Burrows Delta: Feature selection and standardization

Stefan Evert

6 March 2016

Contents

1 Evaluating three variants of Burrows Delta

1.1 Data sets

Load relative frequencies and z-scores for the German, English and French data set. For technical reasons, the data structures store the transposed document-term matrices $\mathbf{F}^T$ and $\mathbf{Z}^T$

```
load("data/delta_corpus.rda")
## FreqDE, FreqEN, FreqFR ... text-word matrix with absolute and relative frequencies
## zDE, zEN, zFR          ... standardized (z-transformed) relative frequencies
## goldDE, goldEN, goldFR ... gold standard labels (= author names)
```

- $\mathbf{F}^T$ is available under the names `FreqDE$S`, `FreqEN$S` and `FreqFR$S`
- $\mathbf{Z}^T$ is available under the names `zDE`, `zEN` and `zFR`
- absolute frequencies $n_{D_j} \cdot f_i(D_j)$ can be found in `FreqDE$M`, `FreqEN$M`, `FreqFR$M`

1.2 A partial replication of Jannidis et al. (2015)

Jannidis et al. (2015) compute clusterings for different versions of the Delta measure based on the most frequent $n_w = 100, 1000, 5000$ words as features. Our results for Burrows Delta Δ_B are:

```
n.vals <- c(100, 1000, 5000)
res <- rbind(
  evaluate(zDE, goldDE, n=n.vals, method="manhattan", label="DE | Burrows D"),
  evaluate(zEN, goldEN, n=n.vals, method="manhattan", label="EN | Burrows D"),
```

```
  evaluate(zFR, goldFR, n=n.vals, method="manhattan", label="FR | Burrows D"))
knitr::kable(res)
```

	n	p	norm.p	accuracy	adj.rand
DE | Burrows D | n=100	100	2	2	93.33	76.41
DE | Burrows D | n=1000	1000	2	2	98.67	87.85
DE | Burrows D | n=5000	5000	2	2	96.00	79.14
EN | Burrows D | n=100	100	2	2	77.33	53.08
EN | Burrows D | n=1000	1000	2	2	94.67	82.20
EN | Burrows D | n=5000	5000	2	2	94.67	81.22
FR | Burrows D | n=100	100	2	2	86.67	71.10
FR | Burrows D | n=1000	1000	2	2	93.33	81.22
FR | Burrows D | n=5000	5000	2	2	94.67	84.37

Quadratic Delta $\sqrt{\Delta_Q}$ achieves a considerably lower accuracy and Rand index than Δ_B, which is in line with the findings of (Jannidis et al. 2015).

```
res <- rbind(
  evaluate(zDE, goldDE, n=n.vals, method="euclidean", label="DE | Quadratic D"),
  evaluate(zEN, goldEN, n=n.vals, method="euclidean", label="EN | Quadratic D"),
  evaluate(zFR, goldFR, n=n.vals, method="euclidean", label="FR | Quadratic D"))
knitr::kable(res)
```

	n	p	norm.p	accuracy	adj.rand
DE | Quadratic D | n=100	100	2	2	93.33	76.41
DE | Quadratic D | n=1000	1000	2	2	94.67	78.65
DE | Quadratic D | n=5000	5000	2	2	85.33	64.07
EN | Quadratic D | n=100	100	2	2	78.67	46.24
EN | Quadratic D | n=1000	1000	2	2	89.33	66.79
EN | Quadratic D | n=5000	5000	2	2	89.33	47.44
FR | Quadratic D | n=100	100	2	2	89.33	78.64
FR | Quadratic D | n=1000	1000	2	2	93.33	80.46
FR | Quadratic D | n=5000	5000	2	2	85.33	61.87

Jannidis et al. (2015) report best clustering results for Cosine Delta $\Delta_\angle$, which is based on cosine similarity (or, equivalently, angular distance) between features vectors rather than their Euclidean distance. This stands in stark contrast to Argamon's probabilistic argumentation, but is confirmed by our replication.

```
res <- rbind(
  evaluate(zDE, goldDE, n=n.vals, method="cosine", label="DE | Cosine D"),
  evaluate(zEN, goldEN, n=n.vals, method="cosine", label="EN | Cosine D"),
  evaluate(zFR, goldFR, n=n.vals, method="cosine", label="FR | Cosine D"))
knitr::kable(res)
```

	n	p	norm.p	accuracy	adj.rand
DE | Cosine D | n=100	100	2	2	89.33	75.35
DE | Cosine D | n=1000	1000	2	2	98.67	96.60
DE | Cosine D | n=5000	5000	2	2	96.00	96.60
EN | Cosine D | n=100	100	2	2	84.00	64.41
EN | Cosine D | n=1000	1000	2	2	98.67	96.60
EN | Cosine D | n=5000	5000	2	2	94.67	93.24

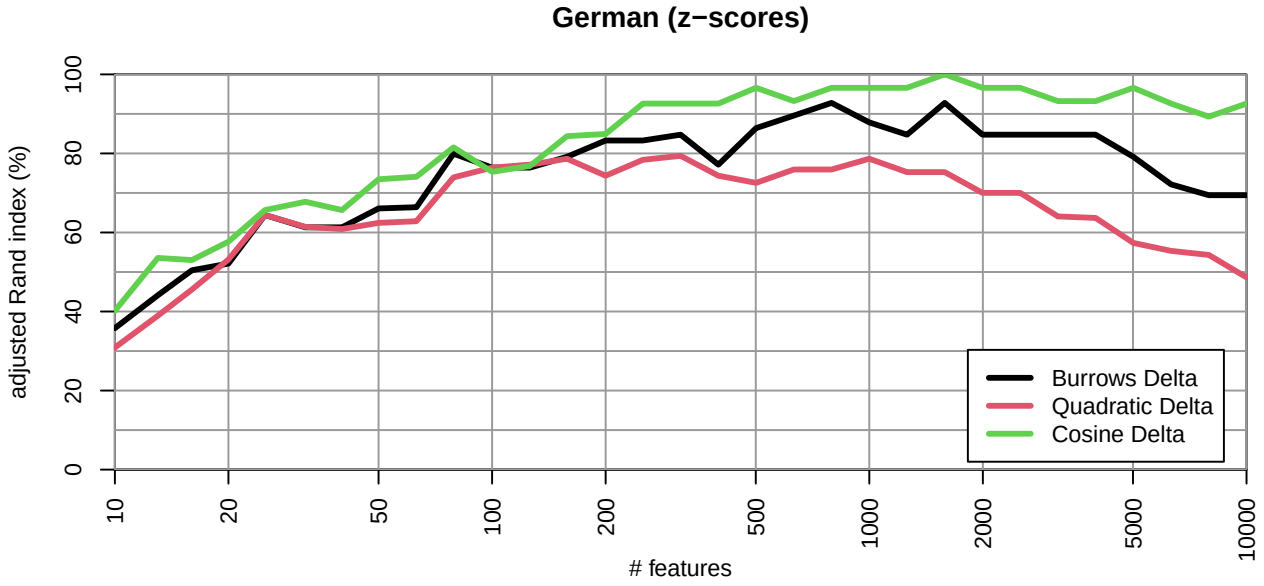
	n	p	norm.p	accuracy	adj.rand
FR | Cosine D | n=100	100	2	2	93.33	82.52
FR | Cosine D | n=1000	1000	2	2	97.33	93.24
FR | Cosine D | n=5000	5000	2	2	93.33	93.24

1.3 Dimensionality and feature selection

We can also visualize the relationship between the number of words used as features and the classification accuracy / ARI achieved for different methods in each language. This gives us a much better overview of the behaviour of different Delta measures and other parameters than showing results for selected dimensionalities in a table.

```
n.vals <- round(10 ^ seq(1, 4, .1)) # logarithmic steps
draw.grid <- function () { # corresponding grid for plot region
  abline(h=seq(0, 100, 10), col="grey60")
  abline(v=c(10,20,50,100,200,500,1000,2000,5000,10000), col="grey60")
}
```

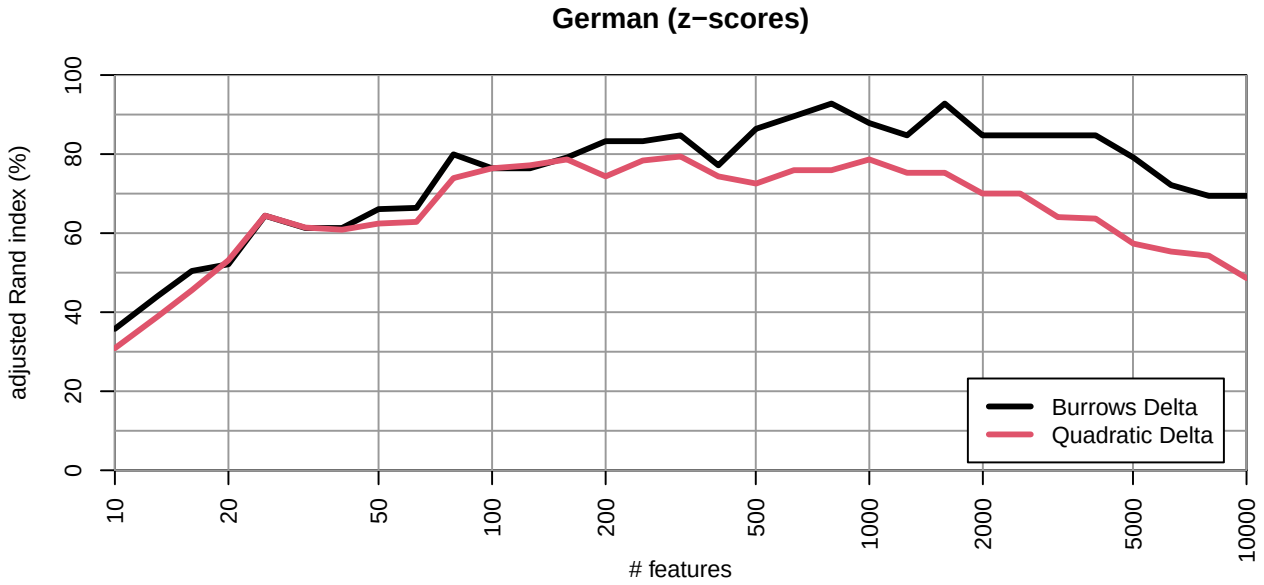
```
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)", main="German (z-scores)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



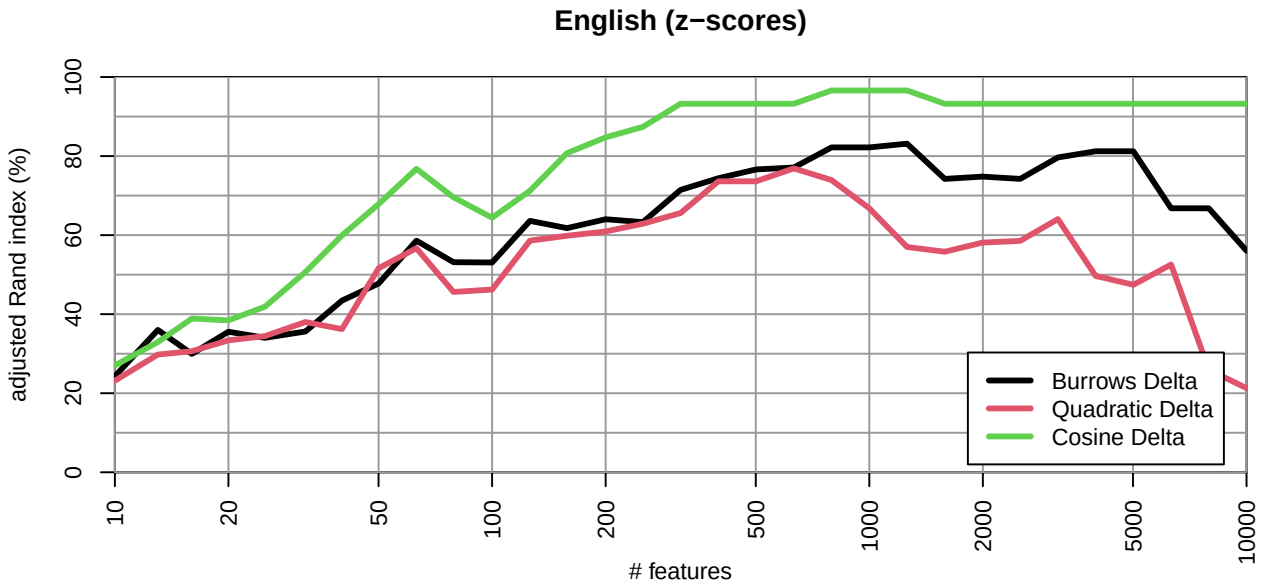
For small $n_w \leq 200$ (mostly function words), the three variants of Delta show very similar performance. As the number of feature dimensions increases, Δ_B is much more robust than Δ_Q, and $\Delta_\angle$ is even better.

A possible explanation would be that the z-transformation scales up small random differences in the frequency counts of less frequent words (with small σ_i), which can result in outlier values. Euclidean distance is known to be more sensitive to such outlier values than Manhattan distance because of the squared feature differences.

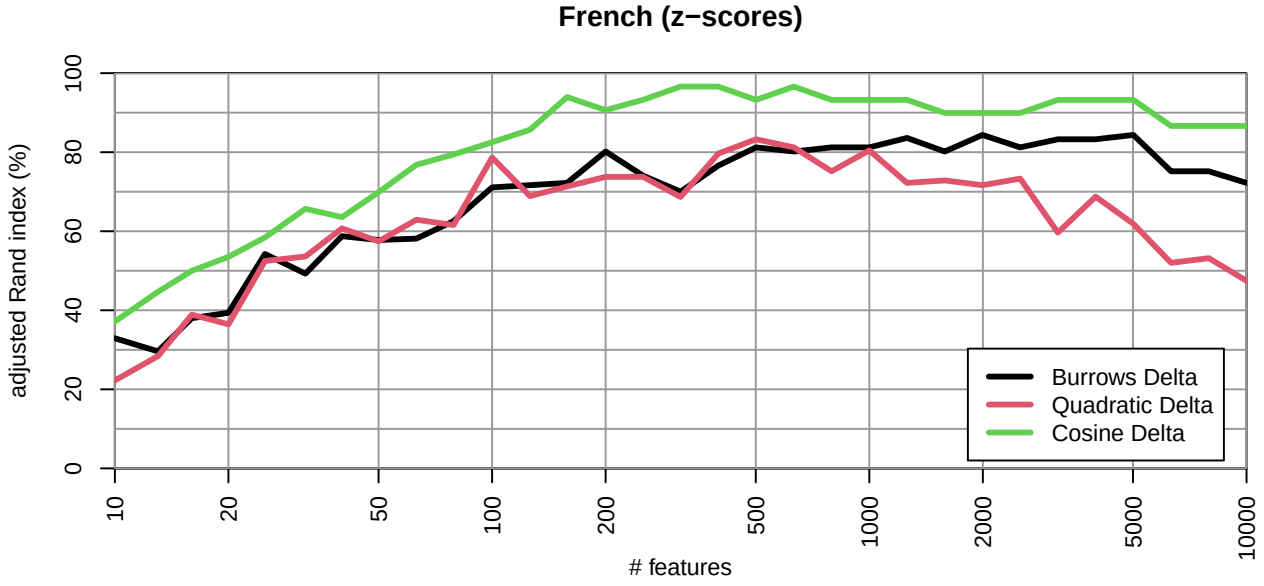
For conference presentations, we also produce incremental plots:




```
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)", main="English (z-scores)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(zEN, goldEN, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(zEN, goldEN, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(zEN, goldEN, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```




```
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)", main="French (z-scores)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(zFR, goldFR, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(zFR, goldFR, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(zFR, goldFR, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```

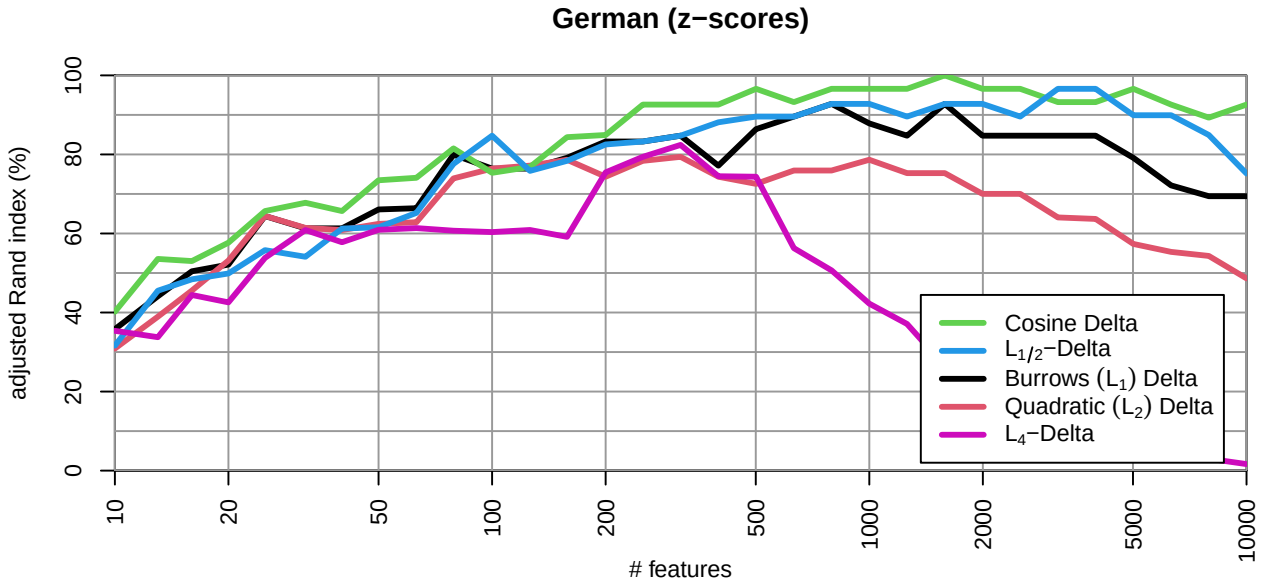


On the English and French data sets, we observe similar patterns. However, Δ_B and Δ_Q diverge only for $n_w > 500$, and $\Delta_\angle$ is always considerably better than Δ_B. It is not clear yet whether there is any connection to typological differences between the languages or whether it simply has to do with the particular authors and texts included in the samples.

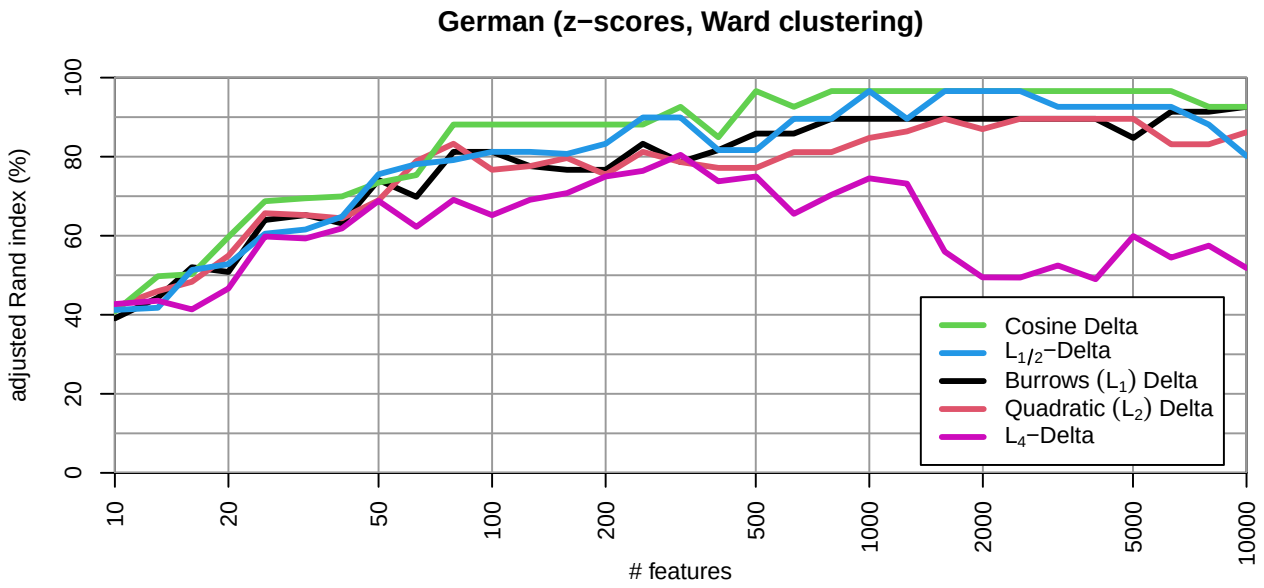
One possible explanation for the poor performance and robustness of Δ_Q is the well-known sensitivity of Euclidean distance to individual outlier values among the features (because large differences are squared and

thus get disproportionately more weight). In order to find support for this hyptohesis, we can compare Δ_B (L_1) and Δ_Q (L_2) to Delta measures based on other p-norms L_p:

```
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)", main="German (z-scores)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, method="minkowski", p=0.5)$adj.rand, lwd=3, col=4)
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, method="minkowski", p=4)$adj.rand, lwd=3, col=6)
legend("bottomright", inset=.02, bg="white", lwd=3, col=c(3,4,1,2,6),
       legend=expression("Cosine Delta", L[1/2]*"-Delta", "Burrows "*(L[1])*" Delta",
                         "Quadratic "*(L[2])*" Delta", L[4]*"-Delta"))
```



This clearly shows that clustering quality gradually decreases for higher L_p norms. However, the clustering method used also seems to play an important role. Repeating the same experiment with Ward's hierarchical clustering method (using the standard `hclust()` implementation), differences between Δ_B and Δ_Q become much smaller and both are robust up to $n_w \approx 10,000$ mfw. This issue will be explored in more detail in "Understanding Delta".

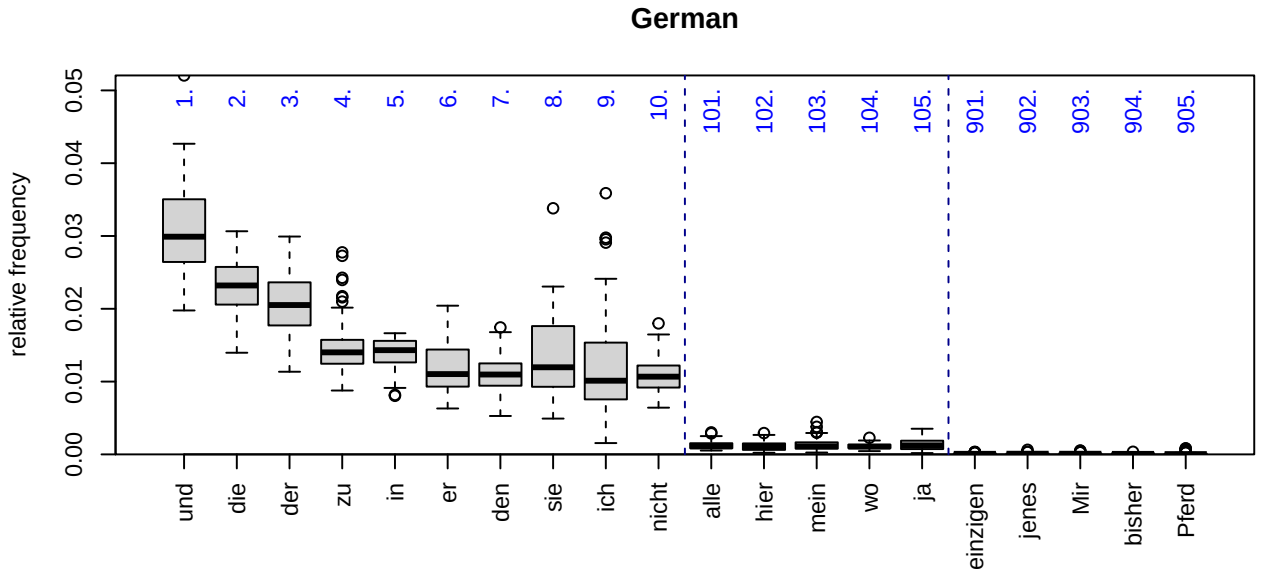


2 The effects of standardization

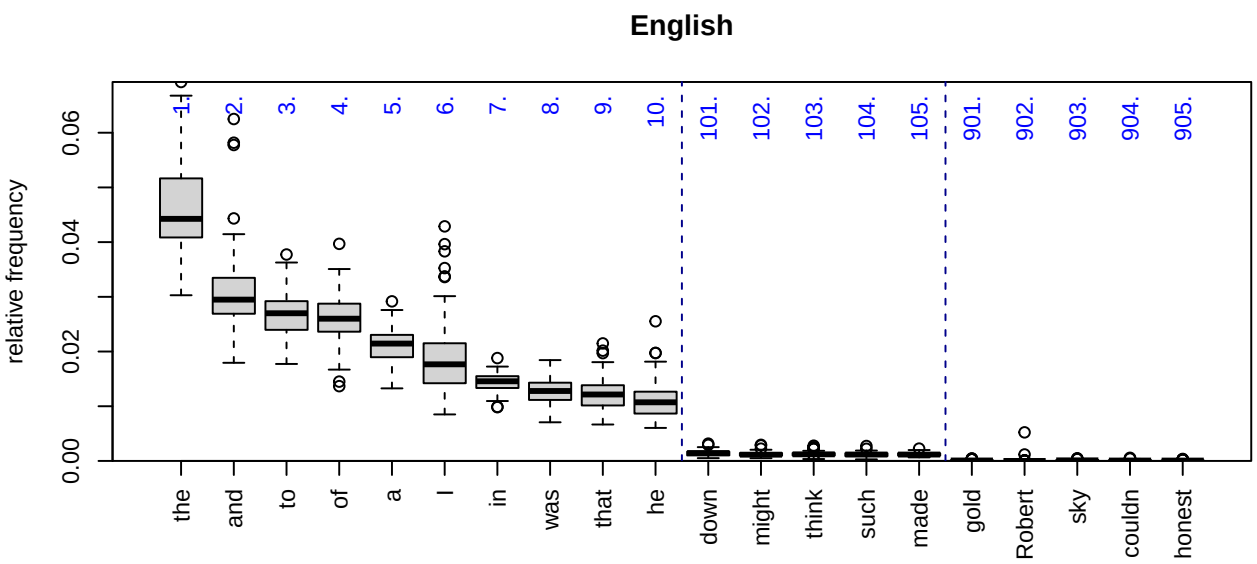
2.1 What is the purpose of standardization?

The average relative frequencies of feature words are wildly different, following a Zipfian distribution illustrated by the boxplots below for the German and English data sets.

