



Mathematics for Linguists: Memory Based Learning

Peter Hendrix



Memory Based Learning

- What is Memory Based Learning?
- Classification technique based on the idea that intelligent behavior can be obtained by analogical reasoning, rather than by the application of abstract mental rules
- Alternative names: similarity-based learning, exemplar-based learning, instance based learning, lazy learning



Memory Based Learning

- Memory Based Learning models take a set of examples (features-value patterns and associated outcome classes) as input and produce a classifier that predicts class membership of new, previously unseen input patterns on the basis of similarity to examples in the training set
- Application domain: classification tasks with symbolic or numeric features and discrete, non-continuous classes



TiMBL

- TiMBL: Tilburg Memory Based Learner (Daelemans et al. (2010))
- Download from: <http://ilk.uvt.nl/timbl>
- Based on k-Nearest Neighbors (k-NN) algorithm

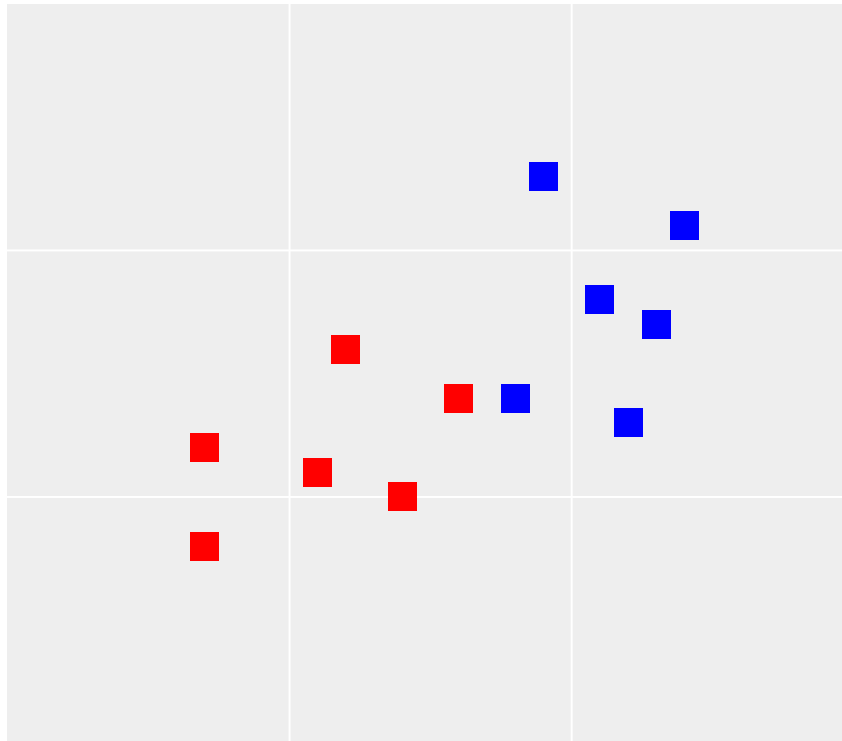


k-Nearest Neighbors

- Store all instances encountered during training in memory
- Present new instance during test
- Find the k most similar example(s) in the training data using some distance metric $\Delta(X, Y)$
- Assign the most frequent class within the set of most similar example(s) (the k -nearest neighbours) to the new instance

