word
- Detection of prefixes and suffixes
- POS
- Gender for surnames

Decomposition tasks (multiple splitting points for one word)

- Compound decomposition
- Morphological decomposition
- Hyphenation

Classification using letters (left to right)

Example: 14 attributes for words containing max. 14 characters.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	class
J	o	n	a	t	h	a	n	=	=	=	=	=	=	m
M	a	x	i	m	i	l	i	a	n	=	=	=	=	m
E	m	i	l	i	a	=	=	=	=	=	=	=	=	w
A	n	t	o	n	i	a	=	=	=	=	=	=	=	w

This task cannot use the last character as attribute, because the last character has no unique attribute number.

Classification using letters (right to left)

-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	class
=	=	=	=	=	=	J	o	n	a	t	h	a	n	m
=	=	=	=	M	a	x	i	m	i	l	i	a	n	m
=	=	=	=	=	=	=	=	E	m	i	l	i	a	w
=	=	=	=	=	=	=	A	n	t	o	n	i	a	w

Better, but no fixed attribute for first character.

Classification using letters (interlaced)

- Fixed attribute numbers for first and last characters.

7	-7	6	-6	5	-5	-4	-4	3	-3	2	-2	1	-1	class
=	=	=	=	=	=	a	t	n	h	o	a	J	n	m
=	=	=	=	m	i	i	l	x	i	a	a	M	n	m
=	=	=	=	=	=	=	=	i	l	m	i	E	a	w
=	=	=	=	=	=	=	o	t	n	n	i	A	a	w

Experiment: First names by gender

1. experiment: ca. 2000 frequent names, well known and mostly „very German“

Nico (??), Hannes (??), Daniela (??), Marianne (??), Annette (??), Olaf (??), Christopher (??), Mohammed (??), Vaclav (??), Marcus (??), Katrin (??), Heike (??), Kerstin (??), Uli (??), Torsten (??), Francisco (??), Helga (??), Albrecht (??), Gabriele (??), Heiko (??), Ingo (??), Bastian (??), Sergej (??), Elke (??), Michail (??), Wilfried (??), Herrmann (??), Erika (??), Hannelore (??), Björn (??), Leo (??), Johanna (??), Sonja (??), Dietrich (??), Rafael (??), Kathrin (??), Stefanie (??), Tanja (??), Sophie (??), Bettina (??), Claus (??), Abbas (??), Ruth (??), Nils (??), Alois (??), Achim (??), Antonio (??), Christa (??), Gordon (??), Johnny (??), …

2. experiment: all 22.000 Baby names from Germany in 2012

Josephin (??), Joris (??), İyad (??), Jorit (??), Jorik (??), Juleen-Summer (??), Curly-Sue (??), Jorin (??), Jorim (??), Cherry (??), Nazlican (??), Elfida (??), Jorid (??), Chada (??), Arnaud (??), Eva-Amaya (??), Donovan (??), Elgin (??), Alexna (??), Chizaram (??), Skye (??), Thalisa (??), Sror (??), Souhayb (??), Léandro (??), Jiangruimiao (??), Roxolana (??), Persis (??), Gavin (??), Nora-Sophia (??), Martys (??), Nora-Sophie (??), Awen (??), Eryk (??), Fadime (??), Elja (??), Benthe (??), Happi (??), Liuan (??), Happy (??), Aleydis (??), Ewing (??), Evgeni (??), Oscar (??), Vitalij (??), Shamsul (??), Brenda (??), Marilou (??), Aissatou (??), Steven (??), Jincy (??), …

Experiment: First names by gender

1. experiment: ca. 2000 frequent names, well known and mostly „very German“

Nico (m), Hannes (m), Daniela (w), Marianne (w), Annette (w), Olaf (m), Christopher (m), Mohammed (m), Vaclav (m), Marcus (m), Katrin (w), Heike (w), Kerstin (w), Uli (m), Torsten (m), Francisco (m), Helga (w), Albrecht (m), Gabriele (w), Heiko (m), Ingo (m), Bastian (m), Sergej (m), Elke (w), Michail (m), Wilfried (m), Herrmann (m), Erika (w), Hannelore (w), Björn (m), Leo (m), Johanna (w), Sonja (w), Dietrich (m), Rafael (m), Kathrin (w), Stefanie (w), Tanja (w), Sophie (w), Bettina (w), Claus (m), Abbas (m), Ruth (w), Nils (m), Alois (m), Achim (m), Antonio (m), Christa (w), Gordon (m), Johnny (m), …