```
r.vals <- c(1:10,101:105,901:905) # ranks of selected features
boxplot(FreqDE$S[, r.vals], las=3, yaxs="i", ylab="relative frequency", main="German")
abline(v=c(10.5,15.5), col="darkblue", lty="dashed")
text(seq_along(r.vals), par("usr")[4], sprintf("%d.  ", r.vals),
     adj=c(1, 0.5), srt=90, col="blue")
```

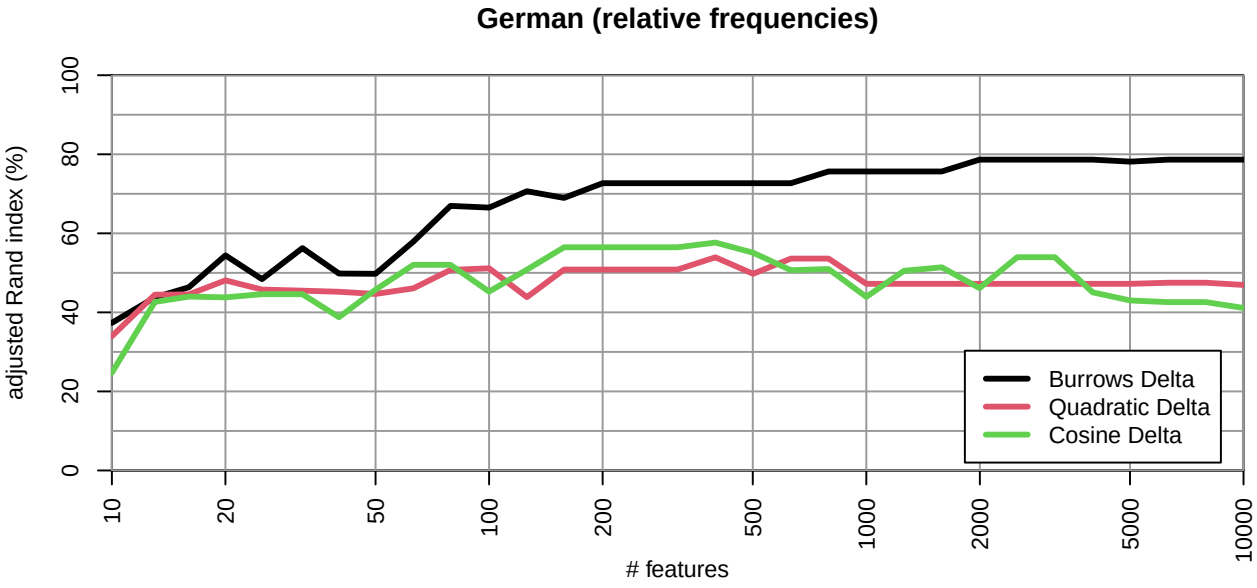



```
boxplot(FreqEN$S[, r.vals], las=3, yaxs="i", ylab="relative frequency", main="English")
abline(v=c(10.5,15.5), col="darkblue", lty="dashed")
text(seq_along(r.vals), par("usr")[4], sprintf("%d.  ", r.vals),
     adj=c(1, 0.5), srt=90, col="blue")
```



If we computed text distances based on unscaled relative frequencies, they would be determined almost exclusively by the first few most frequent function words. The resulting classification or clustering accuracy is very low, especially for Δ_Q and $\Delta_\angle$, and stabilizes quickly after the first few dozens of features.

```
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)", main="German (relative frequencies)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(FreqDE$S, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(FreqDE$S, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(FreqDE$S, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



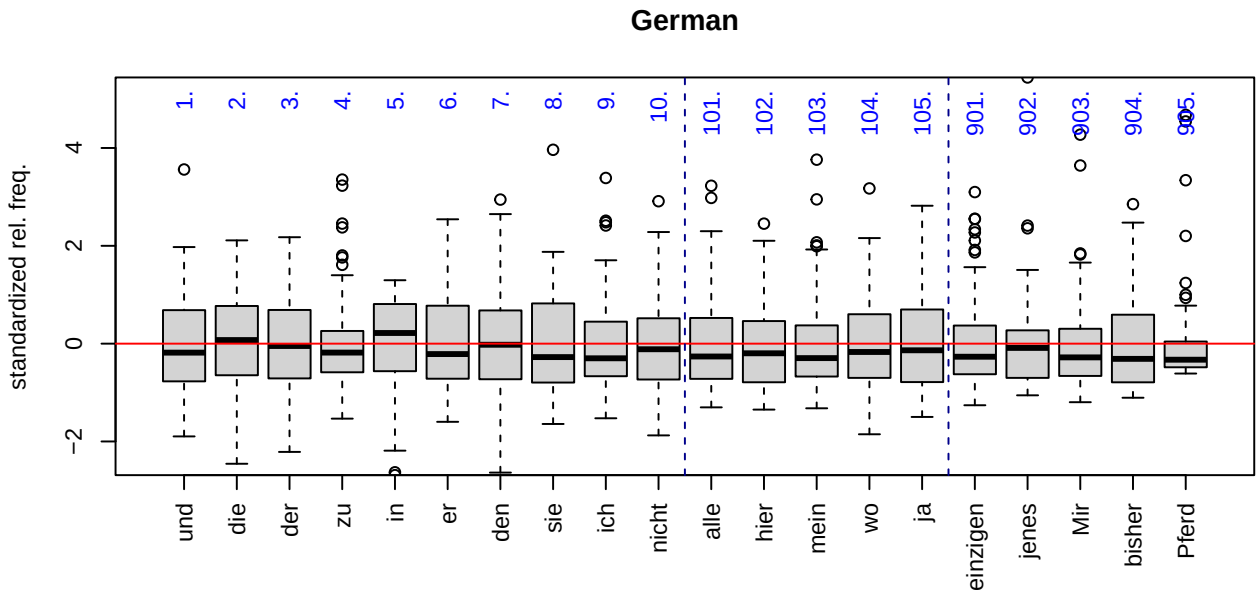
Standardization of the features, i.e. a z-transformation of each column of the term-document matrix, ensures that every feature makes the same overall contribution to the distances between texts (at least in combination

with Δ_Q). Keep in mind that each feature is standardized separately, i.e.

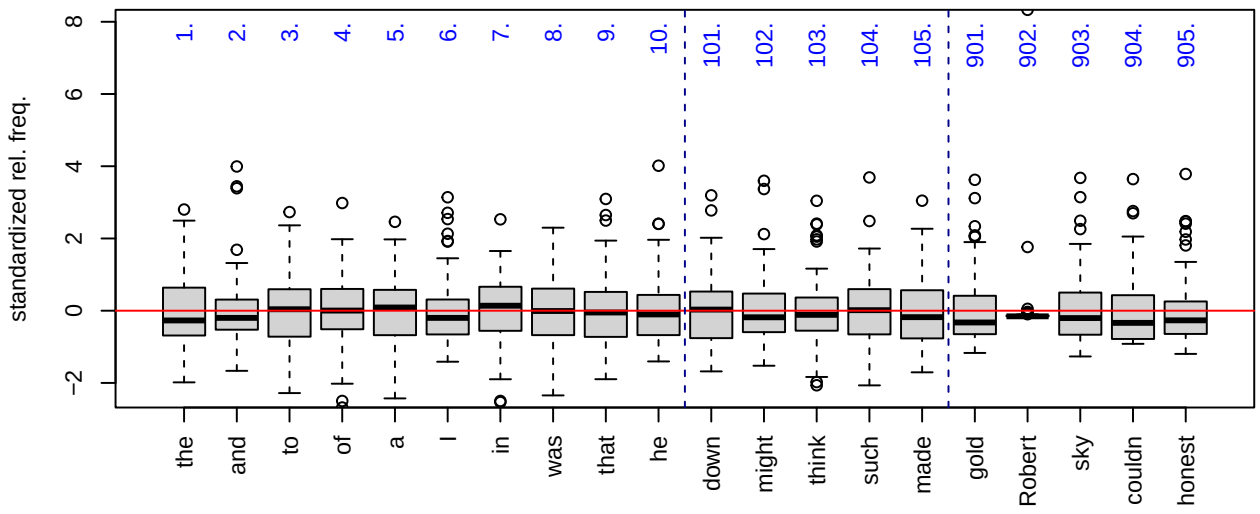
$$z_i(D) = \frac{p_i(D) - \mu_i}{\sigma_i},$$

where μ_i is the mean and σ_i the standard deviation of the relative frequencies of w_i across all texts $D \in \mathcal{D}$.

```
boxplot(zDE[, r.vals], las=3, yaxs="i", ylab="standardized rel. freq.", main="German")
abline(h=0, col="red")
abline(v=c(10.5,15.5), col="darkblue", lty="dashed")
text(seq_along(r.vals), par("usr")[4], sprintf("%d.  ", r.vals),
     adj=c(1, 0.5), srt=90, col="blue")
```




```
boxplot(zEN[, r.vals], las=3, yaxs="i", ylab="standardized rel. freq.", main="English")
abline(h=0, col="red")
abline(v=c(10.5,15.5), col="darkblue", lty="dashed")
text(seq_along(r.vals), par("usr")[4], sprintf("%d.  ", r.vals),
     adj=c(1, 0.5), srt=90, col="blue")
```

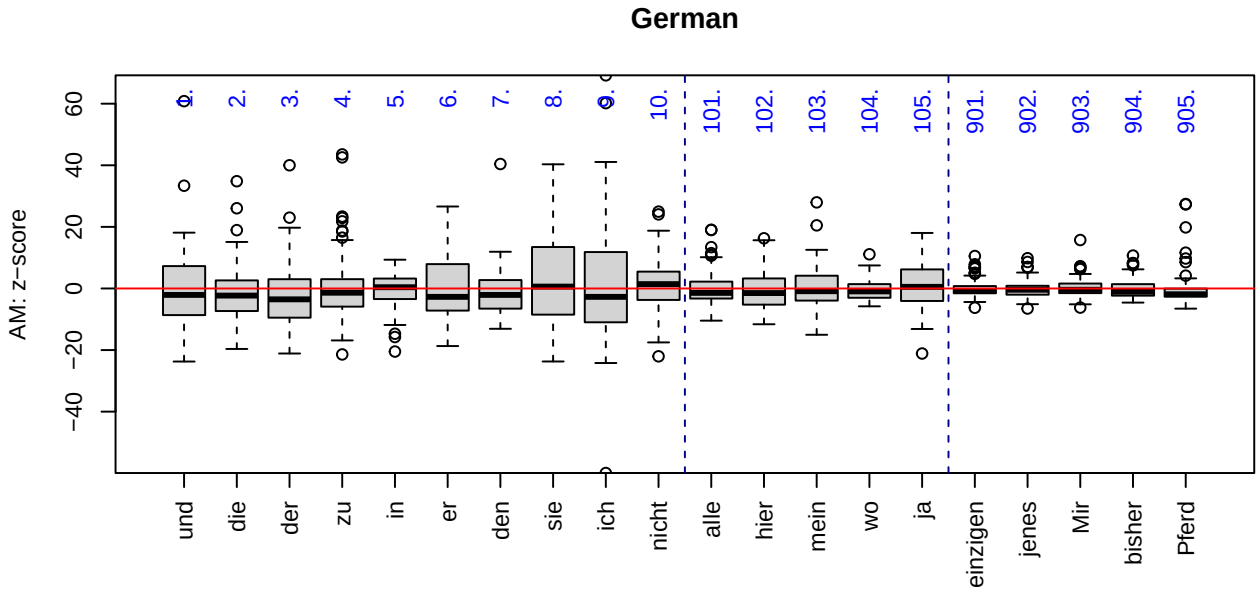


In distributional semantics, it is uncommon to apply a z-transformation in order to reduce the dominance of high-frequency words among the features, not least because this would turn the sparse term-document (or term-term) matrix into a dense matrix.[1]

Sometimes, absolute frequencies are simply log-transformed, or relevance weights from Information Retrieval are applied (such as tf.idf). Various statistical association measures compare the frequency of a word in each text against its average frequency in the entire collection (i.e. its expected frequency in the text). Normalization of the divergence between observed and expected frequency is usually based on the *expected* standard deviation of frequencies across random samples rather than their observed s.d. in the text collection. This gives less weight to features that are relatively stable across the texts, while a z-transformation would scale up the remaining small differences.

As a result, the contributions of different features are not completely equalized if one of these weighting schemes is applied. We illustrate this below with the z-score association measure, which is also based on the idea of a z-transformation (but using expected values for μ_i and σ_i rather than sample estimates).

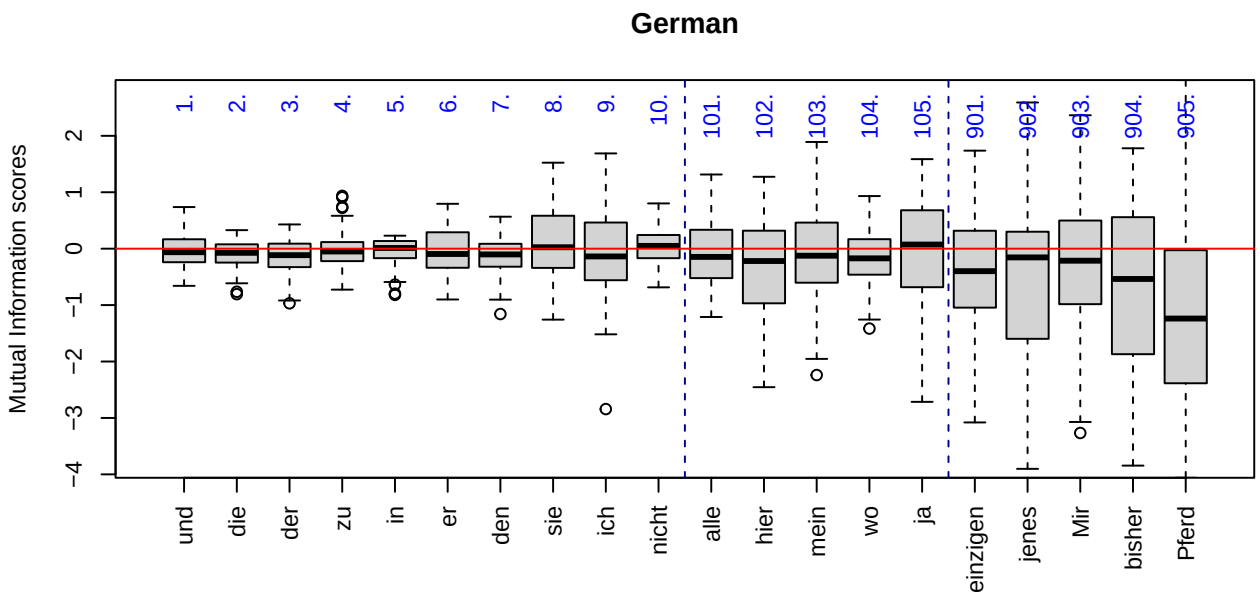
```
tmp <- dsm.score(FreqDE, score="z-score", sparse=FALSE)
boxplot(tmp$S[, r.vals], las=3, yaxs="i", ylab="AM: z-score", main="German")
abline(h=0, col="red")
abline(v=c(10.5,15.5), col="darkblue", lty="dashed")
text(seq_along(r.vals), par("usr")[4], sprintf("%d.  ", r.vals),
     adj=c(1, 0.5), srt=90, col="blue")
```



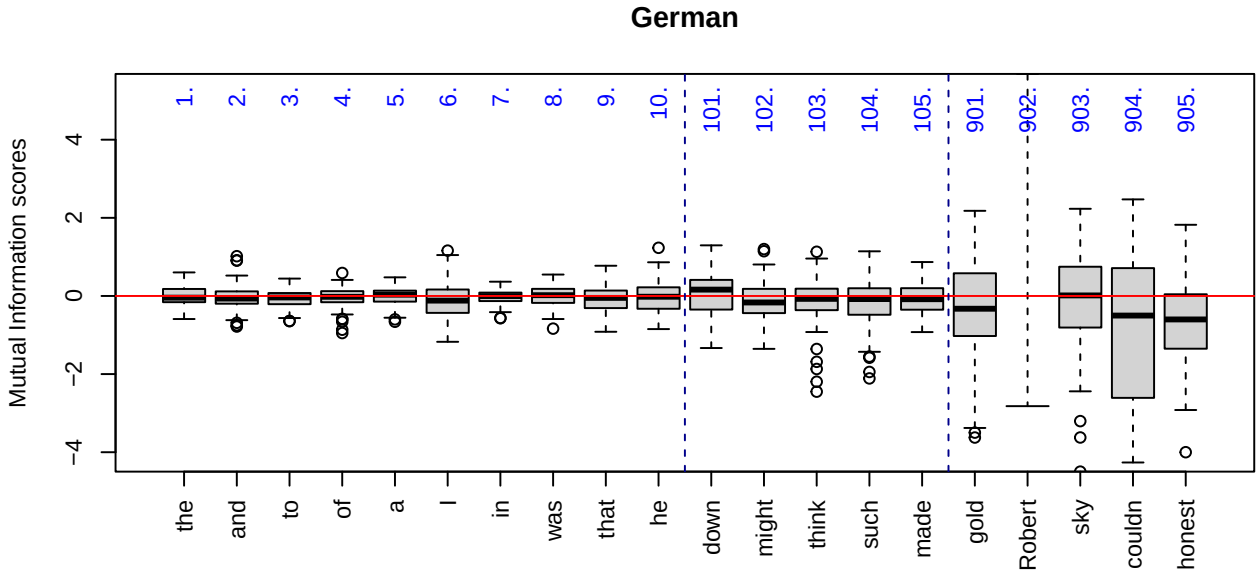
Note that other popular association measures, in particular Mutual Information shown below, can have an opposite effect and will result in erratic behaviour as small frequency differences are scaled up.

```
tmp <- dsm.score(FreqDE, score="MI", sparse=FALSE)
boxplot(tmp$S[, r.vals], las=3, yaxs="i", ylab="Mutual Information scores", main="German")
abline(h=0, col="red")
abline(v=c(10.5,15.5), col="darkblue", lty="dashed")
text(seq_along(r.vals), par("usr")[4], sprintf("%d.  ", r.vals),
     adj=c(1, 0.5), srt=90, col="blue")
```

[1] $p_i(D) = 0$ always corresponds to a negative z-score $z_i(D) < 0$



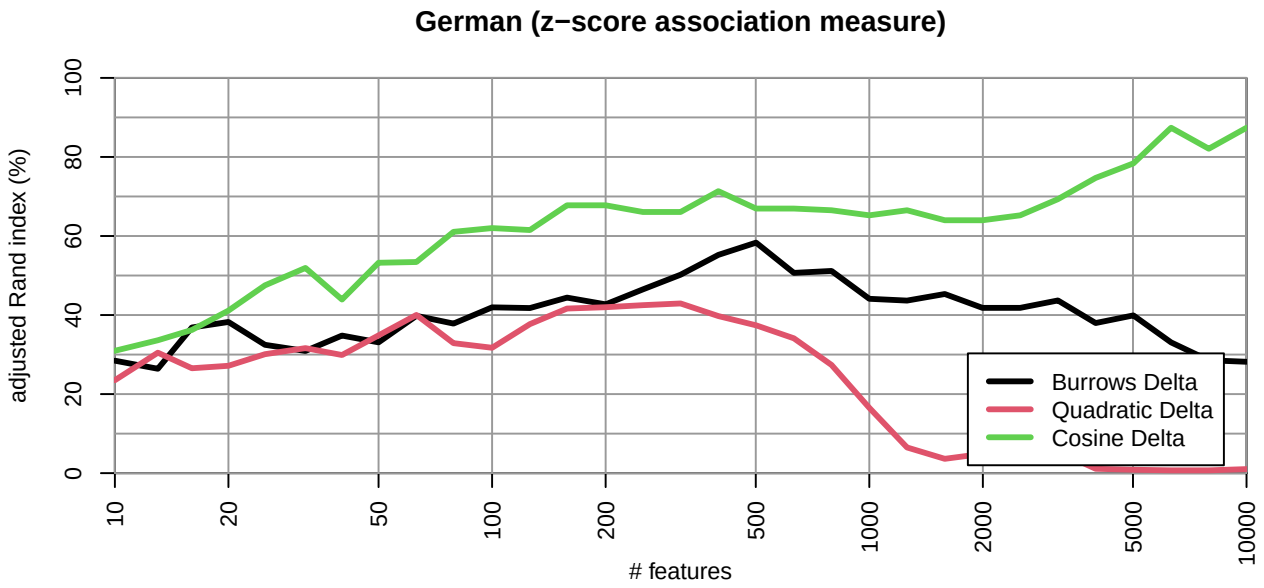

```
tmp <- dsm.score(FreqEN, score="MI", sparse=FALSE)
boxplot(tmp$S[, r.vals], las=3, yaxs="i", ylab="Mutual Information scores", main="German")
abline(h=0, col="red")
abline(v=c(10.5,15.5), col="darkblue", lty="dashed")
text(seq_along(r.vals), par("usr")[4], sprintf("%d.  ", r.vals),
     adj=c(1, 0.5), srt=90, col="blue")
```



Empirically, weighting features with association measures or similar schemes doesn't work well, and Burrows's z-transformation seems to be a much better choice.

```
tmp <- dsm.score(FreqDE, score="z-score", sparse=FALSE)
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)", main="German (z-score association measure)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp$S, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp$S, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
```

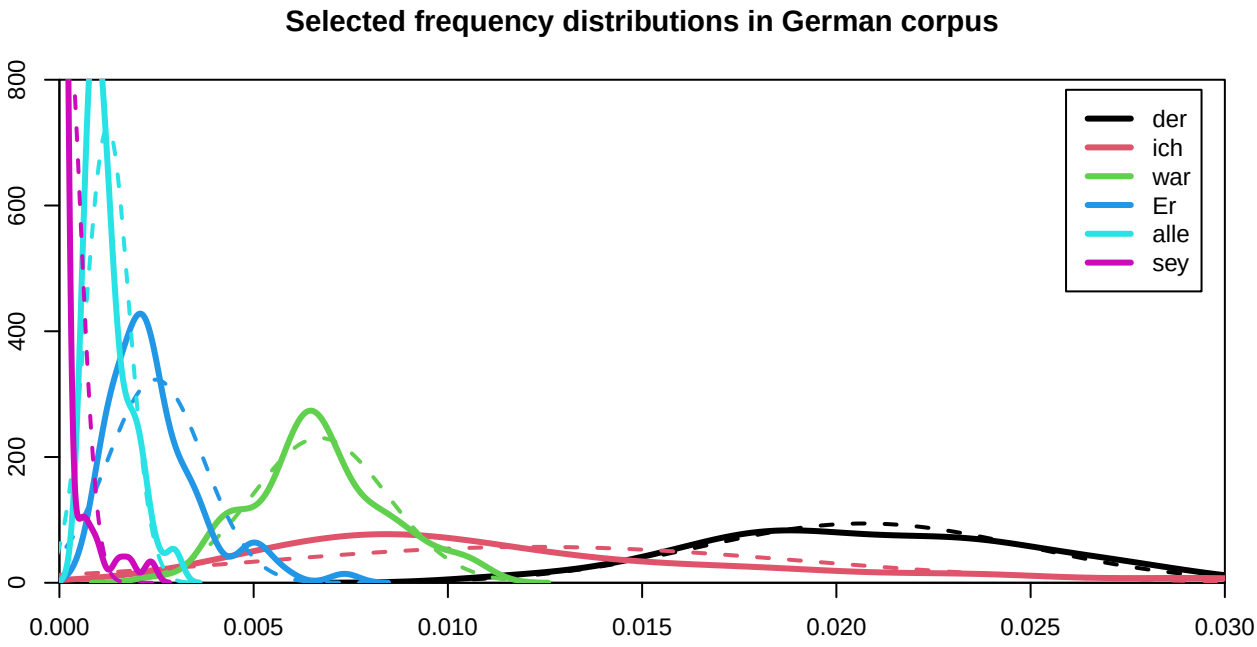
```
lines(n.vals, evaluate(tmp$S, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



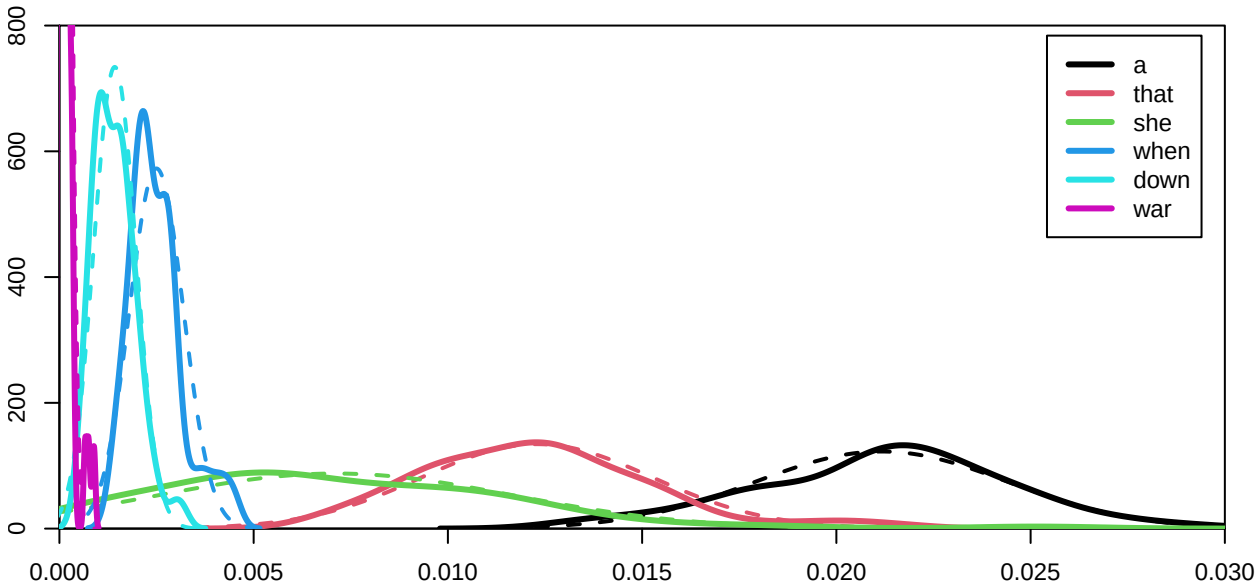
2.2 Is standardization sensible?

The standardization proposed by Burrows (2002) is considered appropriate if the distribution of relative frequencies $p_i(D)$ across texts is approximately Gaussian for each feature w_i. This is indeed the case for most high-frequency words, but less and less so as frequency decreases (the dashed lines show what a perfectly Gaussian distribution would look like).