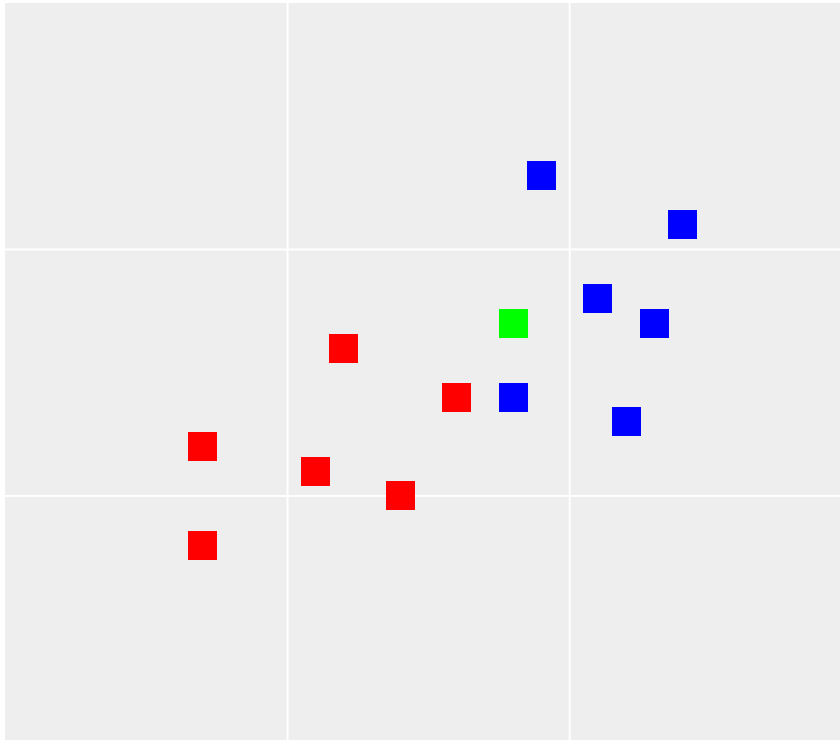


k-Nearest Neighbors



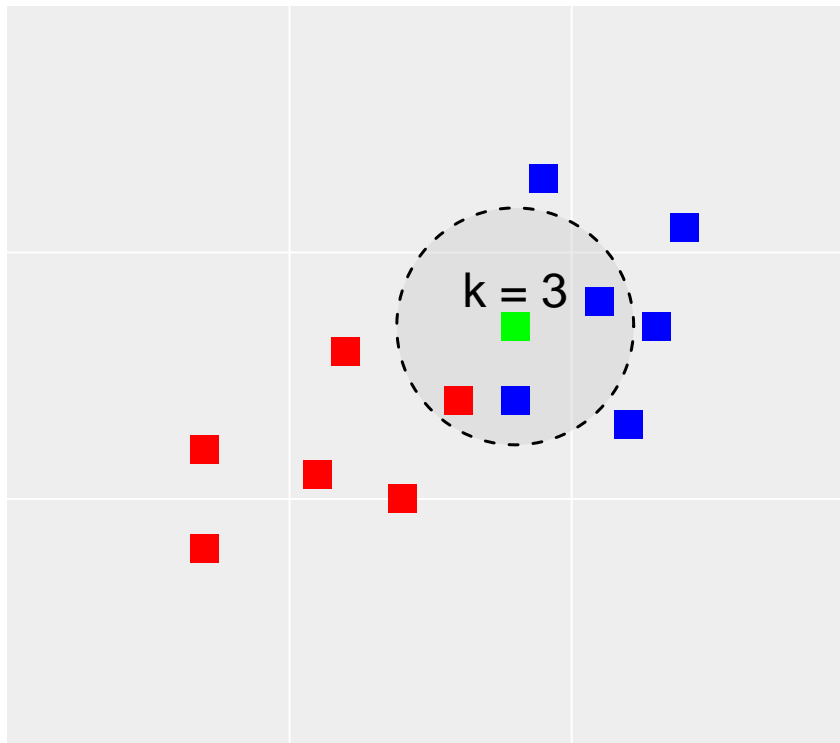


k-Nearest Neighbors



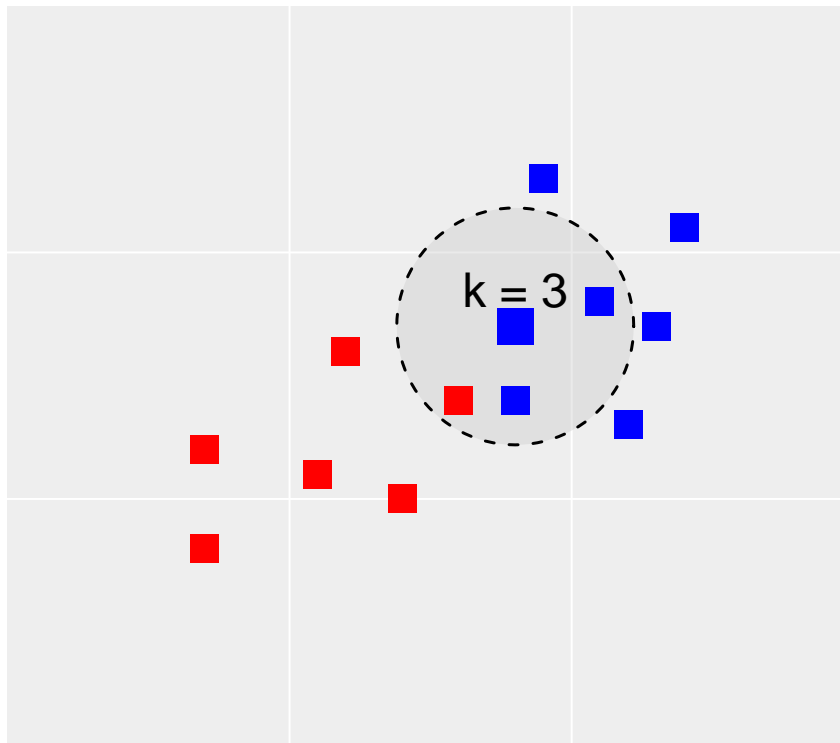


k-Nearest Neighbors





k-Nearest Neighbors





k-Nearest Neighbors

- Note: usually k refers to the number of neighbors taken into account
- In TiMBL k is the number of nearest **distances** taken into account
- With $k = 1$, therefore, TiMBL's nearest neighbor set can contain multiple training instances that are equally distant to the test stimulus



Classification in TiMBL

- 1) Define classification task
- 2) Learning phase
- 3) Performance phase:
 - a) Choose distance metric
 - b) Set k (number of nearest distances)
 - c) Choose how to extrapolate from nearest neighbors
 - d) Run classifier



Classification in TiMBL

- 1) Define classification task
- 2) Learning phase
- 3) Performance phase:
 - a) Choose distance metric
 - b) Set k (number of nearest distances)
 - c) Choose how to extrapolate from nearest neighbors
 - d) Run classifier



Classification in TiMBL

- Definition of classification task is driven by (linguistic) theory
- Three steps:
 - Define outcome classes: what do we want to predict?
 - Define features: which information might be relevant to predict the outcome classes?
 - Define feature-values: how do we want to encode the features?



Classification in TiMBL

- Example from TiMBL reference guide: **Dutch diminutive suffix**
- Dutch diminutives are formed by attaching a diminutive suffix to the base form of a noun