2. experiment: all 22.000 Baby names from Germany in 2012

Josephin (w), Joris (m), İyad (m), Jorit (m), Jorik (m), Juleen-Summer (w), Curly-Sue (w), Jorin (m), Jorim (m), Cherry (w), Nazlican (w), Elfida (w), Jorid (w), Chada (w), Arnaud (m), Eva-Amaya (w), Donovan (m), Elgin (w), Alexna (w), Chizaram (w), Skye (w), Thalisa (w), Sror (m), Souhayb (w), Léandro (m), Jiangruimiao (w), Roxolana (w), Persis (w), Gavin (m), Nora-Sophia (w), Martys (m), Nora-Sophie (w), Awen (w), Eryk (m), Fadime (w), Elja (w), Benthe (w), Happi (w), Liuan (m), Happy (w), Aleydis (w), Ewing (m), Evgeni (m), Oscar (w),

Vitalij (w), Shamsul (w), Brenda (w), Marilou (w), Aissatou (w), Steven (m), Jincy (w), …

Design of different experiments

Algorithm: Decision tree (IB1) and kNN with k=1, 3 or 5

Data sets:

•2000 frequent surnames

•2000 out of all the 20.000 surnames

•All 20.000 surnames

Always 90% training data and 10% test data

Problem with small data sets:

In the case of 10% test data (200 instances) and 90% quality, there are about 20 misclassified instances. Results based on such small numbers are not very reliable.

Left2right, right2left or interlaced?

Experiment: 2000 frequent names, kNN with k=1

(similar results for the other algorithms)

Attribut selection	Quality (%)
right2left	83,8
left2right	76,2
interlaced	84,8

Attribute weights

Using information gain

Attribut selection	Attribute weights
right2left	0, 0, … 0, .01, .10, .30, .34, .48, .41, .56, .72, .67, 1,10, 3.31
left2right	.08, .10, .10, .07, .10, .10, .18, .31, .63, .56, .49, .48, 0, 0 ,…, 0
interlaced	0, 0, … 0, .48, .49, .42, .38, .13, .17, .09, .13, .09, .28, .08, .84

Size of the data sets

Interlaced using kNN with k=1

Data	Quality (%)
2.000 frequent names	84,8
2.000 of 20.000 names	73,9
all 20.000 names	79,6

Observations:

1. Rare foreign names drop quality because of less regularity.
2. More training data help.

TimblWrapper

- TIMBL‘s input format is typically „bulky“
- Preparing data for Timbl rather sophisticated process
- Might be too complex for some users from the humanities
- Few typical input patterns for linguistic analyses on word level

-> Created a wrapper to generate input files from simple word lists

word	**class**
Tisches	-2
Flugzeuge	-1
Studenten	-2

Typical input patterns

- Single letters – focus on end of word

```
= = = T i s c h e s -2
= S t u d e n t e n -2
```

- Single letters – focus on beginning of word

```
T i s c h e s = = = -2
S t u d e n t e n = -2
```

- Prefixes and Suffixes – at the beginning and end of words

```
T Ti Tis Tisc Tisch Tisches isches sches ches hes es s -2
S St Stu Stud Stude udenten denten enten nten ten en n -2
```

`->` initially only this option is available

Starting the Wrapper

```
java -jar TimblWrapper.jar trainingdata testdata mode
```

- Training data: word list with classes to build a model
- Test data: word list with or without classes
- Mode:

  -c: classify (classes unknown)

  -t: test (classes known – compute precision of classifier)

- Output:
  - „testdata“.output: results of classification
  - „testdata“.summary: feature weights, precision and so on