```
r.vals <- c(3, 9, 20, 52, 101, 1000)
plot(0, 0, type="n", xlim=c(0, 0.03), ylim=c(0, 800), xaxs="i", yaxs="i",
     xlab="", ylab="", main="Selected frequency distributions in German corpus")
for (i in seq_along(r.vals)) {
  p <- FreqDE$S[, r.vals[i]]
  res <- density(p)
  lines(res, lwd=3, col=i)
  lines(res$x, dnorm(res$x, mean=mean(p), sd=sd(p)), lwd=2, col=i, lty="dashed")
}
legend("topright", inset=.02, bg="white", legend=FreqDE$cols$term[r.vals],
       lwd=3, col=seq_along(r.vals))
```



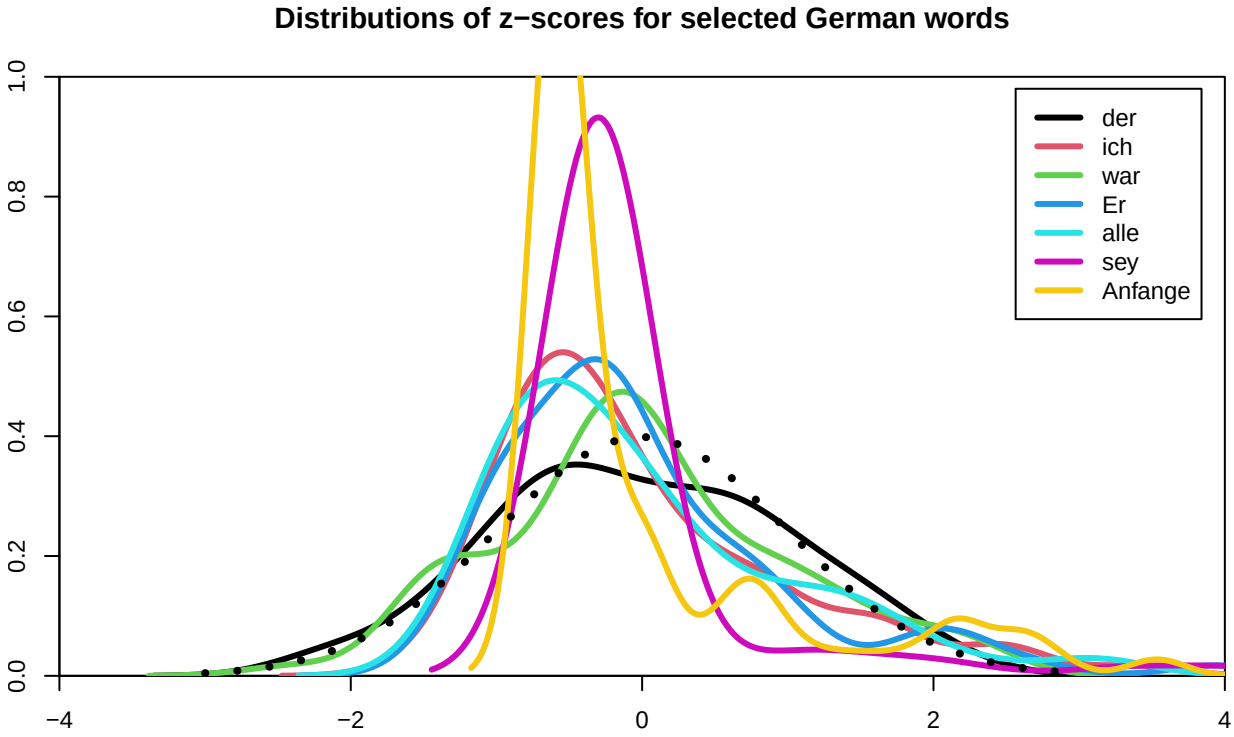

```
r.vals <- c(5, 9, 19, 50, 101, 999)
plot(0, 0, type="n", xlim=c(0, 0.03), ylim=c(0, 800), xaxs="i", yaxs="i",
     xlab="", ylab="", main="Selected frequency distributions in English corpus")
for (i in seq_along(r.vals)) {
  p <- FreqEN$S[, r.vals[i]]
  res <- density(p)
  lines(res, lwd=3, col=i)
  lines(res$x, dnorm(res$x, mean=mean(p), sd=sd(p)), lwd=2, col=i, lty="dashed")
}
legend("topright", inset=.02, bg="white", legend=FreqEN$cols$term[r.vals],
       lwd=3, col=seq_along(r.vals))
```



The differences between the distributions become particulary clear after standardization. The dotted line

shows the ideal shape of a standard normal distribution with $\mu = 0$ and $\sigma = 1$.

```
r.vals <- c(3, 9, 20, 52, 101, 1000, 7500)
plot(0, 0, type="n", xlim=c(-4, 4), ylim=c(0, 1), xaxs="i", yaxs="i",
     xlab="", ylab="", main="Distributions of z-scores for selected German words")
for (i in seq_along(r.vals)) {
  z <- zDE[, r.vals[i]]
  lines(density(z), lwd=3, col=i)
}
x <- seq(-3, 3, .1)
lines(x, dnorm(x), lwd=4, col="black", lty="dotted")
legend("topright", inset=.02, bg="white", legend=FreqDE$cols$term[r.vals],
       lwd=3, col=seq_along(r.vals))
```



For words that occur just in a few texts (i.e. where all other $p_i(D) = 0$ the distribution is not even remotely Gaussian. In this case, standardization shifts and expands the skewed and long-tailed frequency distribution in a meaningless way. It also destroys the inherent sparseness of such distributions: each $p_i(D) = 0$ is replaced by a small negative value $z_i(D) = -\mu_i/\sigma_i$.

As an illustration, we show the frequency profiles of selected feature words from different frequency ranges across the first 12 texts in the German corpus:

```
res <- t(FreqDE$M[1:12, c(1,10,100,500,1000,2000,5000,7500,10001,50001,240010)])
colnames(res) <- sprintf("T%02d", 1:12) # get rid of text names for readable display
knitr::kable(res)
```

	T01	T02	T03	T04	T05	T06	T07	T08	T09	T10	T11	T12
und	1131	1372	2973	2460	950	3752	4045	3440	2110	2337	8465	2765
nicht	780	459	946	719	369	1316	1760	1343	572	1157	2734	1184
sehr	95	45	199	55	104	78	470	201	90	129	283	213
Stunden	18	8	17	11	8	29	14	24	18	27	37	33
sey	0	0	0	1	22	0	0	0	0	0	0	0

	T01	T02	T03	T04	T05	T06	T07	T08	T09	T10	T11	T12
schließlich	7	0	7	0	0	0	4	27	0	0	1	0
allenfalls	0	0	0	0	1	0	1	1	2	0	8	1
Anfange	0	3	0	0	2	0	0	0	0	0	1	5
Dämonen	0	0	0	0	0	0	0	0	0	0	0	1
Kaffeewirt	0	0	0	0	0	0	2	0	0	0	1	0
Petersilius	0	0	0	0	0	0	1	0	0	0	0	0

1. Words up to rank 500 occur in virtually every text (*und, nicht, sehr, Stunden*). They comprise function words as well as very common nouns, verbs and adjectives.
2. Words with ranks between 500 and 5000 are less common words (*sey, schließlich, allenfalls*) that only occur in some of the texts (often depending on the topics addressed). The corresponding distributions become increasingly sparse.
3. Words above rank 10000 are highly specialized (*Dämonen, Kaffeewirt, Petersilius*) and have an extremely sparse distribution, occurring in a few texts or just a single text. They include the names of characters from one of the novels. For these words, standardization doesn't make any mathematical sense.

Insight: the range of feature words that can sensibly be standardized corresponds quite well to the number of feature dimensions (up to $n_w \approx 5000$) that yields good performance in the authorship attribution task.

2.3 Feature selection and dimensionality

The considerations above suggest that instead of using the n_w most frequent words as features, it might be a good idea to select features by their nonzero counts, i.e. the number of different texts they occur in. For the German texts, we obtain the following distribution:

```
fdist <- rev(table(FreqDE$cols$nnzero))
head(fdist, 15) # how many distinct words occur in exactly <k> texts
```

```
##
##  75  74  73  72  71  70  69  68  67  66  65  64  63  62  61
## 487 184 130 125 115 128 128 107 110 123  94 119 125 136 121
```

```
tail(fdist, 10) # words that occur only in a single text are completely useless
```

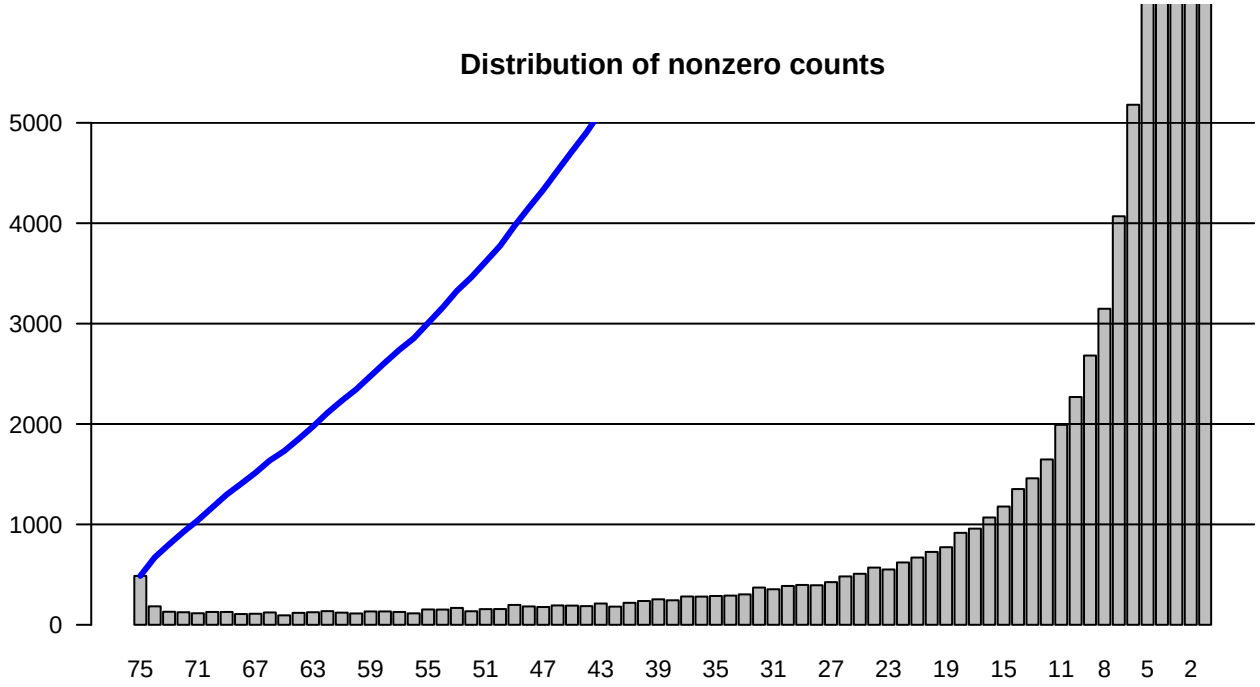
```
##
##     10      9      8      7      6      5      4      3      2      1
##   2269   2682   3148   4069   5180   6943  10048  16324  33477 145725
```

```
head(cumsum(fdist), 15) # and how many occur in <k> or more texts
```

```
##   75   74   73   72   71   70   69   68   67   66   65   64   63   62   61
##  487  671  801  926 1041 1169 1297 1404 1514 1637 1731 1850 1975 2111 2232
```

Visualize the full distribution of nonzero counts with a bar plot, showing the cumulative type count of words occurring in $\geq k$ texts as a blue line.

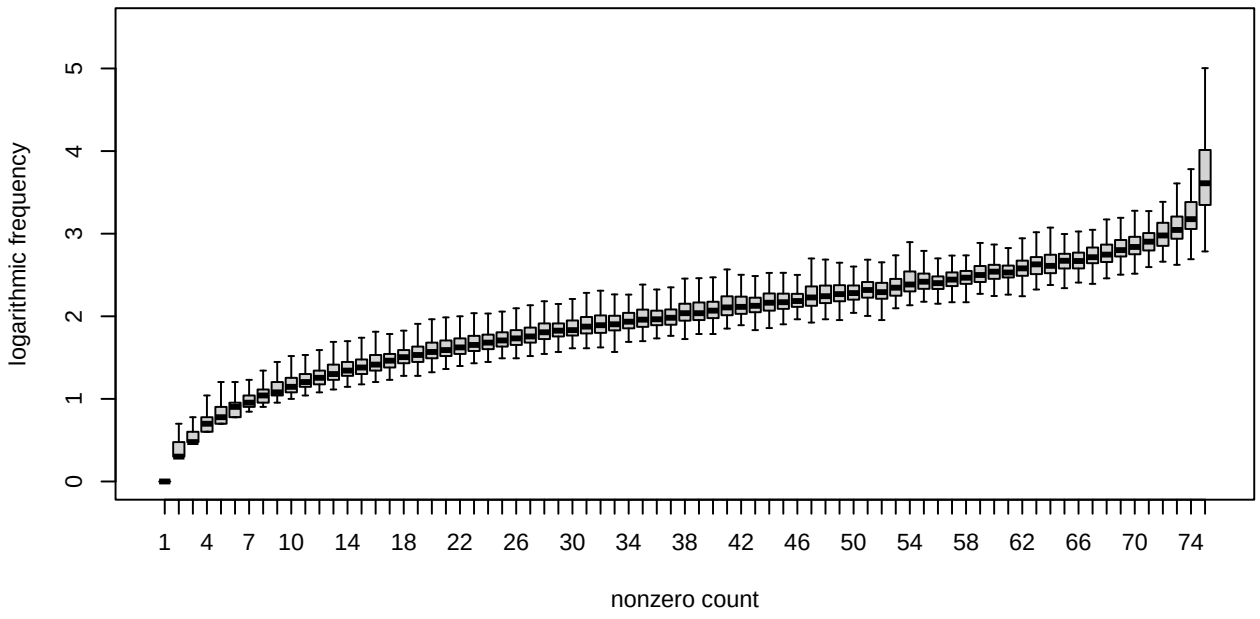
```
x <- barplot(fdist, ylim=c(0, 5000), las=1, main="Distribution of nonzero counts")
abline(h=(1:5)*1000)
lines(x, cumsum(fdist), lwd=3, col="blue")
```



A reasonable criterion for feature selection could be to include only words w_i that occur in at least 50% of the texts (i.e. 35 or more) in order to avoid author- or genre-specific features. This results in a set of approx. 6000 features, which our replication of (Jannidis et al. 2015) above shows to be in a reasonable range. If we require 80% or more nonzero values (i.e. occurrences in 60 or more texts), we are close to optimal performance at $n_w \approx 2000$ feature dimensions.

Nonzero counts correlate strongly with frequency, but the ordering will not be exactly the same, as can be seen from the boxplot below. Whenever the whiskers of two boxes overlap, there will be a substantial number of w_i whose ordering by nonzero count is different from their frequency ordering.

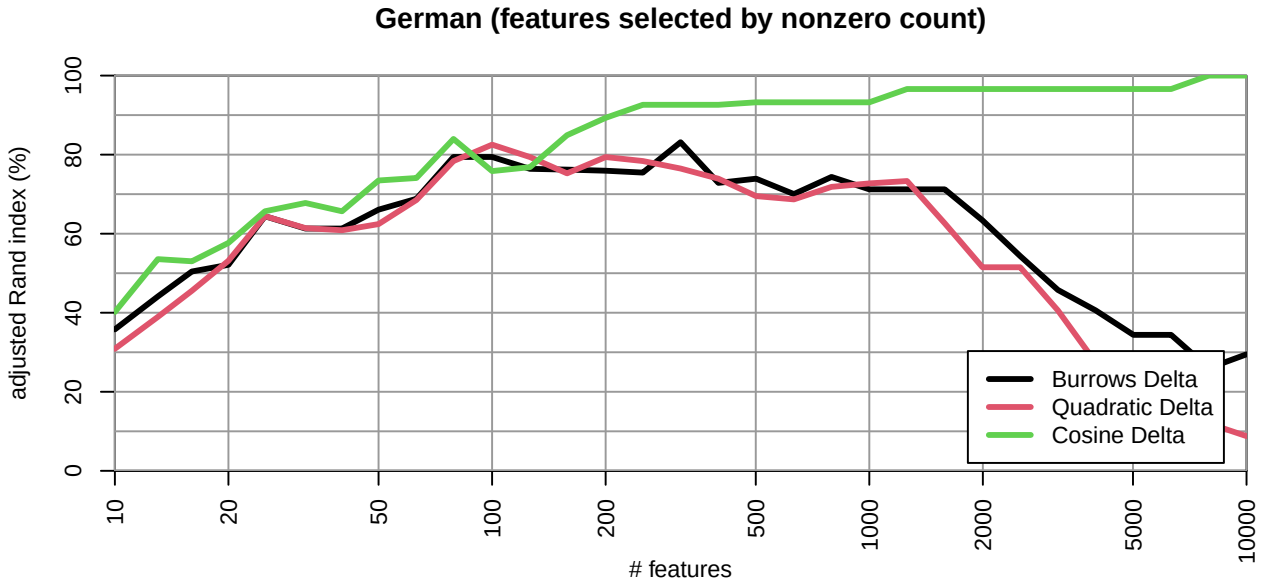
```
with(FreqDE$cols, boxplot(log10(f) ~ nnzero, pch=NA, whisklty="solid",
                          xlab="nonzero count", ylab="logarithmic frequency"))
```



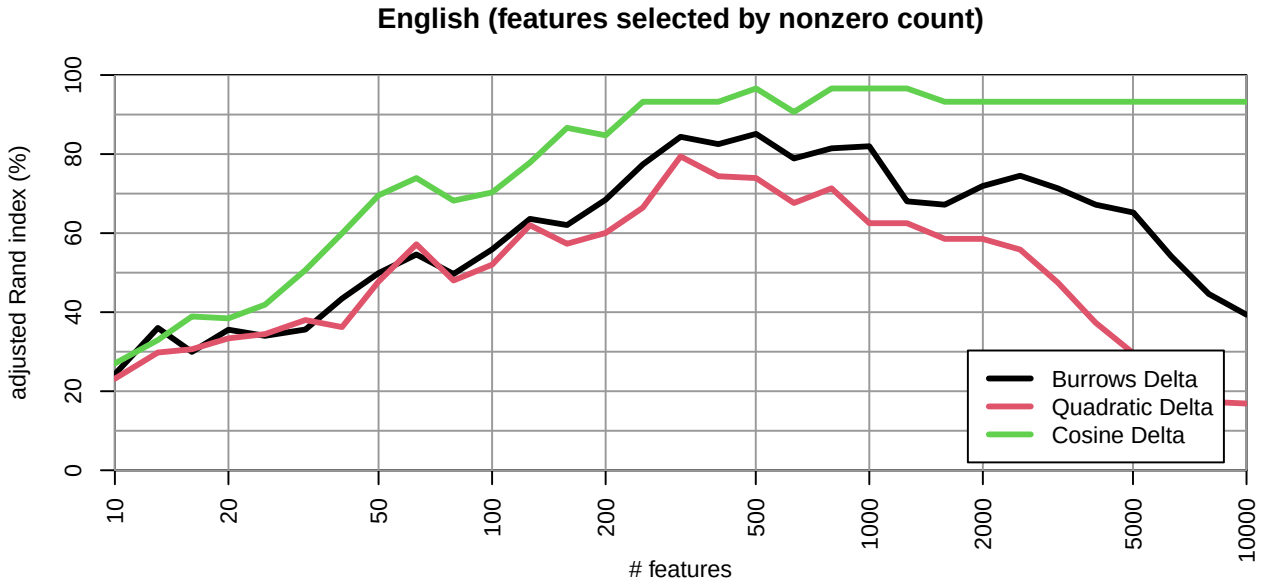
Surprisingly, the empirical evaluation shown below reveals that nonzero-based feature selection doesn't agree

well with Δ_B and Δ_Q. Results with $\Delta_\angle$ are better and even lead to a small improvement on the German data set.

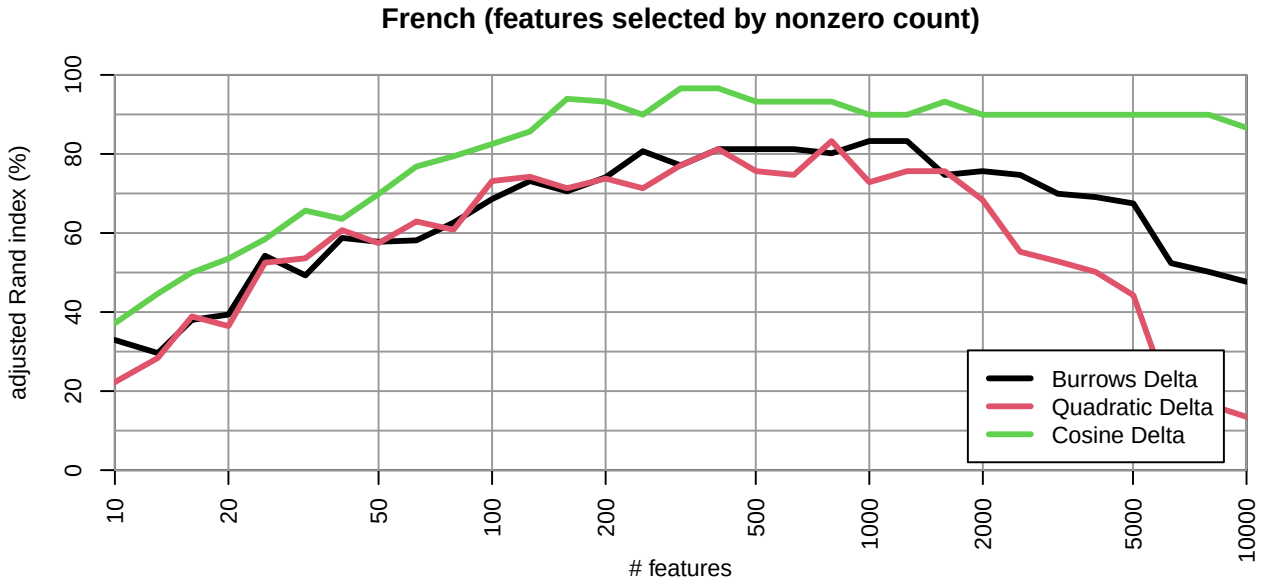
```
idxDE <- with(FreqDE$cols, order(nnzero, f, decreasing=TRUE))
tmp <- zDE[, idxDE]
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="German (features selected by nonzero count)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```




```
idxEN <- with(FreqEN$cols, order(nnzero, f, decreasing=TRUE))
tmp <- zEN[, idxEN]
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)", main="English (features selected by nonzero cou
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```




```
idxFR <- with(FreqFR$cols, order(nnzero, f, decreasing=TRUE))
tmp <- zFR[, idxFR]
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)", main="French (features selected by nonzero count
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