- The suffix shows variation in its surface form (allomorphy)



Classification in TiMBL

Dutch diminutive suffix allomorphy:

noun	english	form	suffix	class
huis	house	huisje	<i>-je</i>	<i>J</i>
man	man	mannetje	<i>-etje</i>	<i>E</i>
raam	window	raampje	<i>-pje</i>	<i>P</i>
woning	house	woninkje	<i>-kje</i>	<i>K</i>
baan	job	baantje	<i>-tje</i>	<i>T</i>



Classification in TiMBL

- Outcome classes: form of diminutive suffix
- Features (for each of last three syllables):
 - Stress
 - Onset
 - Nucleus
 - Coda



Classification in TiMBL

Feature encoding:

syllable	syllable	syllable	outc.	noun	english
+ b i =	- z @ =	- m A nt	J	biezenmand	bulrush basket
= = = =	= = = =	+ b l χ	E	big	piglet
= = = =	+ b K =	- b a n	T	baan	job
= = = =	+ b K =	- b @ l	T	bijbel	bible



Classification in TiMBL

- 1) Define classification task
- 2) **Learning phase**
- 3) Performance phase:
 - a) Choose distance metric
 - b) Set k (number of nearest distances)
 - c) Choose how to extrapolate from nearest neighbors
 - d) Run classifier



Classification in TiMBL

- During the learning phase the training data are stored in memory
- Importantly, **no abstraction or restructuring of information** occurs



Classification in TiMBL

Input format:

```
# =,=,=,=,+,k,e,=-,r,@,l,T
# =,=,=,=-,fr,i,=,+,z,I,n,E
# =,=,=,=,=,=,=,=,+,sn,},f,J
# =,=,=,=,+,l,I,=-,x,a,m,P
# =,=,=,=,=,=,=,=,+,tr,A,p,J
# =,=,=,=,+,k,E,rst,-,k,I,nt,J
# +,r,i,=-,j,a,=-,b,e,lt,J
# =,=,=,=-,v,I,n,+,j,E,t,J
# -,b,0,=,+,t,i,=-,n,@,=,T
# +,b,A,k,-,st,0,=-,p,@,r,T
# ...
# 2999 lines
```



Classification in TiMBL

- 1) Define classification task
- 2) Learning phase
- 3) **Performance phase:**
 - a) Choose distance metric
 - b) Set k (number of nearest distances)
 - c) Choose how to extrapolate from nearest neighbors
 - d) Run classifier



Classification in TiMBL

- During the performance phase new instances are classified based on the training data
- Prior to running the classifier we have to set a number of technical parameters



Classification in TiMBL

- 1) Define classification task
- 2) Learning phase
- 3) Performance phase:
 - a) Choose distance metric
 - b) Set k (number of nearest distances)
 - c) Choose how to extrapolate from nearest neighbors
 - d) Run classifier



Classification in TiMBL

- The most basic distance metric is the overlap metric:

$$\Delta(X, Y) = \sum_{i=1}^n \delta(x_i, y_i)$$

where:

$$\delta(x_i, y_i) = \begin{cases} \left| \frac{x_i - y_i}{\max_i - \min_i} \right| & \text{if numeric, else} \\ 0 & \text{if } x_i = y_i \\ 1 & \text{if } x_i \neq y_i \end{cases}$$



Classification in TiMBL

- More sophisticated distance metrics use:
 - a weighting scheme for feature relevance
 - graded similarity measures



Classification in TiMBL

- More sophisticated distance metrics use:
 - a weighting scheme for feature relevance
 - graded similarity measures



Classification in TiMBL

- Similarity metrics weighted for feature relevance are based on the idea that some features may be better predictors of class membership than others
- All features are equal, but some features are more equal than others:

$$\Delta(X, Y) = \sum_{i=1}^n w_i \delta(x_i, y_i)$$



Classification in TiMBL

- Establish the relevance of features by looking at which features are good predictors
- Examples:
 - Information Gain
 - Gain Ratio



Classification in TiMBL

- Information Gain (IG): feature weighting based on information theory
- Compute the difference in uncertainty (i.e.; entropy) between the situations without and with knowledge of the value of that feature:

$$w_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v)$$

where C is the set of class labels, $H(C)$ is the entropy of the class labels and V_i is the set of values for feature i

- $P(v)$ estimated from the relative frequencies of the feature values in the training data



Classification in TiMBL

- The more relevant a feature, the greater the information gain
- Problem: IG overestimates the relevance of features with large numbers of values
- Gain Ratio alleviates this problem through division by the entropy of the feature values:

$$w_i = \frac{H(C) - \sum_{v \in V_i} P(v) \times H(C|v)}{- \sum_{v \in V_i} P(v) \log_2 P(v)}$$



Classification in TiMBL

- More sophisticated distance metrics use:
 - a weighting scheme for feature relevance
 - **graded similarity measures**



Classification in TiMBL

- Overlap metric treats all feature values as equally dissimilar
- For the Dutch diminutive classification task we would like to use the information that “b” and “p” are phonetically more similar than “b” and “a”
- Graded similarity metrics measure the similarity of feature values based on the co-occurrence of feature values with outcome classes
- Example: modified value difference metric (MVDM)



Classification in TiMBL

- MVDM describes the similarity of two feature values v_1 and v_2 as:

$$\delta(v_1, v_2) = \sum_{i=1}^n |P(C_i|v_1) - P(C_i|v_2)|$$

where $C_{i...n}$ are the outcome classes

- Warning: MVDM may lead to suboptimal results when the data are sparse



Classification in TiMBL

- 1) Define classification task
- 2) Learning phase
- 3) Performance phase:
 - a) Choose distance metric
 - b) **Set k (number of nearest distances)**
 - c) Choose how to extrapolate from nearest neighbors
 - d) Run classifier



Classification in TiMBL

- Choosing k wisely allows for optimal performance
- Setting k too low will result in potentially informative exemplars not being taken into account
- Setting k too high will lead to uninformative exemplars being taken into account
- Choosing an appropriate value of k is an empirical issue



Classification in TiMBL

- Default setting of k in TiMBL is 1
- Given that TiMBL uses the k nearest **distances** this is often the optimal setting when using overlap distance metrics in classification tasks with a small number of features
- When using graded similarity measures $k = 1$ tends to restrict the set of nearest neighbors to a single exemplar
- When using MVDM it is therefore useful to experiment with higher values of k