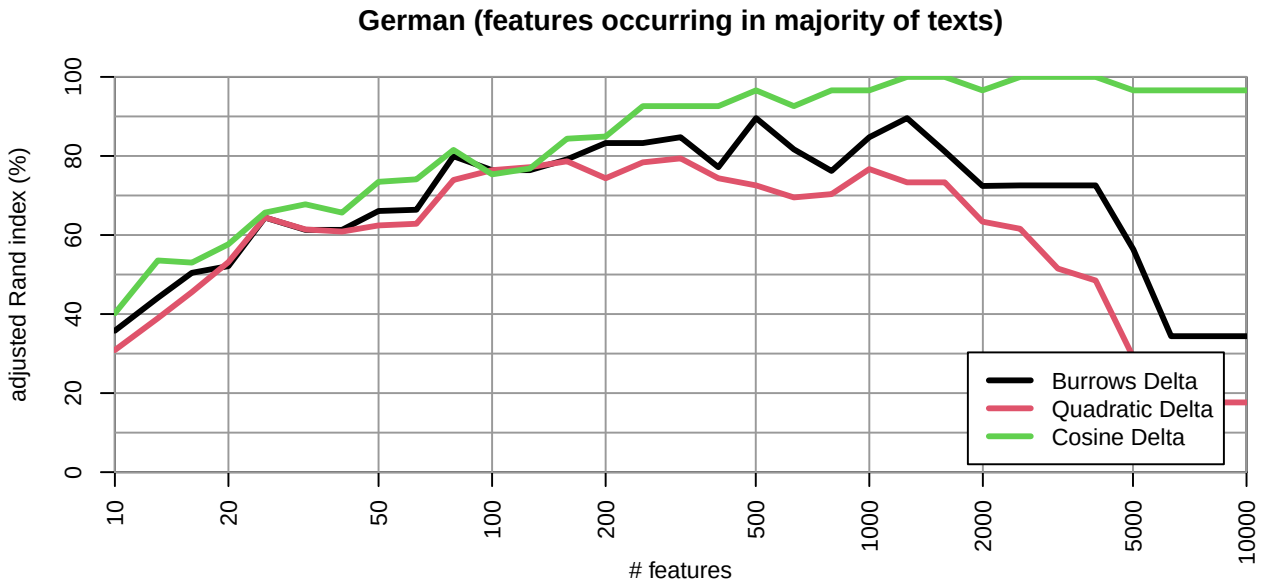
It might make sense to use the document frequency criterion to discard potentially idiosyncractic words (e.g. non-standard spellings used by a particular author or publisher) and those found only among a specific group (e.g. typographic conventions, lack of normalization for some texts, words specific to a sub-genre such as Jules Verne's science-fiction). A nonzero count of $df \geq 38$ (i.e. words must occur in the majority of texts) seems reasonable and leaves a sufficient number of features for frequency-based selection:

```
idx.DE <- FreqDE$cols$nnzero >= 38
idx.EN <- FreqEN$cols$nnzero >= 38
idx.FR <- FreqFR$cols$nnzero >= 38
data.frame(German=sum(idx.DE), English=sum(idx.EN), French=sum(idx.FR),
           row.names="features with df >= 38")
```

```
##                         German English French
## features with df >= 38   6249    6334   5729
```

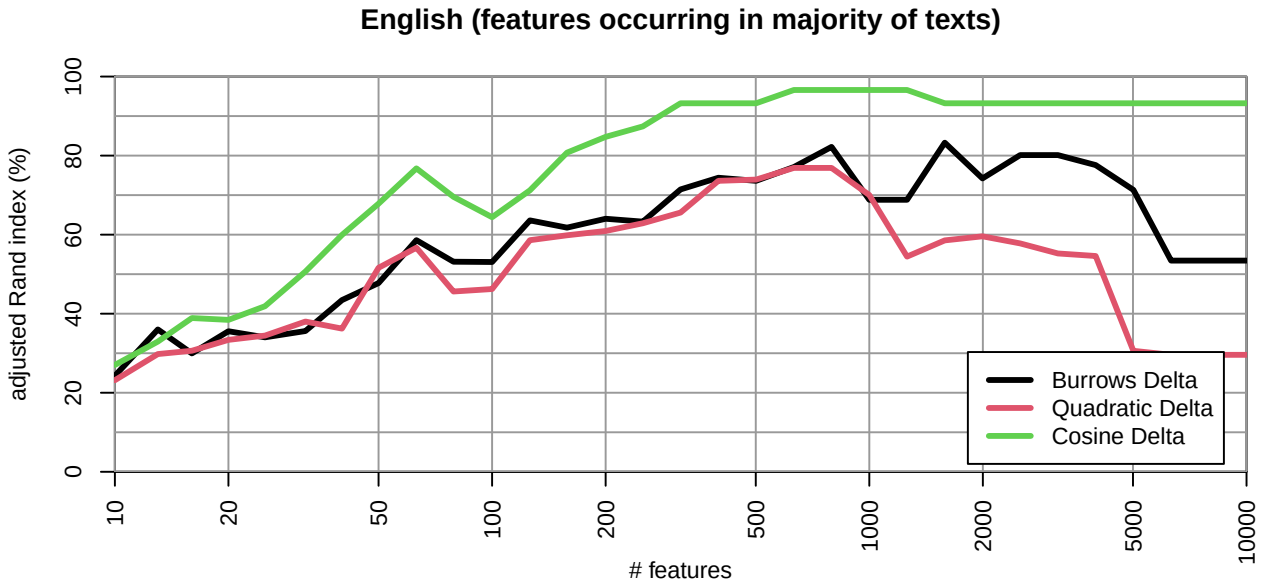
The nonzero threshold is clearly detrimental for Δ_B and Δ_Q, but seems to stabilize the good performance of $\Delta_\angle$. On the French data, performance still deteriorates for $n_w \geq 1000$, but the threshold helps to ensure that no circumstantial features (such as typographic conventions) are exploited. It is not clear, whether this strategy is superior to feature selection based on nonzero counts if $\Delta_\angle$ is used, though.

```
tmp <- zDE[, idx.DE]
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="German (features occurring in majority of texts)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```

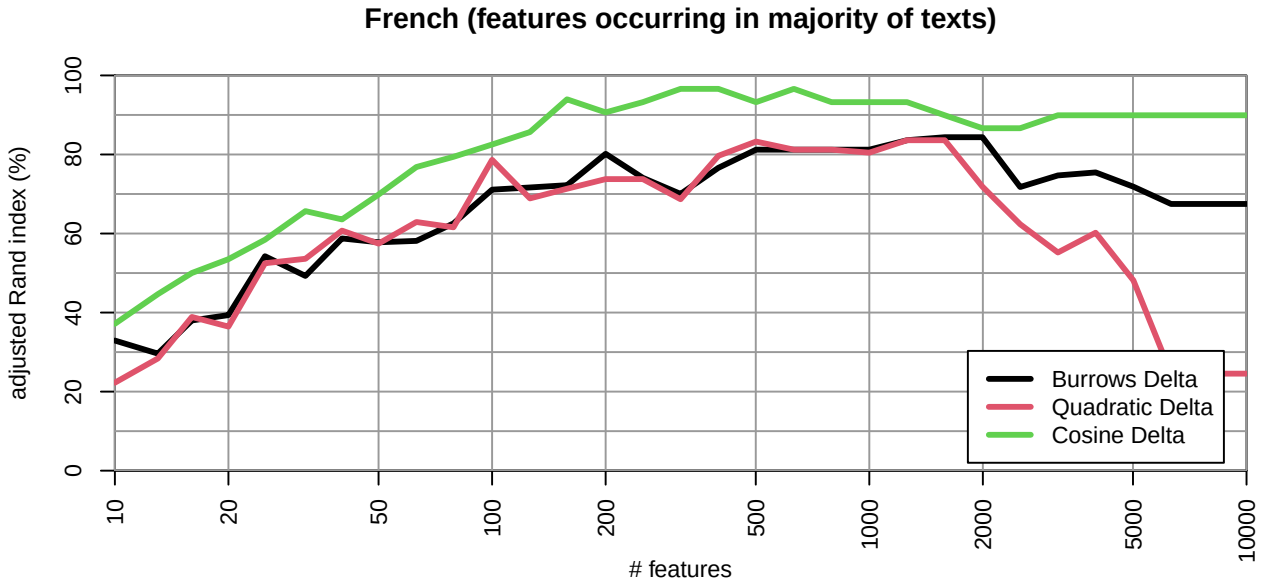



```
tmp <- zEN[, idx.EN]
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="English (features occurring in majority of texts)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
```

```
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```

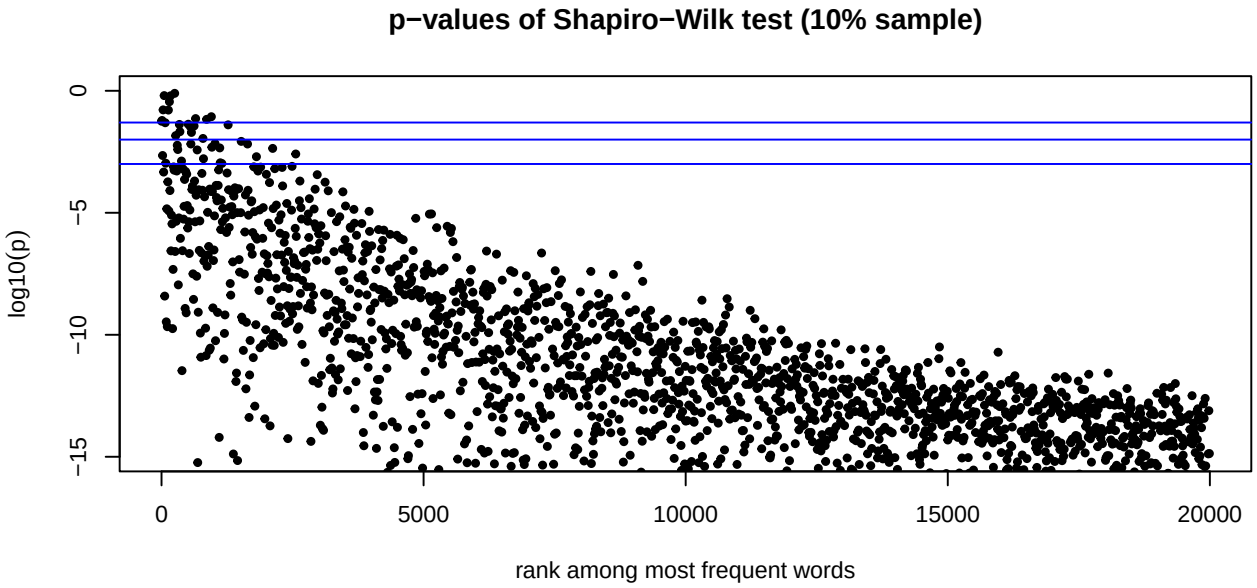



```
tmp <- zFR[, idx.FR]
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="French (features occurring in majority of texts)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



Yet another approach for ordering or selecting features is to assess whether their distribution is approximately Gaussian using e.g. the Shapiro-Wilk normality test (Shapiro and Wilk 1965; Shapiro, Wilk, and Chen 1968). Since the test is computationally quite expensive, we apply it only to the 20,000 most frequent words.

```
W.pval <- apply(zDE[, 1:20000], 2, function (x) {
  shapiro.test(x)$p.value
})
idx <- seq(1, 20000, 10)
plot(idx, log10(W.pval[idx]), pch=20, ylim=c(-15, 0),
     xlab="rank among most frequent words", ylab=expression(log10(p)),
     main="p-values of Shapiro-Wilk test (10% sample)")
abline(h=log10(c(.05, .01, .001)), col="blue")
```



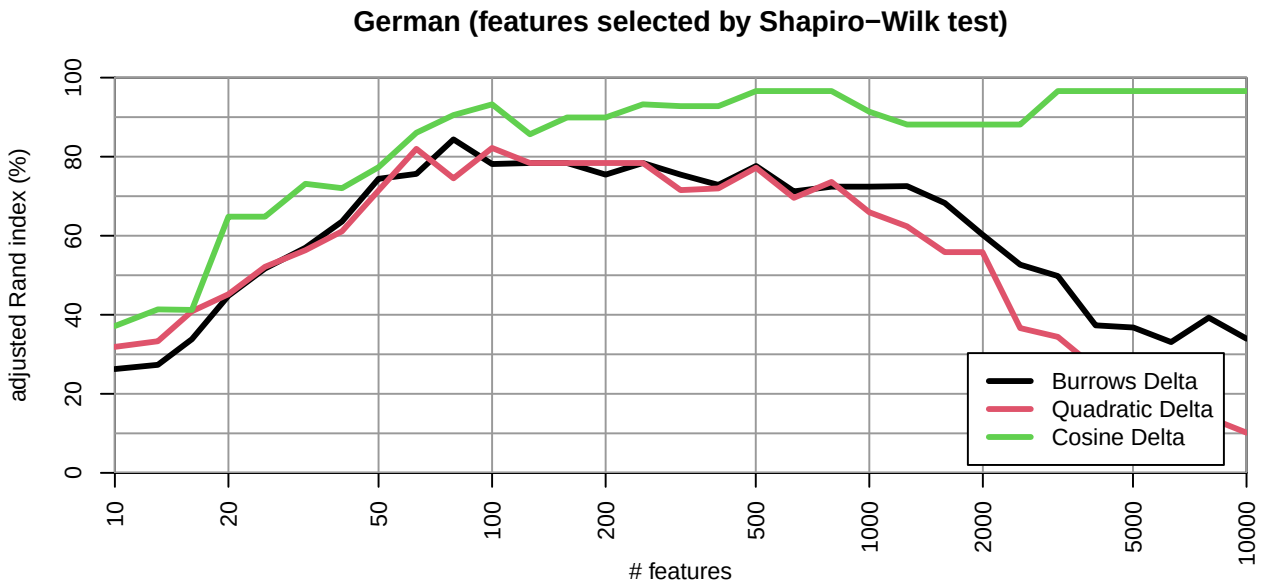
The blue lines indicate the common significance levels $\alpha = .05, .01, .001$. Since there are only a few words that do not deviate significantly from a Gaussian distribution, the test cannot be used as a criterion to determine dimensionality; it would leave us with $n_w < 500$ features. Another problems is that p-values depend strongly on the number of texts in the collection.

```
sapply(c(.05, .01, .001), function (alpha) sum(W.pval >= alpha))
```

```
## [1] 134 249 468
```

The Shapiro-Wilk test might be useful as a mathematically justified quality measure for feature selection, though. Here, feature words are ordered by the normality of their distribution across texts, operationalized in terms of the Shapiro-Wilk p-value. We then select the first n_w features according to this measure.

```
tmp <- zDE[, order(W.pval, decreasing=TRUE)] # first 20,000 columns
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="German (features selected by Shapiro-Wilk test)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



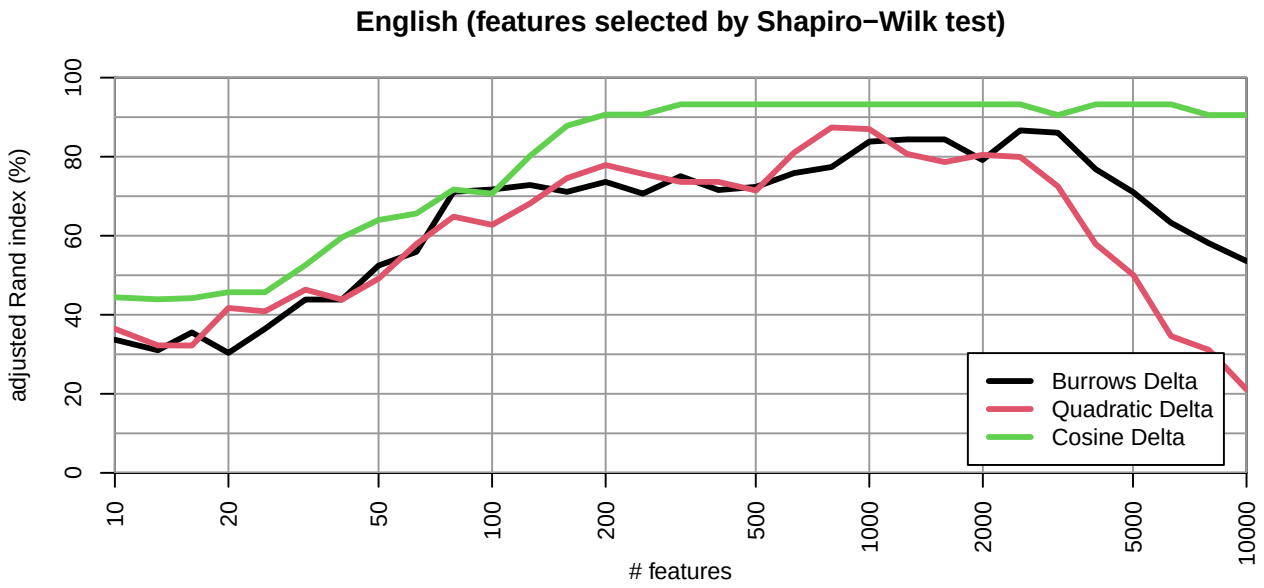
The plot indicates that the $n_w \approx 200$ words that do not deviate significantly from a Gaussian distribution are indeed highly effective features: clustering quality improves faster than for feature selection based on frequency or nonzero counts. However, it soon begins to deteriorate as more features are added. One possible explanation is that the Shapiro-Wilk test weeds out words that are indicative of a sub-genre, historical period or orthographic convention and therefore exhibit a bimodal distribution. Since these features help to narrow down the range of possible authors, they provide useful information for the authorship attribution task.

The corresponding plots for English and French largely confirm this interpretation.

```
W.pval <- apply(zEN[, 1:20000], 2, function (x) {
  shapiro.test(x)$p.value
})
sapply(c(.05, .01, .001), function (alpha) sum(W.pval >= alpha))
```

```
## [1] 184 294 552
```

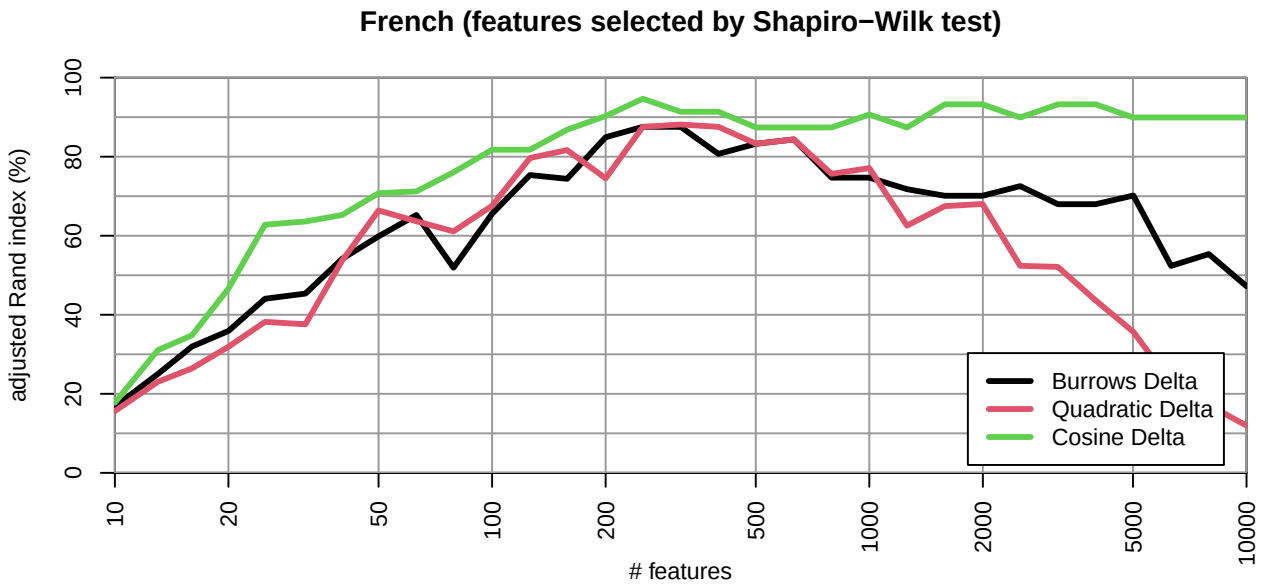
```
tmp <- zEN[, order(W.pval, decreasing=TRUE)]
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="English (features selected by Shapiro-Wilk test)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```




```
W.pval <- apply(zFR[, 1:20000], 2, function (x) {
  shapiro.test(x)$p.value
})
sapply(c(.05, .01, .001), function (alpha) sum(W.pval >= alpha))
```

```
## [1] 130 245 454
```

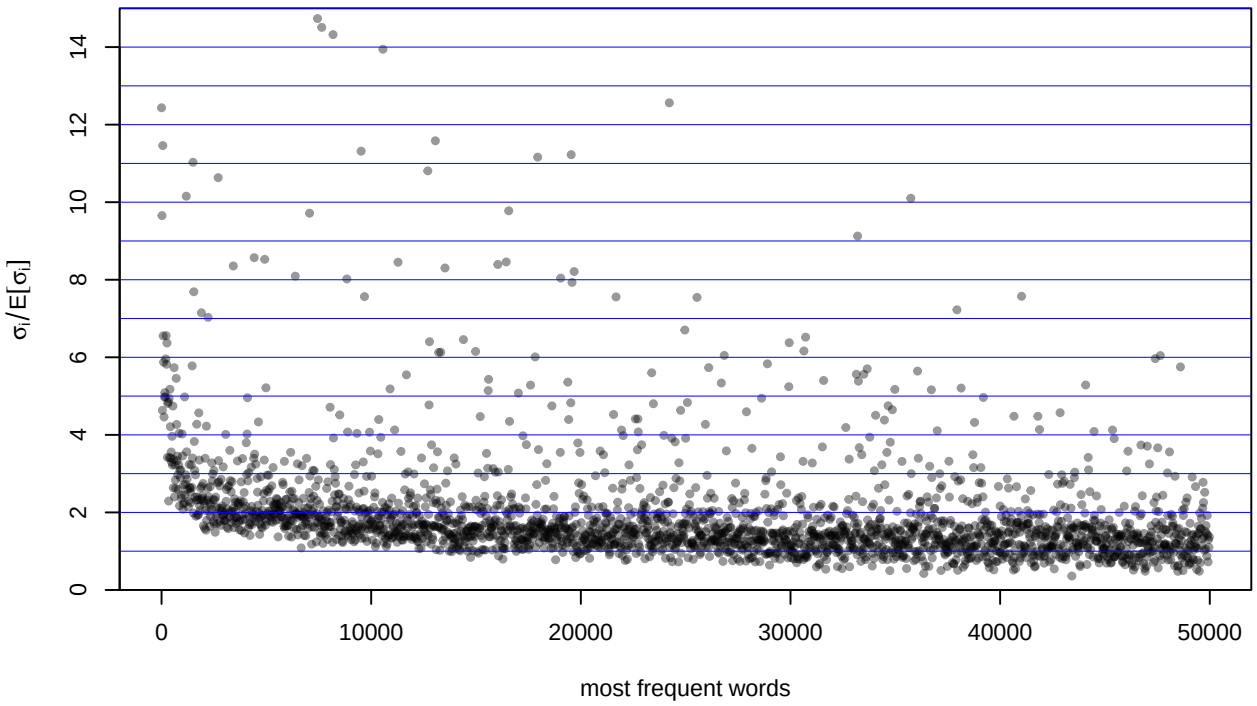
```
tmp <- zFR[, order(W.pval, decreasing=TRUE)] # first 20,000 columns
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="French (features selected by Shapiro-Wilk test)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



An important goal of feature selection is to choose words that provide useful information for authorship attribution rather than just random noise. We can operationalize this by comparing the variance (or standard deviation) of relative frequencies $f_i(D)$ with the variance expected under a binomial sampling model for the texts. If the empirical variance is not substantially larger than the expected value, frequency differences between the texts are little more than random noise.