Classification in TiMBL

- 1) Define classification task
- 2) Learning phase
- 3) Performance phase:
 - a) Choose distance metric
 - b) Set k (number of nearest distances)
 - c) Choose how to extrapolate from nearest neighbors
 - d) Run classifier



Classification in TiMBL

- Exemplars in set of nearest neighbors vote for the class of a new item
- Most straightforward voting scheme: **majority voting**
- Vote of each neighbor receives equal weight
- Class with the highest number of votes is chosen



Classification in TiMBL

- Alternative voting schemes use **distance-weighted voting**
- Votes are weighted for the distance between each neighbor and the test item: votes from nearby neighbors are deemed more important than votes from faraway friends
- Distance-weighted voting often outperforms majority voting
- Examples:
 - Inverse Linear distance-weighting
 - Exponential Decay distance-weighting



Classification in TiMBL

- Inverse Linear distance-weighting assigns a weight w_j to each neighbor that linearly decreases as the distance between the neighbor and the test item increases:

$$w_j = \begin{cases} \frac{d_k - d_j}{d_k - d_1} & \text{if } d_k \neq d_1 \\ 1 & \text{if } d_k = d_1 \end{cases}$$

where d_j is the distance of the j -th neighbor to the test item, d_1 the distance of the closest neighbor and d_k the distance of the furthest neighbor



Classification in TiMBL

- Shephard (1987): the relevance of a previous stimulus for the generalization to a new stimulus is an exponentially decreasing function of the distance between the new stimulus and the previous stimulus in a psychological space
- Exponential decay weighting function:

$$w_j = e^{-\alpha d_j^\beta}$$

where α and β are constants determining the slope and the power of the exponential decay function



Classification in TiMBL

- What if the voting results in a tie?
- TiMBL procedure for breaking ties:
 - 1) increase k to $k + 1$
 - 2) assign class that is most frequent in the training data
 - 3) randomly assign a class



Classification in TiMBL

- 1) Define classification task
- 2) Learning phase
- 3) Performance phase:
 - a) Choose distance metric
 - b) Set k (number of nearest distances)
 - c) Choose how to extrapolate from nearest neighbors
 - d) **Run classifier**



Classification in TiMBL

Reminder:

noun	english	form	suffix	class
huis	house	huisje	<i>-je</i>	<i>J</i>
man	man	mannetje	<i>-etje</i>	<i>E</i>
raam	window	raampje	<i>-pje</i>	<i>P</i>
woning	house	woninkje	<i>-kje</i>	<i>K</i>
baan	job	baantje	<i>-tje</i>	<i>T</i>

Classification task: predict allomorph on the basis of stress and phonological onset, nucleus and coda of the base noun



Classification in TiMBL

- Provide test data in same format as training data
- Run TiMBL



Classification in TiMBL

Input format test data:

```
# =, =, =, =, =, =, =, =, =, +, p, e, =, T
# =, =, =, =, +, k, u, =, -, b, l, u, m, E
# +, m, I, =, -, d, A, G, -, d, }, t, J
# -, t, @, =, -, l, |, =, -, G, @, n, T
# -, =, I, n, -, s, t, r, y, =, +, m, E, n, t, J
# =, =, =, =, =, =, =, =, =, +, b, r, L, t, J
# =, =, =, =, +, z, w, A, =, -, m, @, r, T
# =, =, =, =, -, f, u, =, +, d, r, a, l, T
# =, =, =, =, =, =, =, =, =, +, l, e, w, T
# =, =, =, =, +, t, r, K, N, -, k, a, r, t, J
# ...
# 950 lines
```



Classification in TiMBL

Run TiMBL:

```
# Timbl -f dimin.train -t dimin.test
#
# This uses the default parameter settings:
# IB1 (standard k-NN) algorithm (-a0)
# overlap similarity (-m0)
# Gain Ratio feature weighting (-dZ)
# k = 1 (-k1)
# no distance-weighting (-dZ)
#
# ...
# overall accuracy: 0.968421 (920/950)
```



Classification in TiMBL

- Default parameter settings usually give decent performance
- For Dutch diminutives: 96.84% of test items classified correctly
- Can parameter optimization further improve performance?



Classification in TiMBL

Run TiMBL with MVDM, $k = 5$ and no feature weighting:

```
# Timbl -a0 -mM -w0 -k5 -dZ -f dimin.train -t dimin.test  
# ...  
# overall accuracy: 0.977895 (929/950)
```



Classification in TiMBL

Run TiMBL with MVDM, $k = 5$ and no feature weighting:

```
# Timbl -a0 -mM -w0 -k5 -dZ -f dimin.train -t dimin.test  
# ...  
# overall accuracy: 0.977895 (929/950)
```

Might seem like a small improvement, but 26.67% less errors!



Classification in TiMBL

- Other useful diagnostics in the output:
 - Feature relevance (default)
 - Voting distributions (+v db)
 - Confusion matrix (+v cm)
 - Class statistics (+v cs)
 - Advanced statistics (+v as)



Classification in TiMBL

Feature relevance:

#	Feats	Vals	InfoGain	GainRatio
# 1		3	0.030971064	0.024891536
# 2		50	0.060860038	0.027552191
# 3		19	0.039562857	0.018676787
# 4		37	0.052541227	0.052620750
# 5		3	0.074523225	0.047699231
# 6		61	0.106044330	0.024471911
# 7		20	0.123486680	0.034953203
# 8		69	0.097198760	0.043983864
# 9		2	0.045752381	0.046816705
# 10		64	0.213887590	0.042844587
# 11		18	0.669704580	0.185070180
# 12		43	1.278076200	0.325371810



Classification in TiMBL

Run TiMBL with most relevant features only:

```
# Timbl -m0:I1-10 -f dimin.train -t dimin.test  
# ...  
# overall accuracy: 0.973684 (925/950)
```



Classification in TiMBL

Run TiMBL with most relevant features only:

```
# Timbl -m0:I1-10 -f dimin.train -t dimin.test  
# ...  
# overall accuracy: 0.973684 (925/950)
```

Removing less relevant features improved the classification performance of the model!



Classification in TiMBL

Voting distributions:

```
# =, =, =, =, =, =, =, =, =, +, pr, 0, p, J, J { E 3.00000, J 12.0000 }  
# =, =, =, =, =, =, =, =, =, +, w, e, t, J, J { J 2.00000 }  
# =, =, =, =, +, t, L, n, -, h, L, s, J, J { J 1.00000 }  
# =, =, =, =, =, =, =, =, =, +, t, L, n, T, T { T 1.00000 }  
# =, =, =, =, =, =, =, =, =, +, z, o, m, P, P { P 3.00000 }  
# +, d, a, =, -, m, @, s, -, kr, A, ns, J, J { J 1.00000 }  
# =, =, =, =, +, =, a, rd, -, m, A, n, E, E { E 2.00000 }
```



Classification in TiMBL

Confusion matrix:

#		T	E	J	P	K
#	-----					
# T		453	0	2	0	0
# E		0	87	4	1	8
# J		1	4	347	0	0
# P		0	3	0	24	0
# K		0	7	0	0	9



Classification in TiMBL

Class statistics:

```
# Scores per Value Class:
# class | TP  FP  TN  FN  prec.  recall  ... F-score  ...
# T     | 453  1  494  2  0.99780  0.99560  ... 0.99670  ...
# E     |  87  14  836  13  0.86139  0.87000  ... 0.86567  ...
# J     | 347  6  592  5  0.98300  0.98580  ... 0.98440  ...
# P     |  24  1  922  3  0.96000  0.88889  ... 0.92308  ...
# K     |  9   8  926  7  0.52941  0.56250  ... 0.54545  ...
```



Classification in TiMBL

		true class	
		C	not C
pred. class	C	TP true positive	FP false positive
	not C	FN false negative	TN true negative



Classification in TiMBL

- Precision: how many of instances labeled as class C were indeed class C ?

$$precision = \frac{TP}{TP + FP}$$

- Recall: how many of instances that were class C were indeed labeled as class C ?