It is difficult to compute a precise value for the expected variance of $f_i(D)$ if the collection contains texts of different lengths, but we can obtain a reasonable approximation by assuming equal document sizes. Writing $k_i(D) = n_D \cdot f_i(D)$ for the number of occurrences of word w_i in document D, $k_i = \sum_{D \in \mathcal{D}} f_i(D)$ for its overall frequency and $n = \sum_{D \in \mathcal{D}} n_D$ for the token count of the entire text collection, the MLE for the occurrence probability of w_i is $p_i = k_i / n$ and the average text size is $\bar{n} = n / n_{\mathcal{D}}$. The expected variance of $k_i(D)$ is thus $\bar{n} p_i (1 - p_i) \approx \bar{n} p_i$. The corresponding variance of $f_i(D) = k_i(D) / n_D \approx k_i(D) / \bar{n}$ is $\bar{n} p_i / \bar{n}^2 = p_i / \bar{n}$, and its expected standard deviation is $E[\sigma_i] = \sqrt{p_i / \bar{n}}$.

```
n <- sum(FreqDE$rows$f)
p <- FreqDE$cols$f / n
Esigma <- sqrt(p / (n / nrow(FreqDE))) # expected s.d.
sigma <- apply(FreqDE$S, 2, sd)        # observed s.d.
rel <- sigma / Esigma
idx <- seq(1,50000,20)
plot(idx, rel[idx], ylim=c(0,15), pch=20, col="#00000066", yaxs="i",
     xlab="most frequent words", ylab=expression(sigma[i] / E * group("[",sigma[i],"]")))
abline(h=1:15, col="blue", lwd=.5)
```



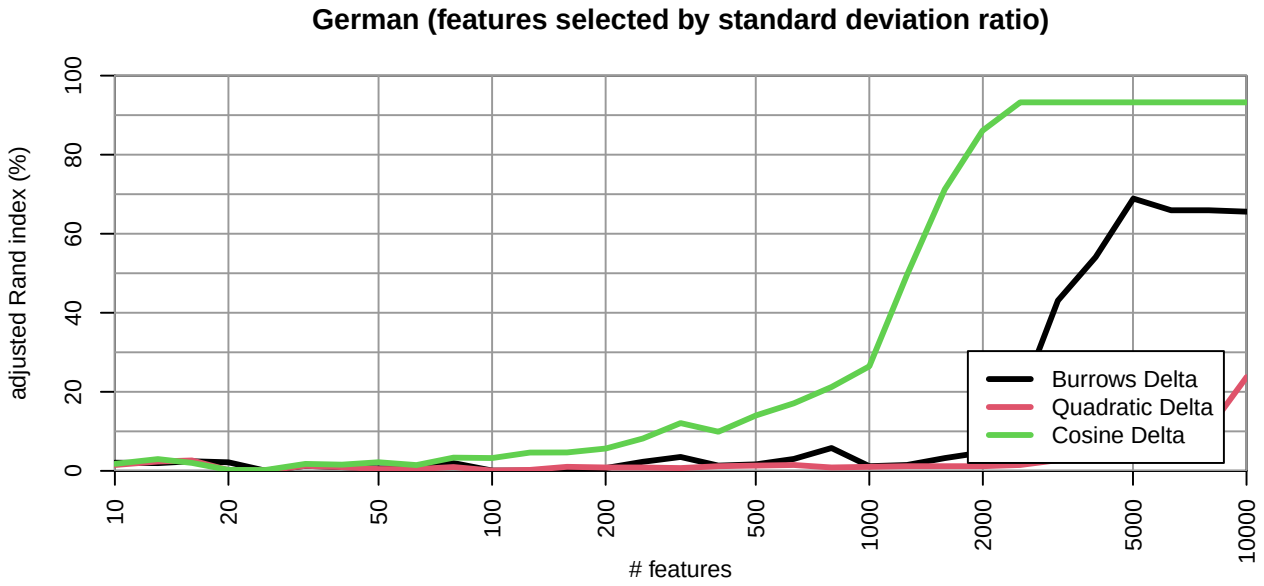
Even very low-frequency words can have a s.d. σ_i that is considerably larger than $E[\sigma_i]$, so we cannot use a simple criterion such as $\sigma_i \geq 2 \cdot E[\sigma_i]$ or a variance test to choose the dimensionality n_w.

```
round(quantile(rel, c(0, .5, .8, .85, .9, .95, .99, .995, .999, 1)), 2)
```

```
##    0%   50%   80%   85%   90%   95%   99% 99.5% 99.9%  100%
##  0.18  1.06  1.72  1.95  2.26  2.89  5.15  7.09 15.23 54.41
```

Instead, we use the ratio $\sigma_i / E[\sigma_i]$ as a relevance measure for features and select the first n_w words. Unfortunately, this strategy turns out to be counter-productive, even if we only choose from the most frequent 20,000 words:

```
tmp <- zDE[, order(rel[1:20000], decreasing=TRUE)]
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="German (features selected by standard deviation ratio)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



TODO: can we find a principled method for choosing a suitable cutoff point n? e.g. based on approximate normality, sparseness, etc; will take "size" of the corpus/texts into account, perhaps indirectly through empirical distributions

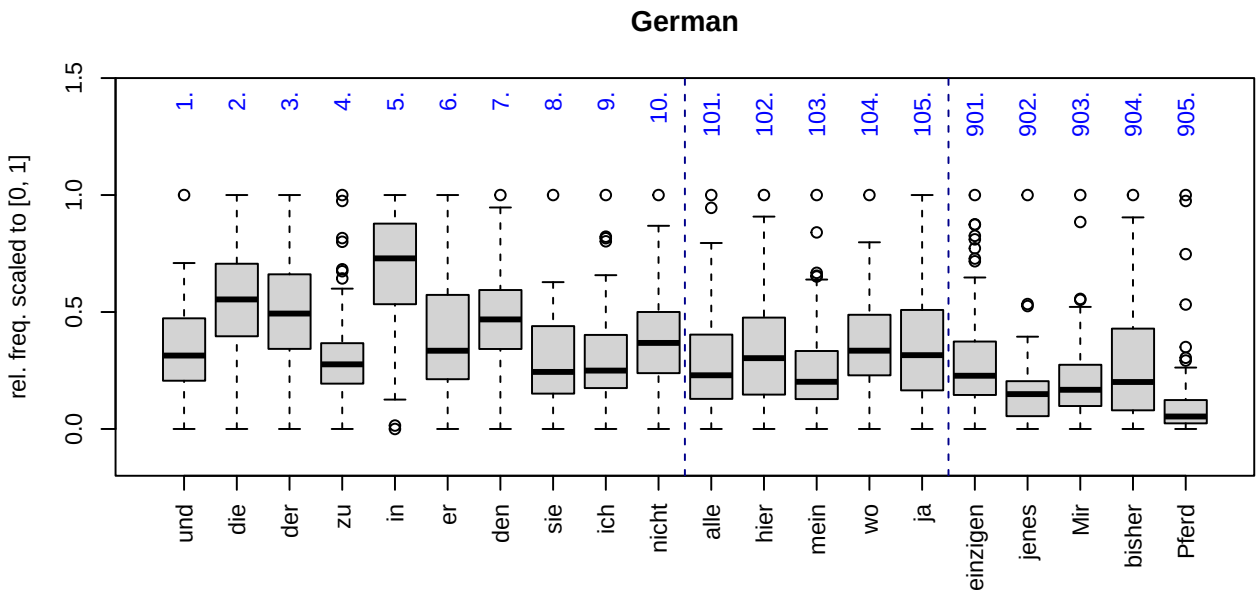
2.4 Are there alternatives to standardization?

In machine-learning approaches, it is not uncommon to scale sparse features (such as word frequencies) to the range $[0, 1]$ rather than standardizing them. This will reduce the weight of low-frequency words (with smaller σ_i) to some degreee. More importantly, the transformation preserves sparseness of the feature vectors because $f_i(D) = 0$ is mapped to 0. Such range scaling has also been used in our supervised feature selection experiments.

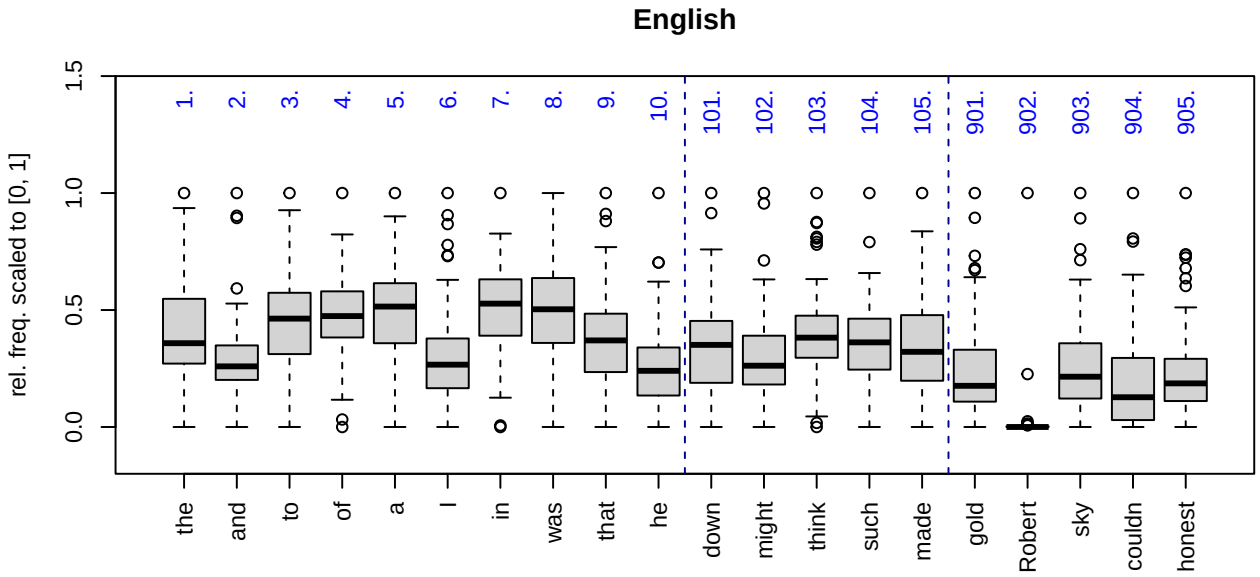
```
scale.range <- function (x, a=0, b=1) {
  l <- apply(x, 2, min) # minimum value of each column vector
  u <- apply(x, 2, max) # maximum value
  tmp <- scale(x, center=l, scale=(u - l) / (b - a)) # range [0, b-a]
  scale(tmp, center=rep(-a, ncol(x)), scale=FALSE)   # shift to range [a, b]
}
```

The boxplot below shows how the transformation affects feature distributions. Note that the scaled $f_i(D)$ of high-frequency words spread across the entire range, while those of low-frequency words tend to be closer to 0 with only a few outlier values above 0.5.

```
tmp <- scale.range(zDE)

r.vals <- c(1:10,101:105,901:905) # ranks of selected features
boxplot(tmp[, r.vals], las=3, yaxs="i", ylim=c(-.2, 1.5),
        ylab="rel. freq. scaled to [0, 1]", main="German")
abline(v=c(10.5,15.5), col="darkblue", lty="dashed")
text(seq_along(r.vals), par("usr")[4], sprintf("%d.  ", r.vals),
     adj=c(1, 0.5), srt=90, col="blue")
```



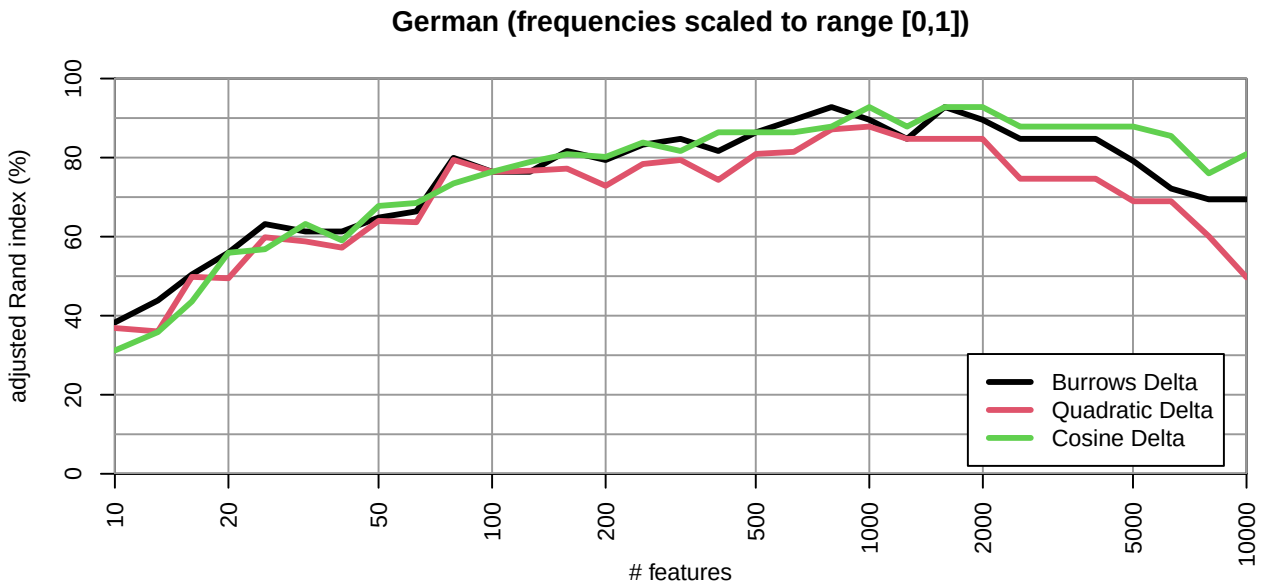

```
tmp <- scale.range(zEN)

r.vals <- c(1:10,101:105,901:905) # ranks of selected features
boxplot(tmp[, r.vals], las=3, yaxs="i", ylim=c(-.2, 1.5),
        ylab="rel. freq. scaled to [0, 1]", main="English")
abline(v=c(10.5,15.5), col="darkblue", lty="dashed")
text(seq_along(r.vals), par("usr")[4], sprintf("%d.  ", r.vals),
     adj=c(1, 0.5), srt=90, col="blue")
```



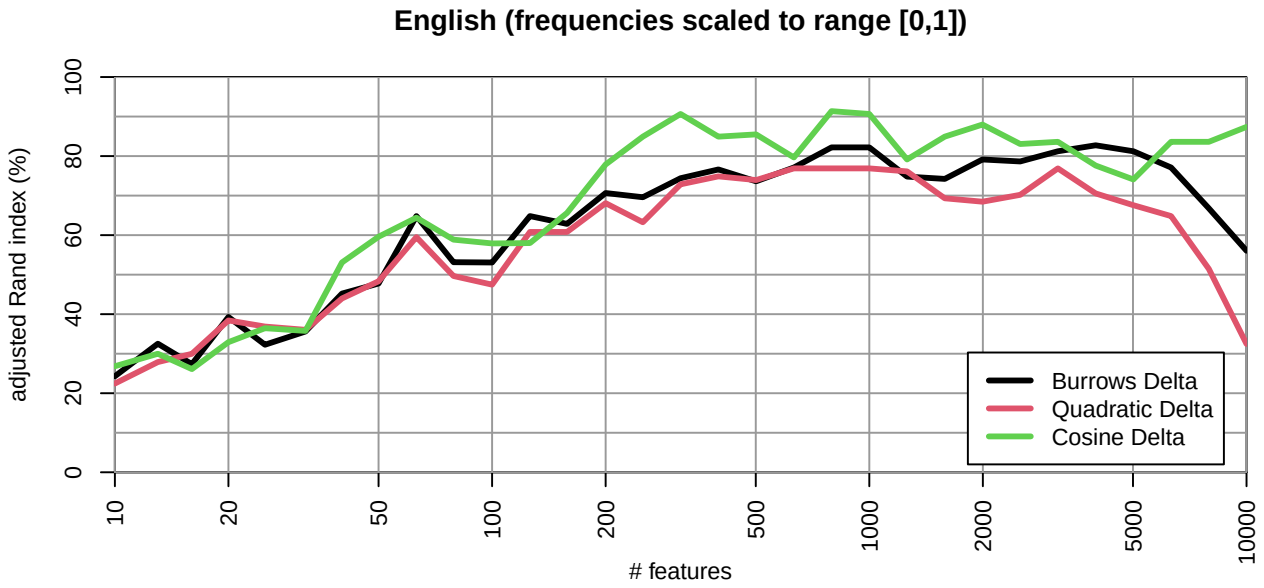
The plots below evaluate the clustering quality of Delta measures based on range-scaled frequencies for the three corpora. Features are selected by word frequency, which has proven to be the best strategy so far.

```
tmp <- scale.range(zDE)
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="German (frequencies scaled to range [0,1])",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
```

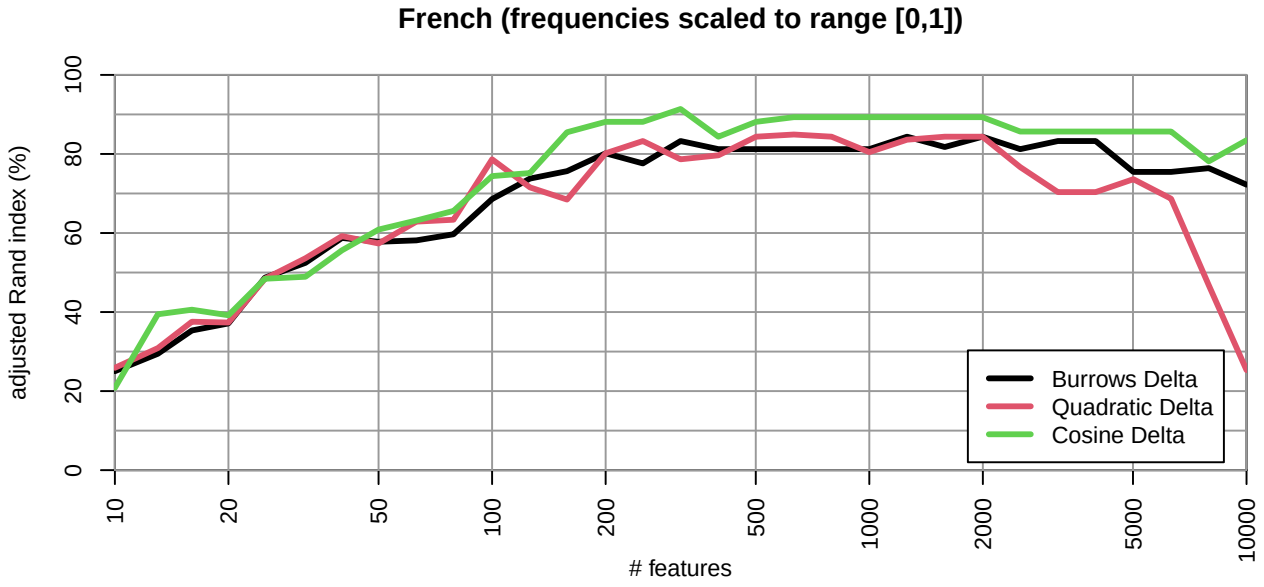
```
draw.grid()
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```




```
tmp <- scale.range(zEN)
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="English (frequencies scaled to range [0,1])",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```




```
tmp <- scale.range(zFR)
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="French (frequencies scaled to range [0,1])",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



A comparison with the original evaluation results shows that the range transformation has little effect on Δ_B and yields a moderate improvement for Δ_Q, which is now almost on par with Δ_B at least up to $n_w \approx 2000$.

The most interesting observation, however, is that the clustering quality of $\Delta_\angle$ decreases. For sparse vectors, cosine similarities are often interpreted as measures of feature overlap (because any coordinates with $x_i = 0$

or $y_i = 0$ do not contribute to the inner product $\mathbf{x}^T\mathbf{y}$). From this perspective, it is sensible to preserve sparseness of the vectors; standardization seems to change the interpretation of cosine similarity considerably.

The fact that Cosine Delta doesn't work with range-transformed frequencies suggests that its good performance is not connected to its inerpretation as a measure of feature overlap or spherical direction, but rather due to its implicit normalization of Euclidean distances. Thus, **the key factor for improving Delta measures seems to be vector normalization**.

The original purpose of standardization in the Delta measure was to "treat all of these words as markers of potentially equal power in highlighting the differences between one style and another" (Burrows 2002, 271). However, this is only the case for (squared) Euclidean distance, i.e. Argamon's Δ_Q. Enumerating the frequencies profile vectors in a text collection as $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(n_\mathcal{D})}$, the sum of squared Euclidean distances between all text pairs evalutes to

$$\sum_{i,j=1}^{n_\mathcal{D}} \|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\|_2^2 = \sum_{i,j=1}^{n_\mathcal{D}} \left(\|\mathbf{z}^{(i)}\|_2^2 + \|\mathbf{z}^{(j)}\|_2^2 - 2(\mathbf{z}^{(i)})^T\mathbf{z}^{(j)}\right) = 2n_\mathcal{D} \sum_{k=1}^{n_w}\sum_{i=1}^{n_\mathcal{D}} (z_k^{(i)})^2 - 2\sum_{i,j=1}^{n_\mathcal{D}} (\mathbf{z}^{(i)})^T\mathbf{z}^{(j)}$$

where the last term vanishes because $\sum_i \mathbf{z}^{(i)} = \mathbf{0}$ after standardization. Each word feature w_k thus contributes an overall amount proportional to $\sum_D z_k(D)^2 = n_\mathcal{D} - 1$ to the squared Euclidean distances.

```
colSums(zDE[,1:12]^2)
```

```
##   und   die   der    zu    in    er   den   sie   ich nicht  sich   das
##    74    74    74    74    74    74    74    74    74    74    74    74
```

However, standardization does not normalize the contributions of different features to Δ_B, i.e. to pairwise Manhattan distances, in the same way. Based on his probabilisitic interpretation, Argamon (2008, 137) suggests to center each feature on its median value m_i and scale by average absolute deviation from m_i (NB: this is *not* the well-known robust MAD = *median* absolute deviation estimator). This is appropriate under the assumption of a Laplace distribution, but does not give each word equal weight in the pairwise Manhattan distances. To derive an appropriate scaling factor, we begin by noting that the contributions of different features are additive:

$$\sum_{i,j=1}^{n_\mathcal{D}} \|\mathbf{f}^{(i)} - \mathbf{f}^{(j)}\|_1 = \sum_{k=1}^{n_w}\sum_{i,j=1}^{n_\mathcal{D}} |f_k^{(i)} - f_k^{(j)}|$$

Denoting the values of a given feature $k \in 1, \ldots, n_w$ across the text collection by $x_i = f_k^{(i)}$, we can simplify the computation if we rearrange the sequence (x_i) in ascending order, written as $y_1 \le y_2 \le \cdots \le y_{n_\mathcal{D}}$.

$$\sum_{i,j} |x_i - x_j| = \sum_{i,j} |y_i - y_j| = 2\sum_{i<j} (y_j - y_i) = 2\sum_j y_j(j-1) - 2\sum_i y_i(n_\mathcal{D} - i)$$

In the last step, we have made use of the fact that each y_j occurs in $j-1$ terms ($i = 1, \ldots, j-1$) in the summation, and each y_i occurs in $n_\mathcal{D} - i$ terms ($j = i+1, \ldots, n_\mathcal{D}$). Changing the index in the first summation from j to i, we obtain a total contribution of

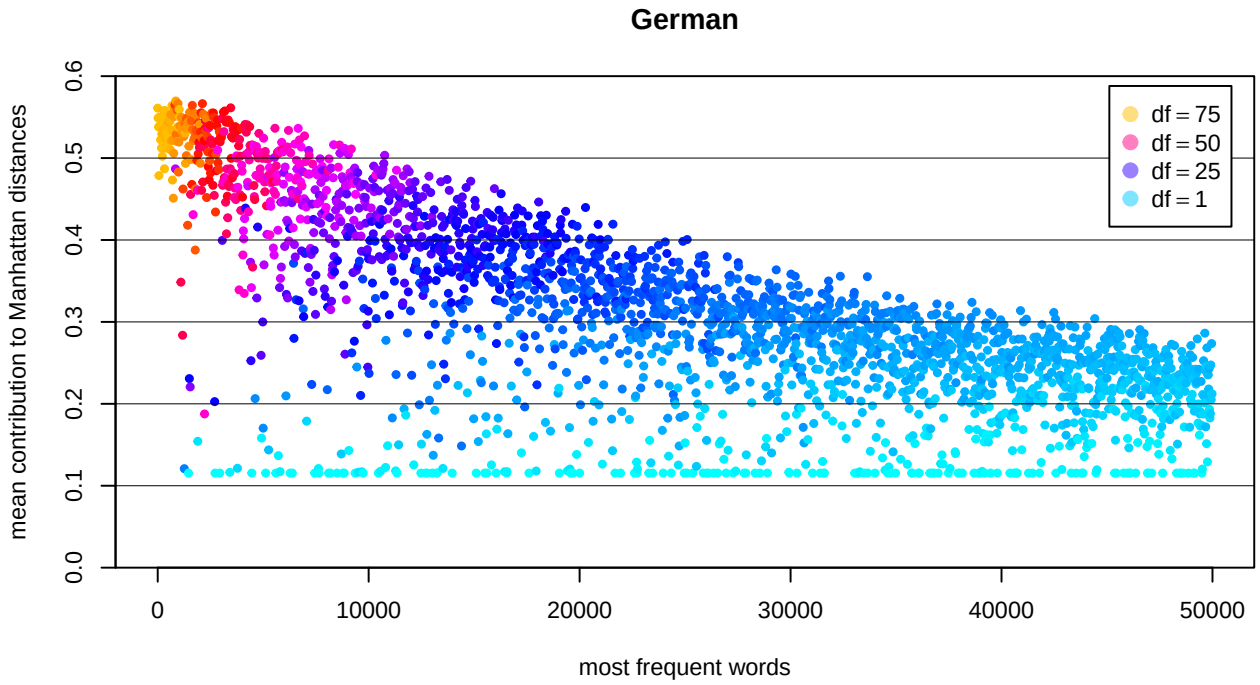
$$\sum_{i=1}^{n_\mathcal{D}} y_i(2i - n_\mathcal{D} - 1)$$

for feature w_k with values $y_1 \le y_2 \le \cdots \le y_{n_\mathcal{D}}$. The function `absdiff()` implements this calculation – taking the average rather than the sum over all text pairs in order to obtain a scaling effect similar to standardization – and can be applied to the columns of a document-term matrix.

```
absdiff <- function (x) {
  y <- sort(x)
  n <- length(y)
  sum(y * (2*(1:n) - n - 1)) / (n * (n-1))
}
```

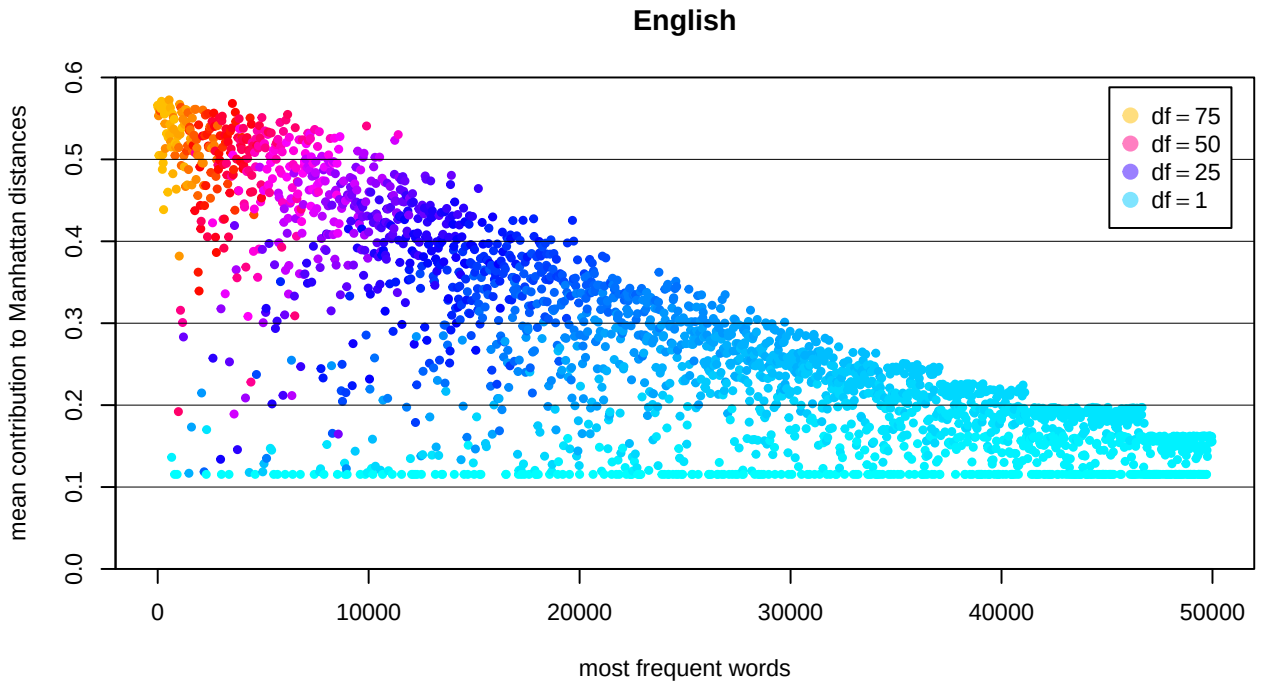
The following plot illustrates the contributions of standardized features to pairwise Manhattan distances between the German texts. The minimum value corresponds to features that occur just in a single text and therefore have identical value distributions after the z-transformation.

```
idx <- seq(1,50000,20)
col.solid <- rainbow(75, start=3/6, end=.75/6, alpha=.5)
col.alpha <- rainbow(75, start=3/6, end=.75/6, alpha=1)

rel <- apply(zDE[, idx], 2, absdiff)
plot(idx, rel, pch=20, col=col.alpha[FreqDE$cols$nnzero[idx]], yaxs="i", ylim=c(0, .6),
     xlab="most frequent words", ylab="mean contribution to Manhattan distances", main="German")
abline(h=seq(0,.6,.1), col="black", lwd=.5)
legend("topright", inset=.02, bg="white", legend=expression(df==75, df==50, df==25, df==1),
       pch=20, pt.cex=2, col=col.solid[c(75,50,25,5,1)])
```