$$recall = \frac{TP}{TP + FN}$$



Classification in TiMBL

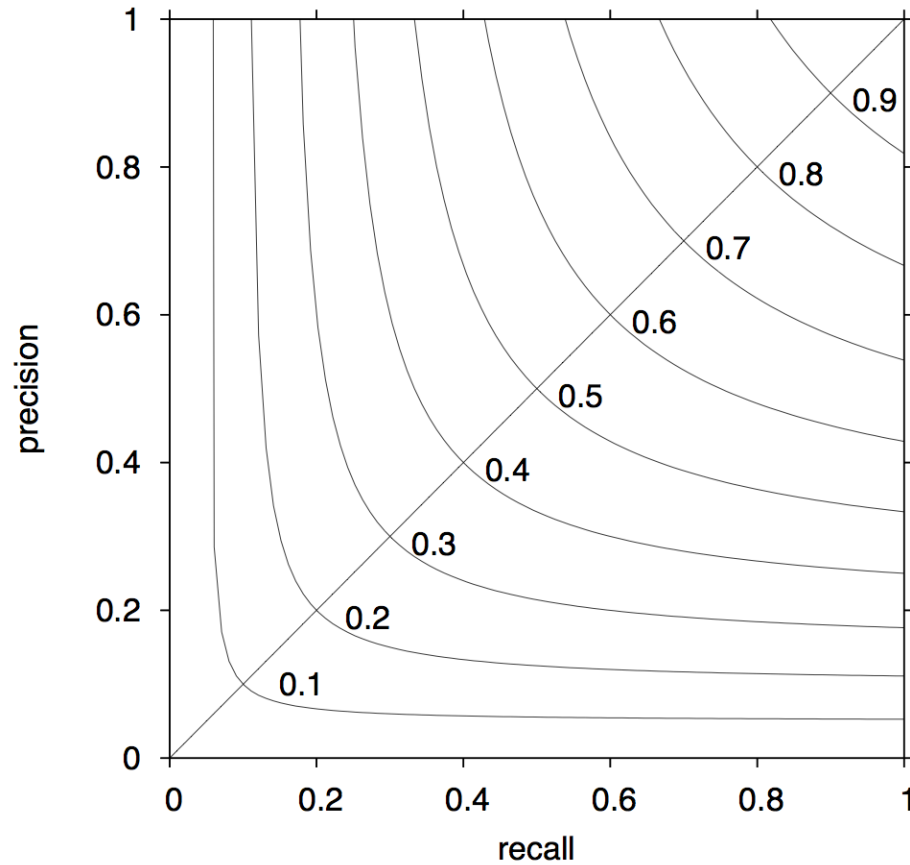
- F-score metric to summarize precision and recall in one measure
- Harmonic mean of precision and recall:

$$\text{F-score} = \frac{2 \times \textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}}$$

- Penalizes large differences between precision and recall



Classification in TiMBL





Classification in TiMBL

Class statistics:

```
# Scores per Value Class:
# class | TP  FP  TN  FN  prec.  recall  ... F-score  ...
# T     | 453  1 494  2 0.99780 0.99560  ... 0.99670  ...
# E     |  87 14 836 13 0.86139 0.87000  ... 0.86567  ...
# J     | 347  6 592  5 0.98300 0.98580  ... 0.98440  ...
# P     |  24  1 922  3 0.96000 0.88889  ... 0.92308  ...
# K     |   9   8 926  7 0.52941 0.56250  ... 0.54545  ...
```



Classification in TiMBL

- Class statistics provide F-scores per class
- Advanced statistics provide F-scores for the full test set
- Two types of averaging:
 - micro-averaging: the F-score for each class is weighted proportionally to the frequency of the class in the test set
 - macro-averaging: all the F-scores are added and the sum is divided by the number of classes



Classification in TiMBL

Advanced statistics:

```
# Scores per Value Class:
# class | TP  FP  TN  FN  prec.  recall  ... F-score  ...
# T     | 453  1 494  2 0.99780 0.99560 ... 0.99670 ...
# E     |  87 14 836 13 0.86139 0.87000 ... 0.86567 ...
# J     | 347  6 592  5 0.98300 0.98580 ... 0.98440 ...
# P     |  24  1 922  3 0.96000 0.88889 ... 0.92308 ...
# K     |  9   8 926  7 0.52941 0.56250 ... 0.54545 ...
# ...
# F-Score beta=1, microav: 0.968123
# F-Score beta=1, macroav: 0.863060
# ...
# overall accuracy: 0.968421 (920/950)
```




TiMBL: Dutch interfixes

- TiMBL has been applied to a wide range of Natural Language Processing and machine-learning tasks
- Krott, Baayen & Schreuder (2001) is an example of an application in psycholinguistic research



TiMBL: Dutch interfixes

- Topic: linking morphemes in Dutch compounds
- Dutch has three linking morphemes:
 - -en (e.g.; “boek**en**plank” (bookshelf))
 - -s (e.g.; “plaatj**s**boek” (pictureboek))
 - no linking element (e.g.; “thee**∅**pot” (teapot))



TiMBL: Dutch interfixes

- Traditional approach: capture distribution of linking morphemes through phonological, morphological and semantic rules
- Example: “no linking morpheme if the first constituent ends with a vowel”
 - “thee∅pot” (teapot)
 - “knie∅schijf” (knee cap)
 - **but:** “pygmee-en-volk” (pygmy people)



TiMBL: Dutch interfixes

- Can memory-based learning capture the distribution of linking morphemes in Dutch compounds?
- Two test-cases:
 - Train TiMBL on existing Dutch compounds
 - Predict the choice of linking morphemes in neologisms



TiMBL: Dutch interfixes

- Two test-cases:
 - Predict the linking morpheme in existing Dutch compounds
 - Predict the linking morpheme in neologisms



TiMBL: Dutch interfixes

- Train TiMBL on 22,994 existing Dutch compounds
- Classes: linking morphemes **-en**, **-s** and \emptyset
- Features:
 - left constituent and right constituent
 - plural suffix left constituent
 - animacy left and right constituent
 - abstractness left and right constituent
 - morphological complexity left constituent



TiMBL: Dutch interfixes

- Training data are test data
- 10-fold cross validation:
 - divide data into 10 random held-out subsets
 - for each held-out subset, predict linking morphemes based on training set of other 90%
 - model performance is the average percentage of correct classification for all 10 subsets



TiMBL: Dutch interfixes

- TiMBL settings:
 - standard k -NN algorithm
 - overlap similarity
 - Information Gain (IG) feature weighting
 - $k = 1$
 - no distance-weighting



TiMBL: Dutch interfixes

Feature relevance:

# feature	IG
# -----	-----
# left constituent	1.11
# right constituent	0.41
# plural suffix left constituent	0.10
# abstractness left constituent	0.07
# animacy left constituent	0.04
# abstractness right constituent	0.02
# animacy right constituent	0.00
# stress final syllable left const.	0.07
# morphological complexity left const.	0.11



TiMBL: Dutch interfixes

- Classification performance: 93.2%
- Using only the first constituent as a feature: 92.5%



TiMBL: Dutch interfixes

- Two test-cases:
 - Predict the linking morpheme in existing Dutch compounds
 - Predict the linking morpheme in neologisms



TiMBL: Dutch interfixes

- Does the model predict the right linking morpheme for new compounds?
- Two steps:
 - 1) Collect human classification behavior for neologisms
 - 2) Compare model predictions against human classifications



TiMBL: Dutch interfixes

- Present participants with non-existing compounds
- Leave a blank between the two constituents
- Ask participants to fill out the most appropriate linking morpheme (if any) at the blank



TiMBL: Dutch interfixes

mier__val



TiMBL: Dutch interfixes

bedrijf__bos



TiMBL: Dutch interfixes

zand__bord



TiMBL: Dutch interfixes

- Systematically vary the bias of the left and right constituent towards **-en** and **-s**
- Experimental results confirm data for existing compounds:
 - strong effect of left constituent bias
 - weaker effect of right constituent bias



TiMBL: Dutch interfixes

- Can TiMBL capture these experimental findings?
- Train TiMBL on existing Dutch compounds and classify the experimental items in the test set
- Compare TiMBL classification to the majority choice of the participants in the experiments



TiMBL: Dutch interfixes

- Classification performance abstract rules: 53.94%
- TiMBL classification performance: 87.35%
- Relevant features:
 - left constituent
 - abstractness of right constituent (abstract right: fewer **-en**, more **-s**)
 - animacy of left constituent (animate left: more **-en**)



TiMBL: Dutch interfixes

- Conclusions:
 - The choice for a linking morpheme in Dutch compounds is primarily guided by the identity of the first constituent
 - Memory Based Learning accurately captures the distribution of linking morphemes in both existing and new Dutch compounds



Memory Based Learning: conclusions

- Memory-based learning often offers excellent classification performance
- Memory-based learning is applicable to a wide range of Natural Language Processing and machine-learning tasks



Memory Based Learning: conclusions

- Downsides:
 - storage requirements and computational costs are proportional to the size of the training data
 - decision tree optimizations alleviate these concerns, but are conceptually similar to a set of rules
 - neuro-biologically implausible



Thank you!