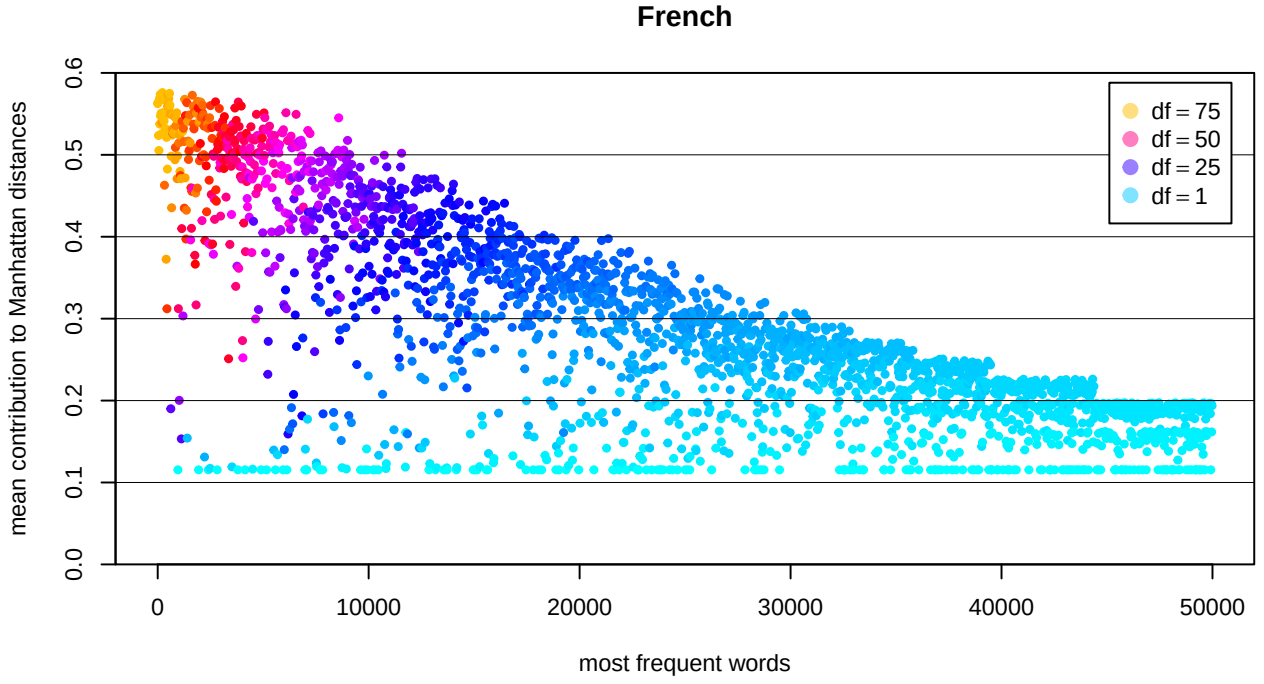


Patterns are very similar for English and French:

```
rel <- apply(zEN[, idx], 2, absdiff)
plot(idx, rel, pch=20, col=col.alpha[FreqEN$cols$nnzero[idx]], yaxs="i", ylim=c(0, .6),
     xlab="most frequent words", ylab="mean contribution to Manhattan distances", main="English")
abline(h=seq(0,.6,.1), col="black", lwd=.5)
legend("topright", inset=.02, bg="white", legend=expression(df==75, df==50, df==25, df==1),
       pch=20, pt.cex=2, col=col.solid[c(75,50,25,5,1)])
```




```
rel <- apply(zFR[, idx], 2, absdiff)
plot(idx, rel, pch=20, col=col.alpha[FreqFR$cols$nnzero[idx]], yaxs="i", ylim=c(0, .6),
     xlab="most frequent words", ylab="mean contribution to Manhattan distances", main="French")
abline(h=seq(0,.6,.1), col="black", lwd=.5)
legend("topright", inset=.02, bg="white", legend=expression(df==75, df==50, df==25, df==1),
       pch=20, pt.cex=2, col=col.solid[c(75,50,25,5,1)])
```

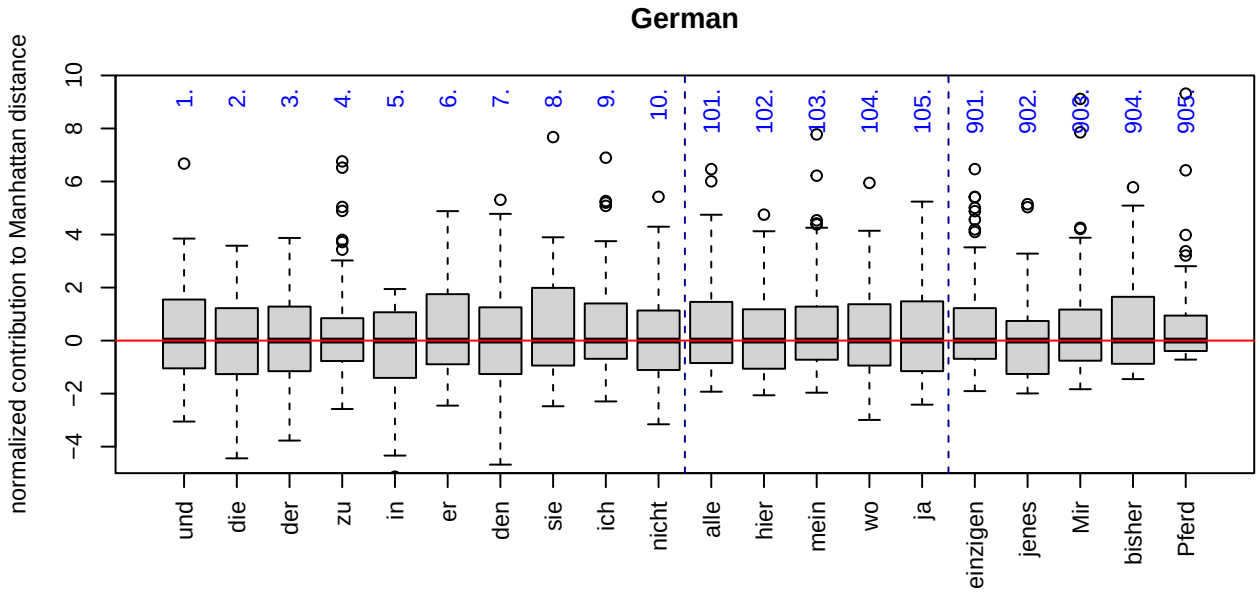


Obviously, the standardization chosen by Burrows gives slightly lower weight to less frequent words; it gives considerably lower weight to sparse words that occur just in a small number of texts. The **implicit relevance criteria** for Δ_B are thus (i) overall frequency and (ii) nonzero count.

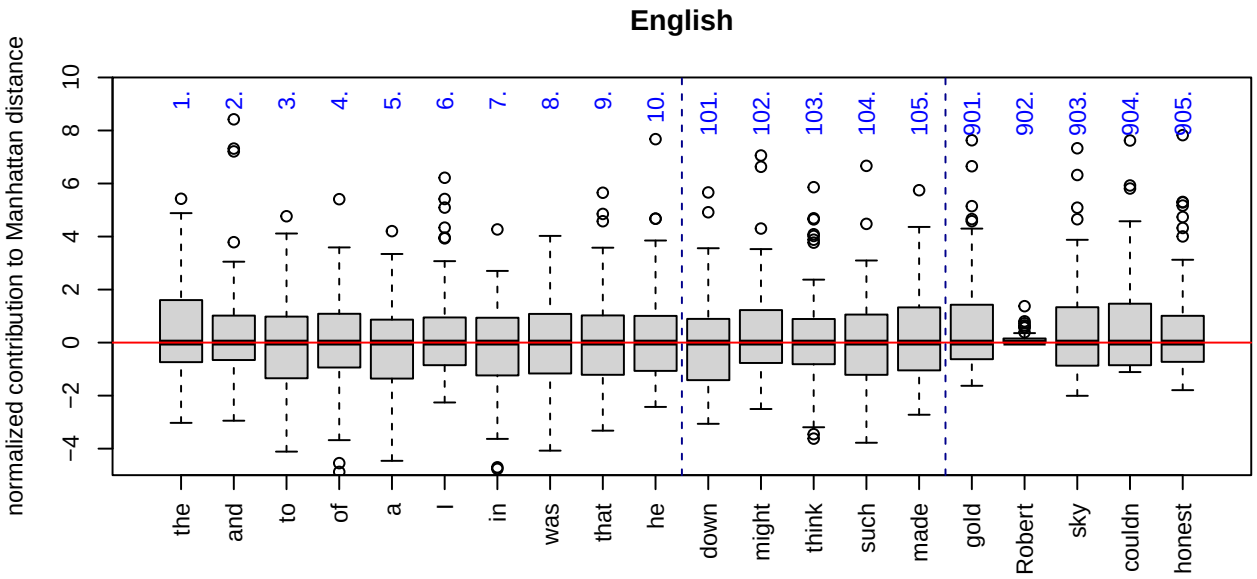
What happens if we rescale features with `scale.absdiff()` so they really make the same contribution to Manhattan distances?

```
scale.absdiff <- function (x, dim=Inf) {
  if (dim < ncol(x)) x <- x[, 1:dim]
  scale(x, center=apply(x, 2, median), # shift so that median value is zero
        scale=apply(x, 2, absdiff))    # and scale so that absdiff == 1
}
```

```
tmp <- scale.absdiff(zDE, dim=20000)

r.vals <- c(1:10,101:105,901:905) # ranks of selected features
boxplot(tmp[, r.vals], las=3, yaxs="i", ylim=c(-5, 10),
        ylab="normalized contribution to Manhattan distance", main="German")
abline(h=0, col="red")
abline(v=c(10.5,15.5), col="darkblue", lty="dashed")
text(seq_along(r.vals), par("usr")[4], sprintf("%d.  ", r.vals),
     adj=c(1, 0.5), srt=90, col="blue")
```

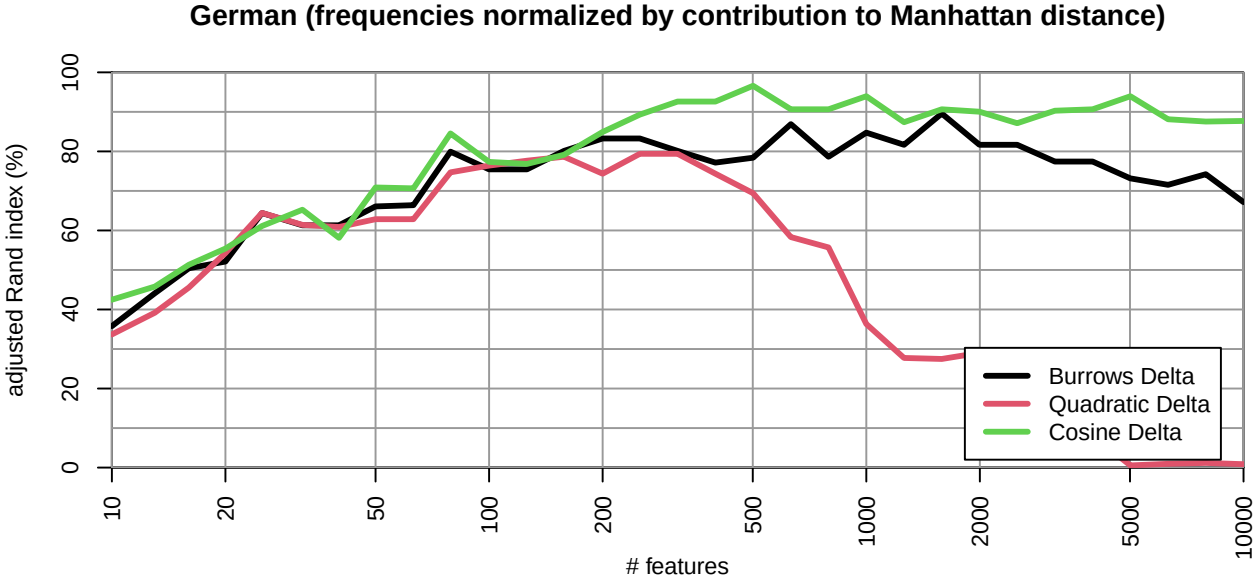



```
tmp <- scale.absdiff(zEN, dim=20000)
boxplot(tmp[, r.vals], las=3, yaxs="i", ylim=c(-5, 10),
        ylab="normalized contribution to Manhattan distance", main="English")
abline(h=0, col="red")
abline(v=c(10.5,15.5), col="darkblue", lty="dashed")
text(seq_along(r.vals), par("usr")[4], sprintf("%d.  ", r.vals),
     adj=c(1, 0.5), srt=90, col="blue")
```



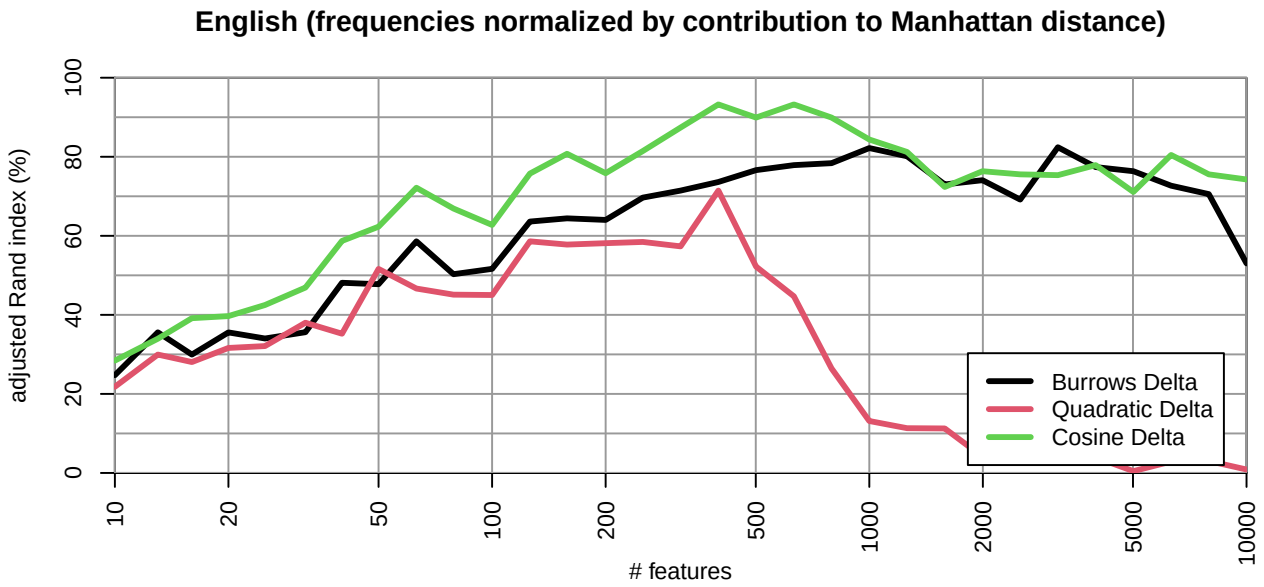
Scaling based on Argamon's average absolute deviation (boxplots not shown here) produces a similar result, but has a tendency to create larger outlier values for low-frequency words (e.g. *Pferd*).

```
tmp <- scale.absdiff(zDE, dim=20000)
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="German (frequencies normalized by contribution to Manhattan distance)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```

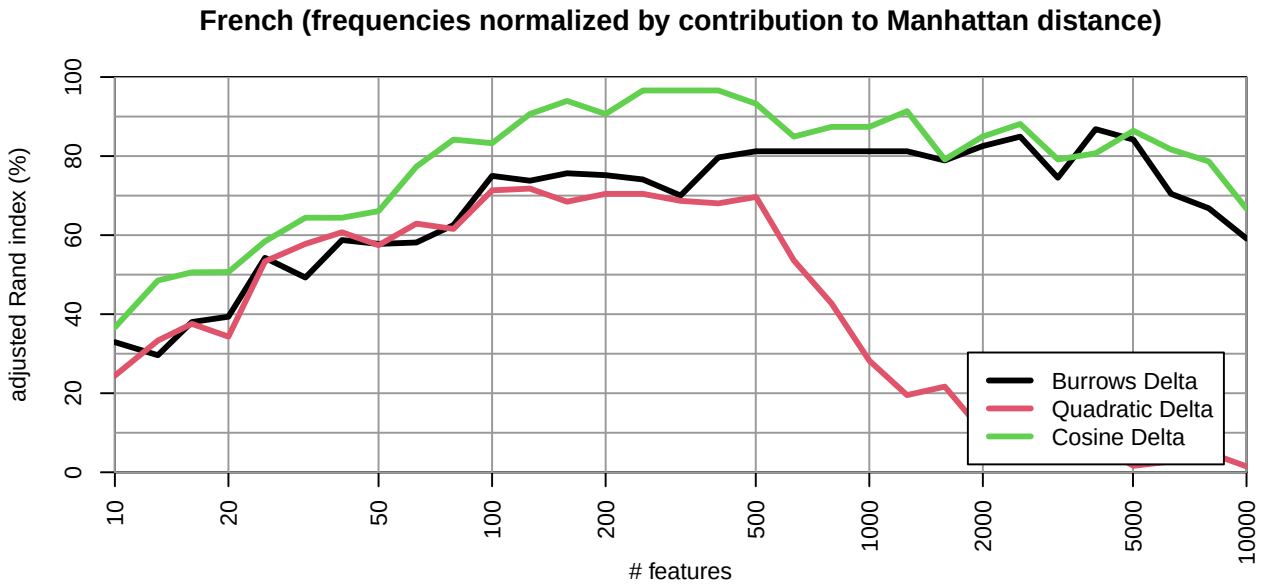



```
tmp <- scale.absdiff(zEN, dim=20000)
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
```

```
     main="English (frequencies normalized by contribution to Manhattan distance)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



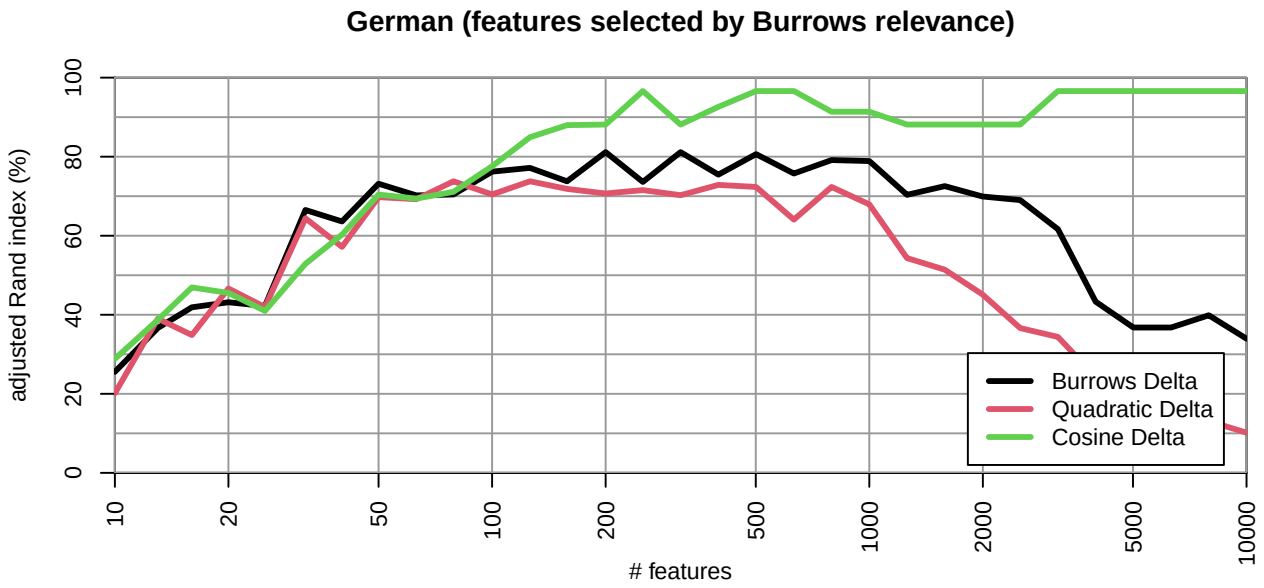

```
tmp <- scale.absdiff(zFR, dim=20000)
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="French (frequencies normalized by contribution to Manhattan distance)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



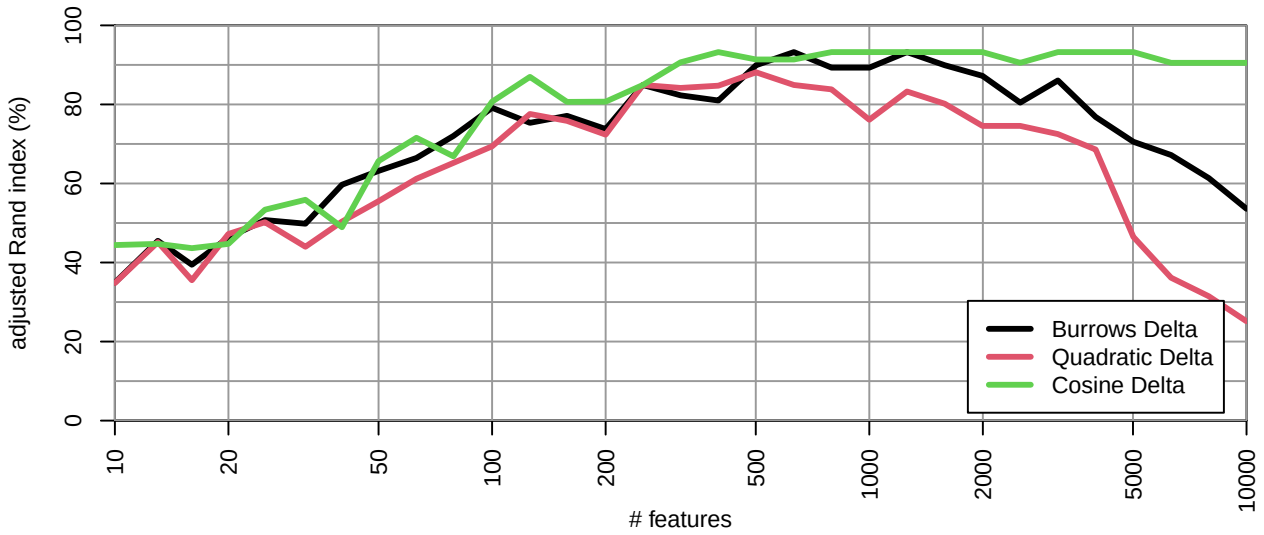
Results are worse than using standardization, even for Δ_B (but it is affected much less than Δ_Q and $\Delta_\angle$). This is actually quite plausible: there is more noise from lower-frequency words and words with a very sparse distribution than for standardized z-scores.

It seems reasonable to prefer z-score features with a large contribution to Δ_B, i.e. to use the value computed by `absdiff()` as a ranking criterion for feature selection. However, the graphs below show that this is less effective than using the most frequent words, especially for Δ_B and Δ_Q.

```
rel <- apply(zDE[, 1:50000], 2, absdiff) # consider 50,000 mfw only (for speed reasons)
tmp <- zDE[, order(rel, decreasing=TRUE)]
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="German (features selected by Burrows relevance)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```

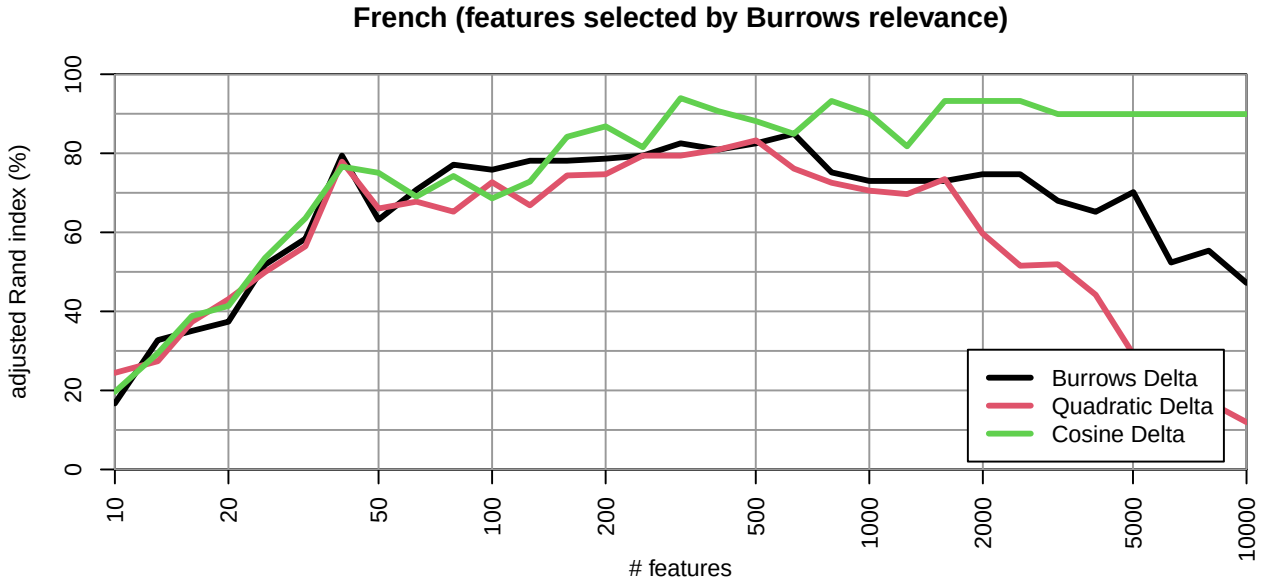



```
rel <- apply(zEN[, 1:50000], 2, absdiff)
tmp <- zEN[, order(rel, decreasing=TRUE)]
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="English (features selected by Burrows relevance)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldEN, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



The English data set is an exception: both Δ_B and Δ_Q improve in the range $200 \leq n_w \leq 3000$. Because of the small sample size, this observation – like many other findings – may well be a random fluke.

```
rel <- apply(zFR[, 1:50000], 2, absdiff) # consider 50,000 mfw only (for speed reasons)
tmp <- zFR[, order(rel, decreasing=TRUE)]
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="French (features selected by Burrows relevance)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldFR, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



2.5 Avoiding extremes: Truncated z-scores & quantile scores

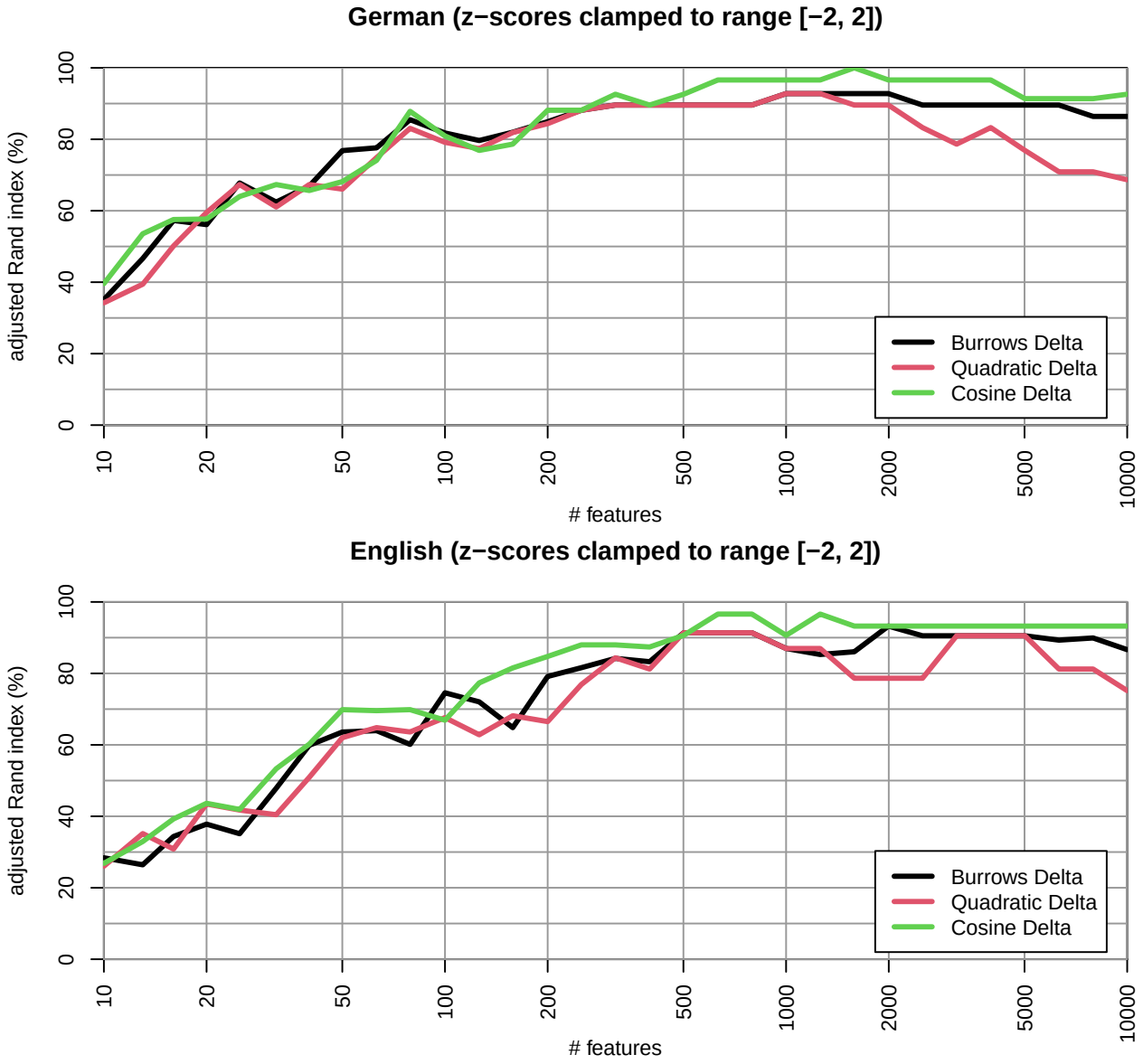
Experiments with normalization have led to two hypotheses on the causes of the observed performance improvements:

1. normalization reduces the influence of extreme z values for individual features in individual texts;
2. like the profile of a key, the relevant aspect of an author's signature is the pattern of positive and negative deviations from the norm, not the magnitude of these deviations.

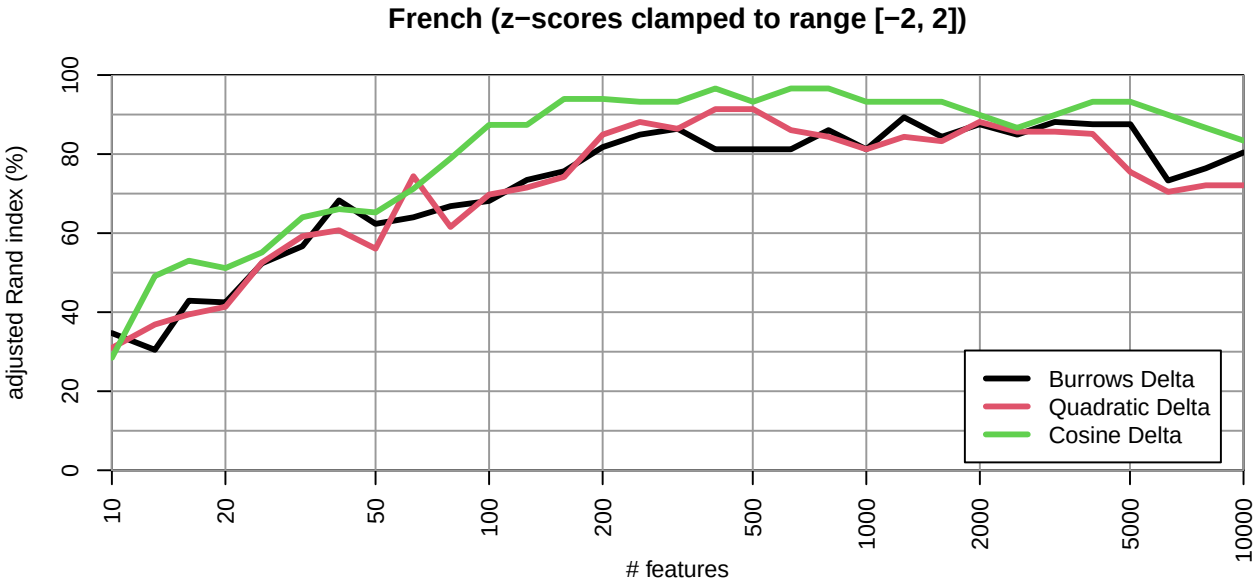
In order to test the first hypothesis, we truncate (or "clamp") the standardized frequencies to the range $[-2, 2]$, so that all features that deviate significantly ($p < .05$) from the average are treated equally and there are no longer any extreme z values. For the German and English data sets, this strategy is highly successful. Δ_B and Δ_Q become much more robust, though performance is still a little worse than with standardization.

```
tmp <- clamp(zDE, -2, 2)
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)",
     main="German (z-scores clamped to range [-2, 2])",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="manhattan")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="euclidean")$adj.rand, lwd=3, col=2)
lines(n.vals, evaluate(tmp, goldDE, n=n.vals, method="cosine")$adj.rand, lwd=3, col=3)
```

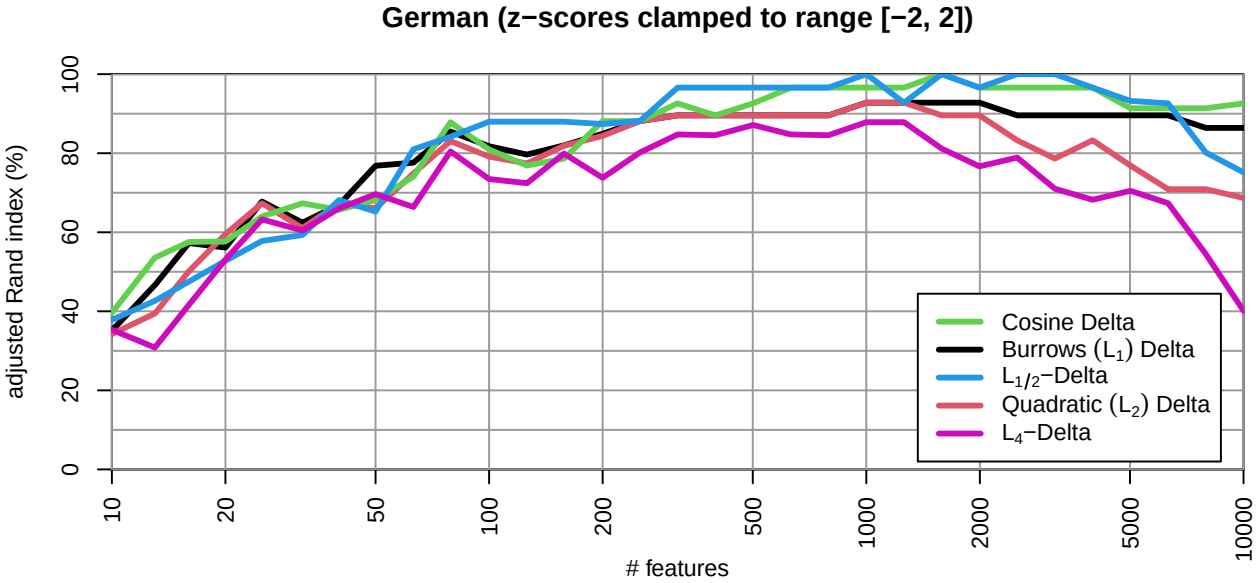
```
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:3,
       legend=c("Burrows Delta", "Quadratic Delta", "Cosine Delta"))
```



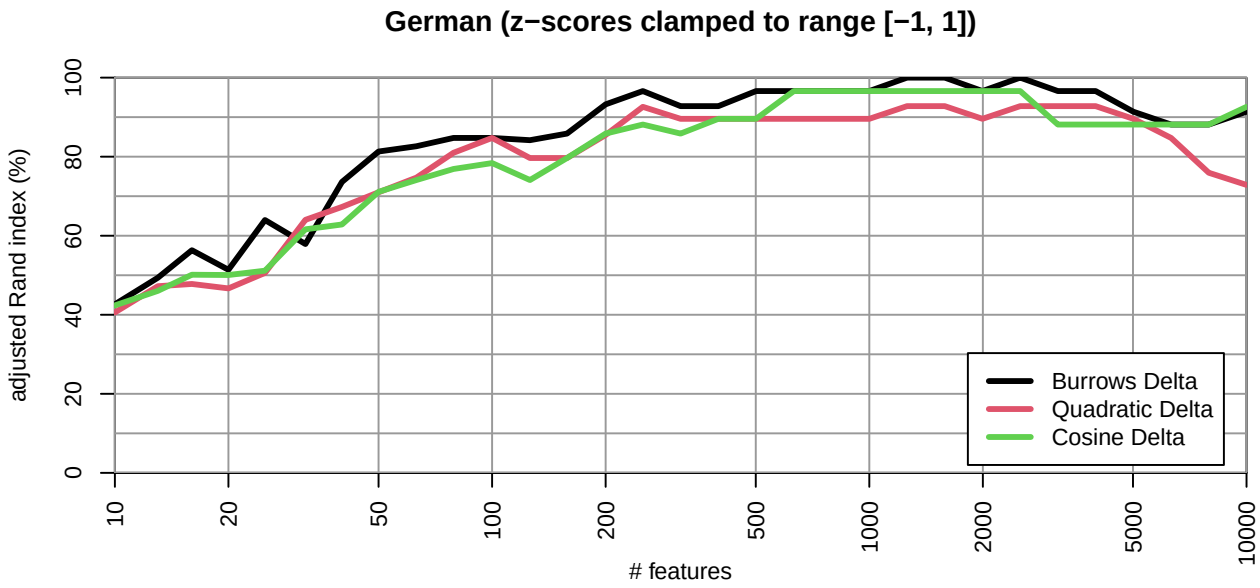
Truncating extreme values is less effective for the French data set, where it only leads to a relatively small improvement.



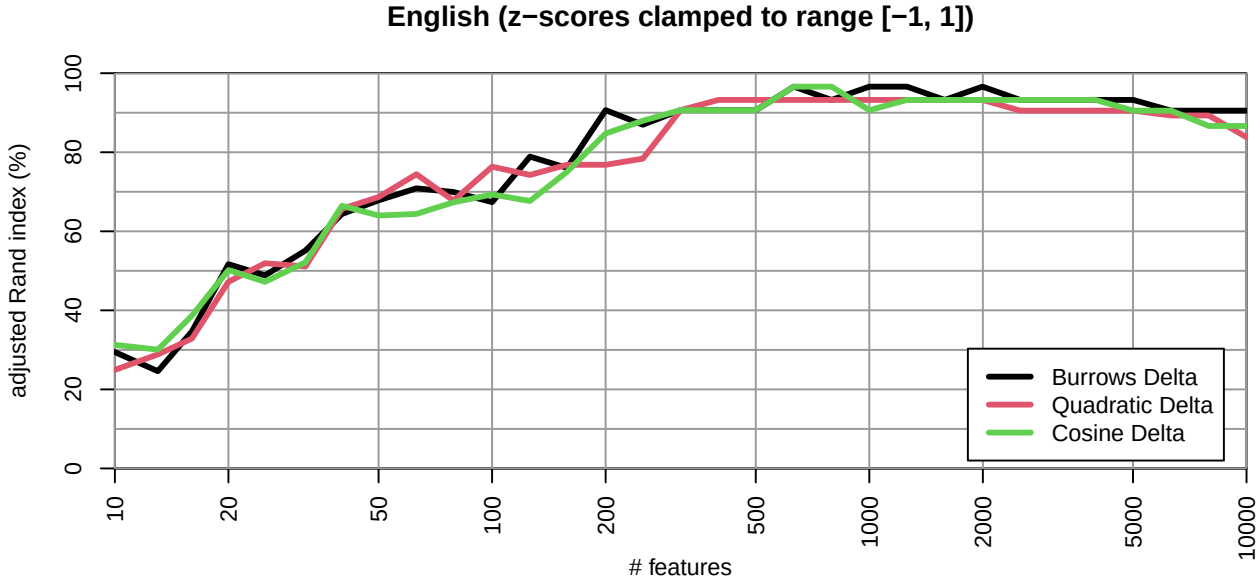
Comparison with Delta measures based on other p-norms confirms that clamping substantially improves robustness of distance metrics that are sensitive to outliers.



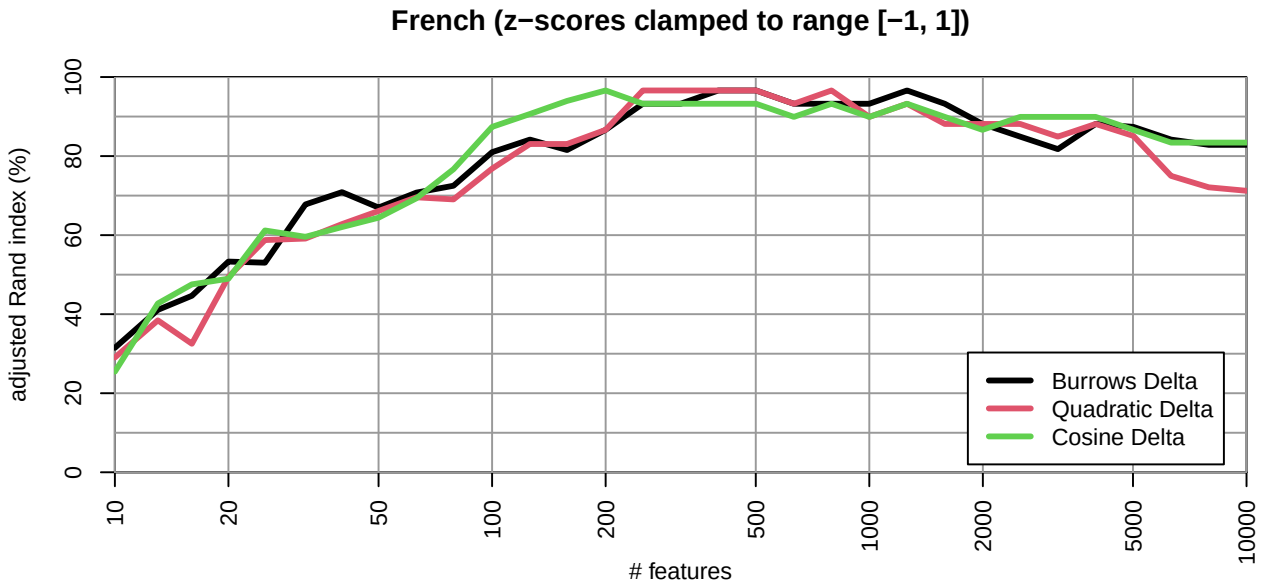
What happens if we truncate more aggressively to one standard deviation from the mean, i.e. the range $[-1, 1]$? This would only distinguish between small deviations in a central range and treat all substantial deviations equally.



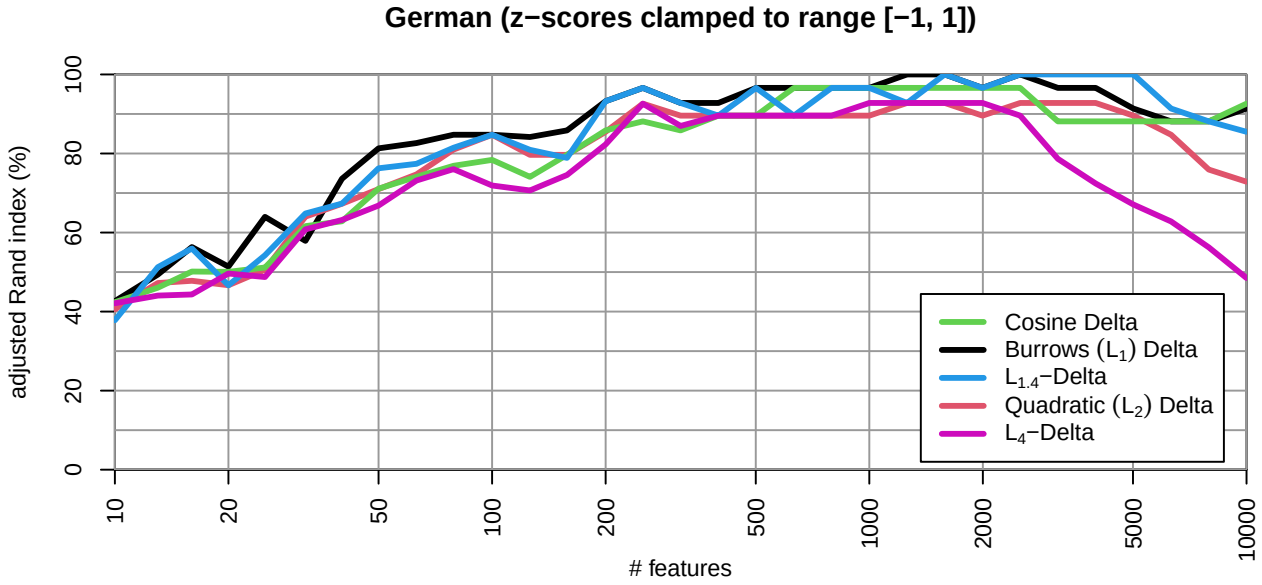
For German and English, the results are impressive: for $200 \leq n_w \leq 5000$, Δ_B is on par with or even better than Cosine Delta and other normalized measures. However, it seems less robust for $n_w > 5000$, and the performance of $\Delta_\angle$ suffers noticeably.



On the French data set, the aggressive truncation is not robust for $n_w \geq 2000$, but again yields good results up to this point.



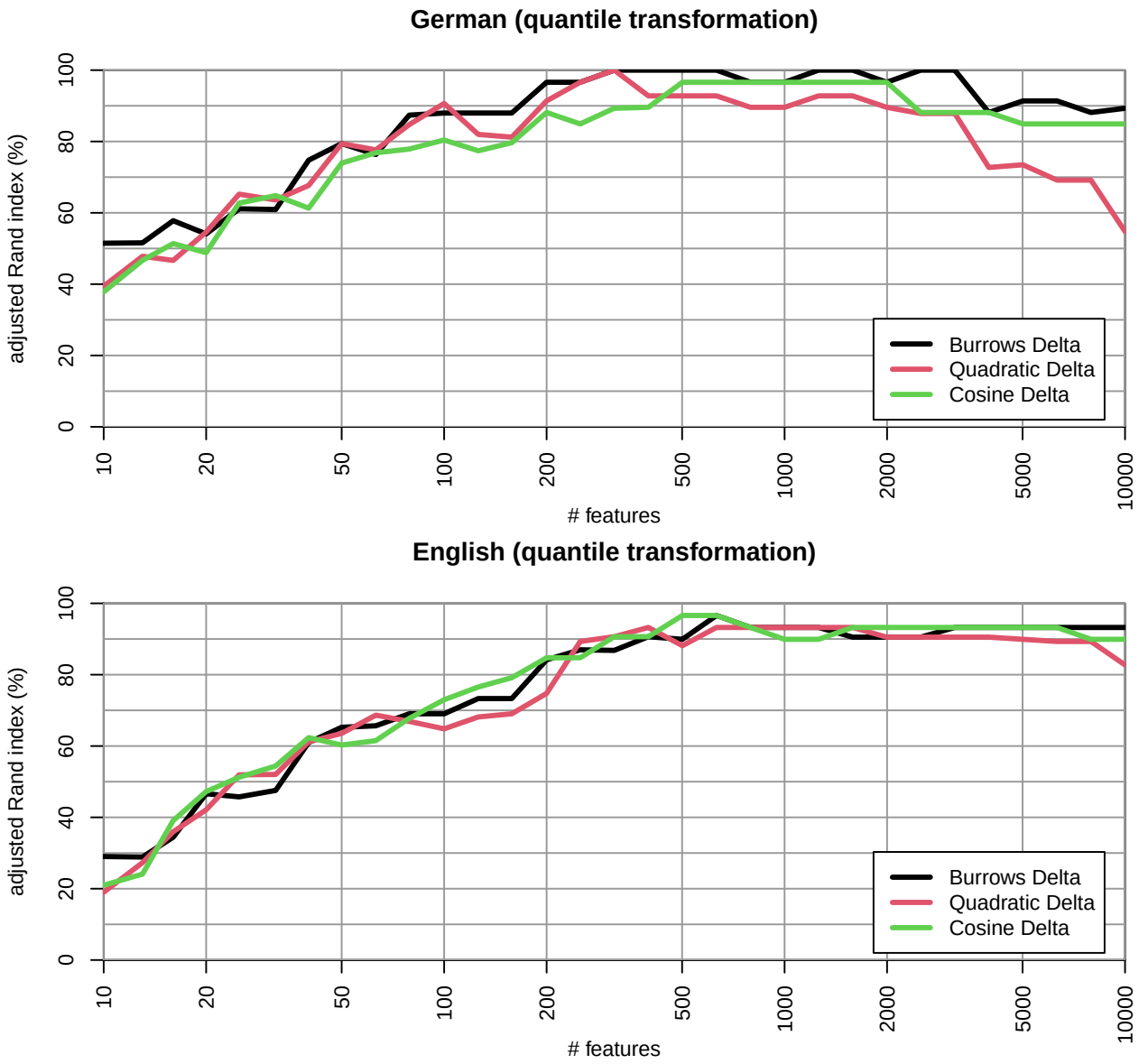
Again, results for other p-norms confirm improved robustness:

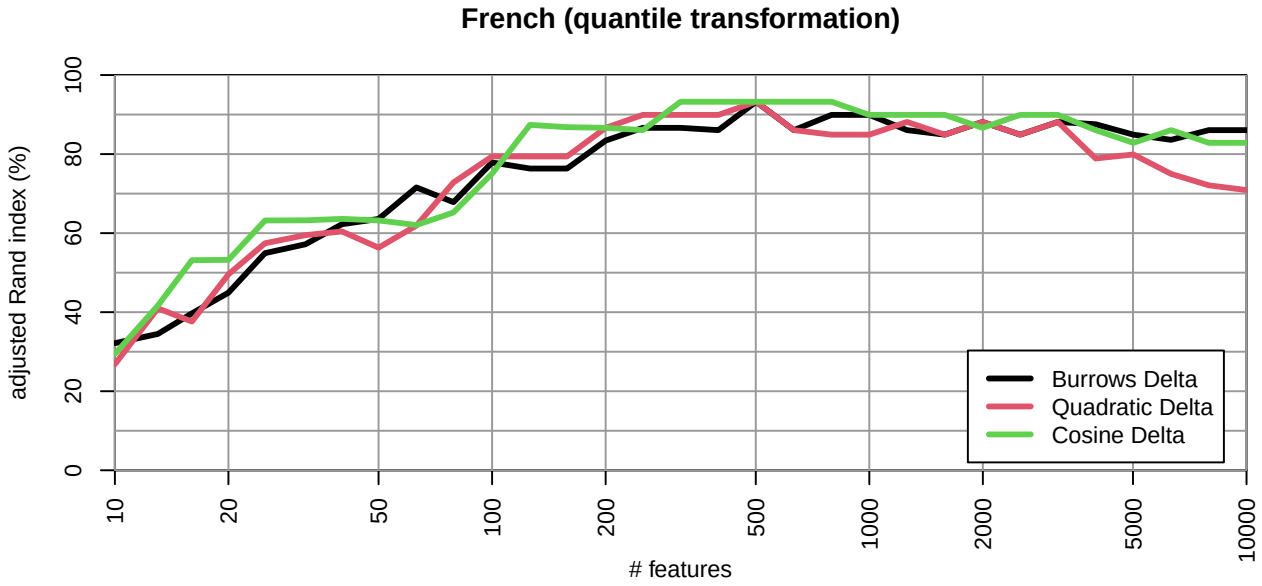


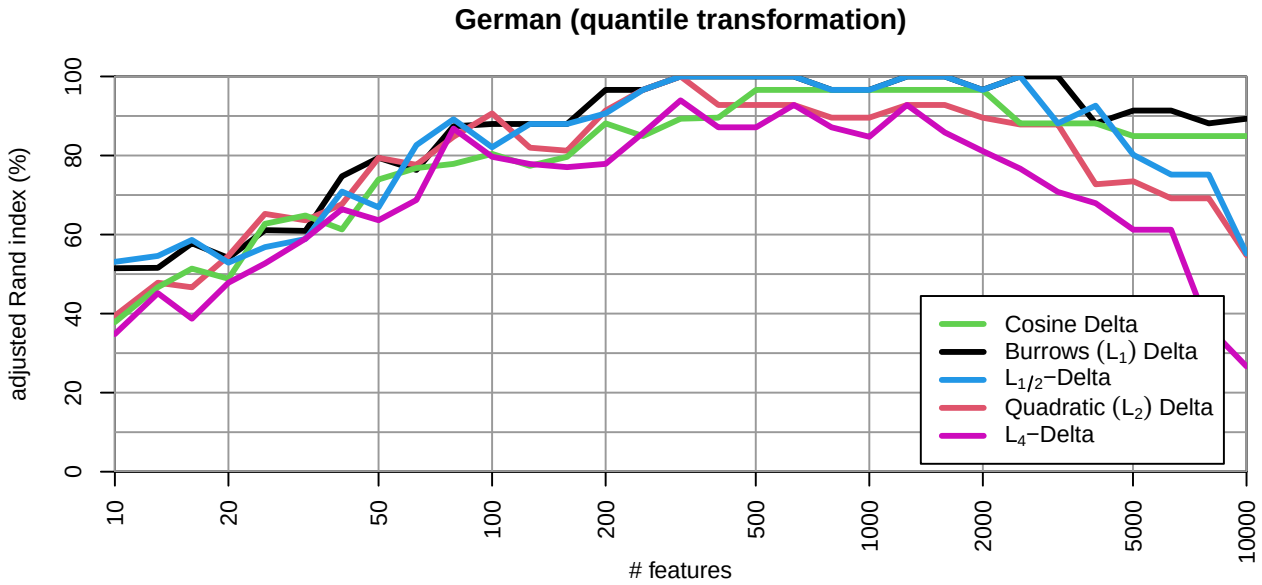
Another way of avoiding extreme score values is to transform the relative frequencies to **quantiles** instead of standardizing them. The function `quantile.score` either re-scales quantiles to the range $[-1, 1]$ (because $\Delta_\angle$ isn't translation-invariant and doesn't work well if the "neutral" value is different from 0) or transforms them into the corresponding z-scores of a standard normal distribution. Note that a quantile score of 0 now corresponds to the *median* rather than the mean of the relative frequencies.

```
quantile.score <- function(x, z.score=FALSE) {
  if (is.matrix(x)) {
    apply(x, 2, quantile.score, z.score=z.score) # apply to columns of matrix
  } else {
    n <- length(x)
    res <- (rank(x, ties="average") - 0.5) / n
    if (z.score) qnorm(res) else 2 * (res - 0.5)
  }
}
```

Results in all three languages are similar to aggressive clamping. For German, Δ_B achieves astonishing quality with perfect or near-perfect clustering over a fairly wide range of n_w. Back-transformation of the quantiles into z-scores – enforcing a normal distribution for each feature – works far less well.



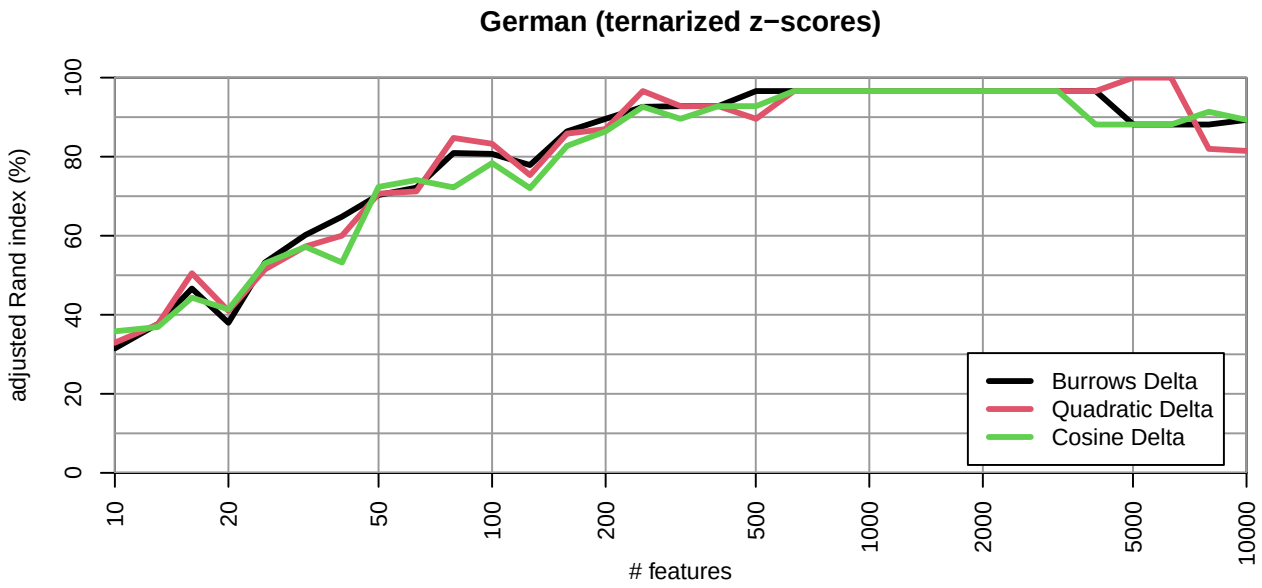


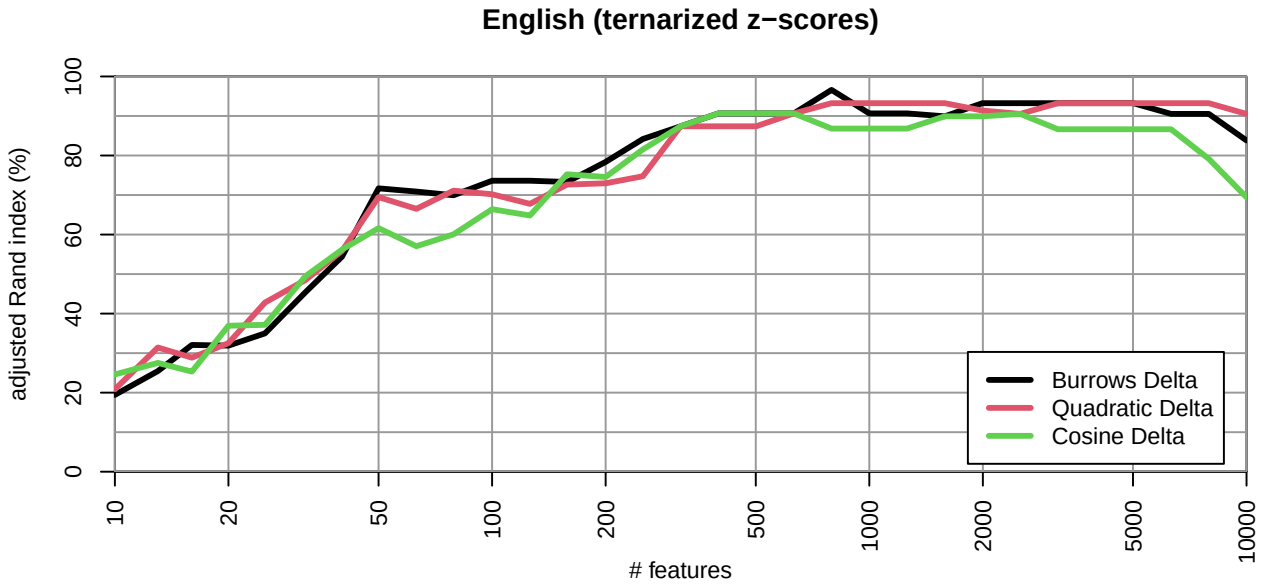


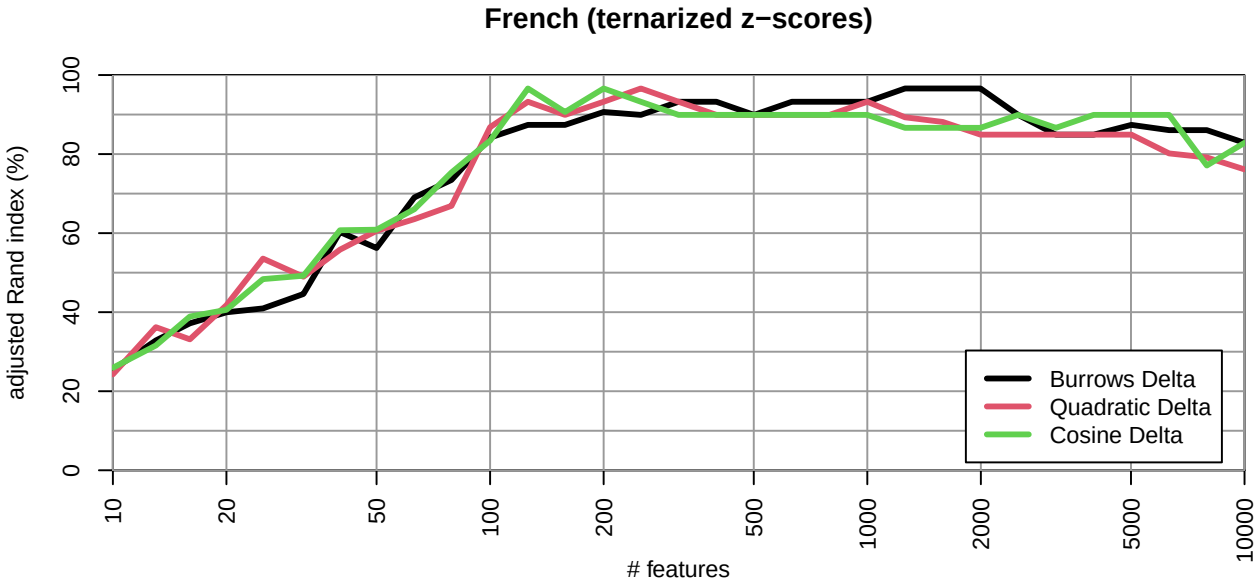
The good results obtained from aggressive clamping also provide some support for the second hypothesis (signatures as "key profiles"). We can test an extreme version of this hypothesis if we **binarize** the vectors, i.e. assign only values $+1$ (for above-average frequency) and -1 (for below-average frequency). A slightly more sophisticated approach interprets small positive and negative z-scores as neutral and assign the score 0 in order to introduce less noise; we refer to this as **ternarization**.

Here we show the results for **ternarized** z-scores, which assign the neutral value 0 if $-0.43 \leq z_i \leq 0.43$. If z_i followed a perfect standard normal distribution, this threshold would lead to equal proportions of -1, 0 and $+1$ in the ternarized vector.

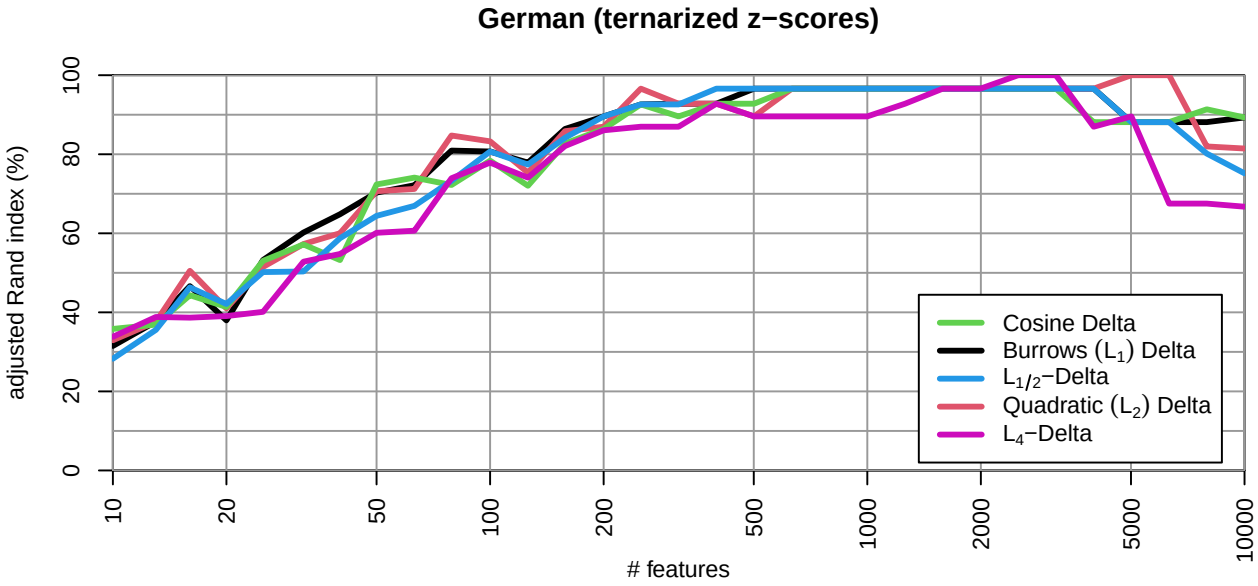
```
tmp <- ternarize(zDE, neutral.p=1/3)
```


German (ternarized z−scores)
adjusted Rand index (%)
0
20
40
60
80
100
10
20
50
100
200
500
1000
2000
5000
10000
features
Burrows Delta
Quadratic Delta
Cosine Delta


English (ternarized z−scores)
adjusted Rand index (%)
0
20
40
60
80
100
10
20
50
100
200
500
1000
2000
5000
10000
features
Burrows Delta
Quadratic Delta
Cosine Delta



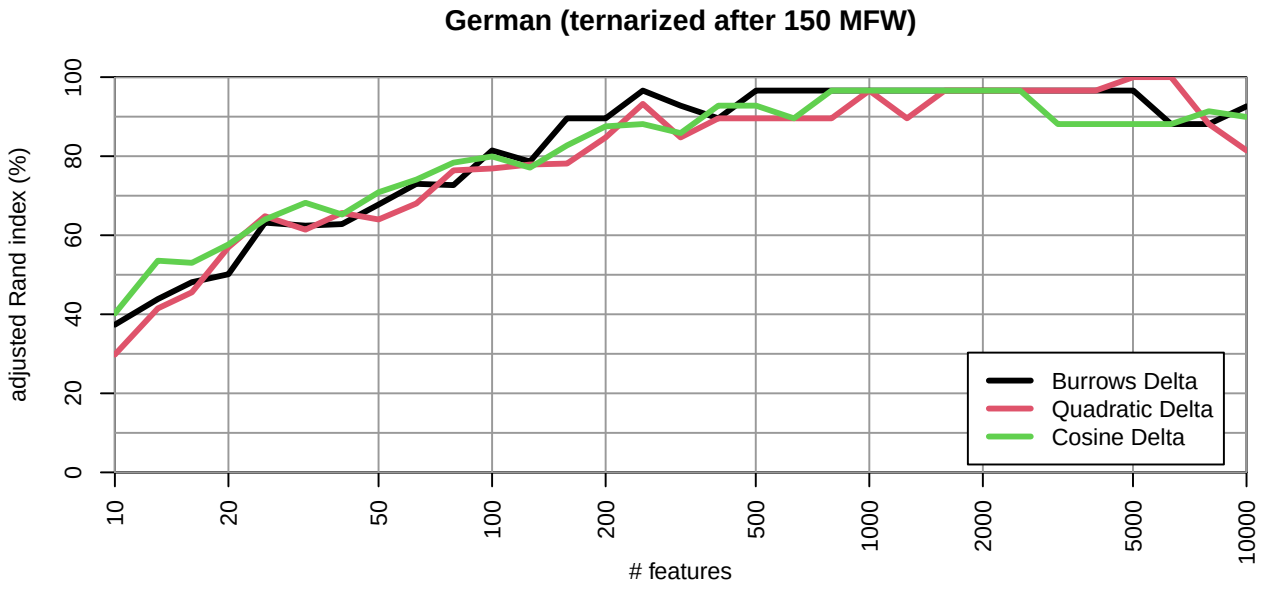
The ternarization leads to astonishingly robust performance on the German data set, but results for English and French are also fairly good. Differences between the different Δ measures are essentially neutralized, which is confirmed for other p-norms:

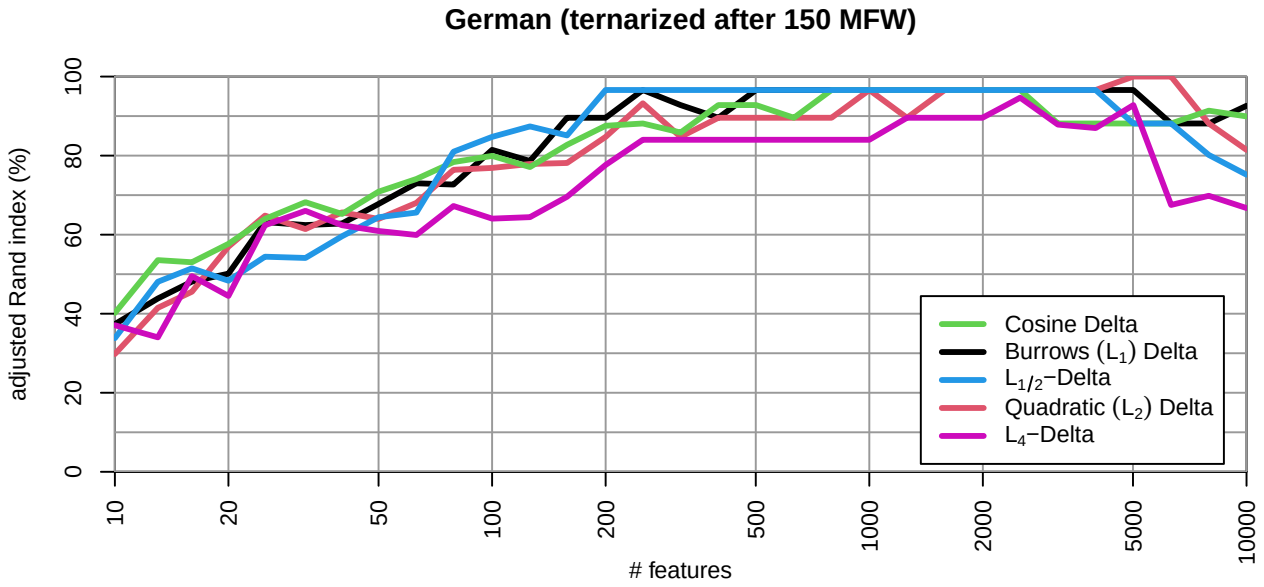


The lower clustering quality for $n_w \leq 200$ is not surprising. These mfw – mostly function words – are present in a majority of the texts; differences between authors will be reflected on a much more fine-grained scale than just overuse vs. underuse. More specialized content words ($n_w > 500$) occur only in few texts and here the pattern of which words occur at all provides more useful information than the word frequencies (which can produce extreme outliers in individual texts). Infrequent words ($n_w > 2000$) have increasingly sparse distributions and provide essentially a *binary* rathern than ternary signal: they are either present in a text or not.

These last results suggest that a suitable adaptive transformation – blending from moderate truncation for the first mfw to binarization of low-frequency freatures – might bring further improvements. A simple approach is a linear blend from unmodified z-scores to ternarization over the first 150 mfw, giving slightly better results for very short feature vectors, as may be expected. Suprisingly, it also seems to lower the robustness of the clustering for $n_w > 1000$ mfw.

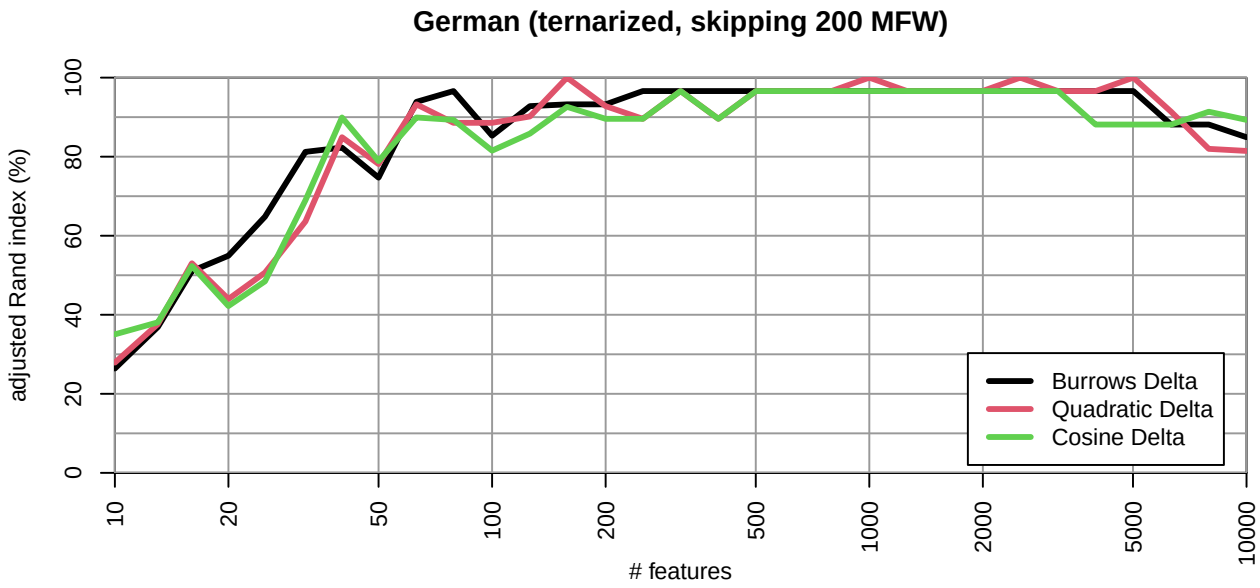
```
tmp <- ternarize(zDE, neutral.p=1/3, crossover=150)
```

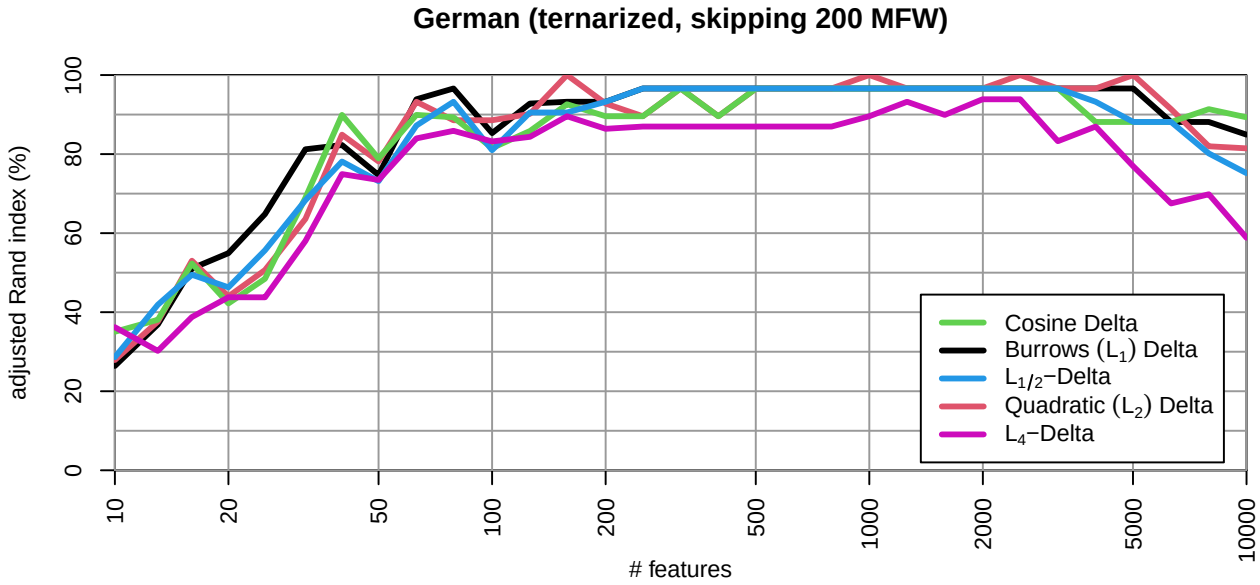




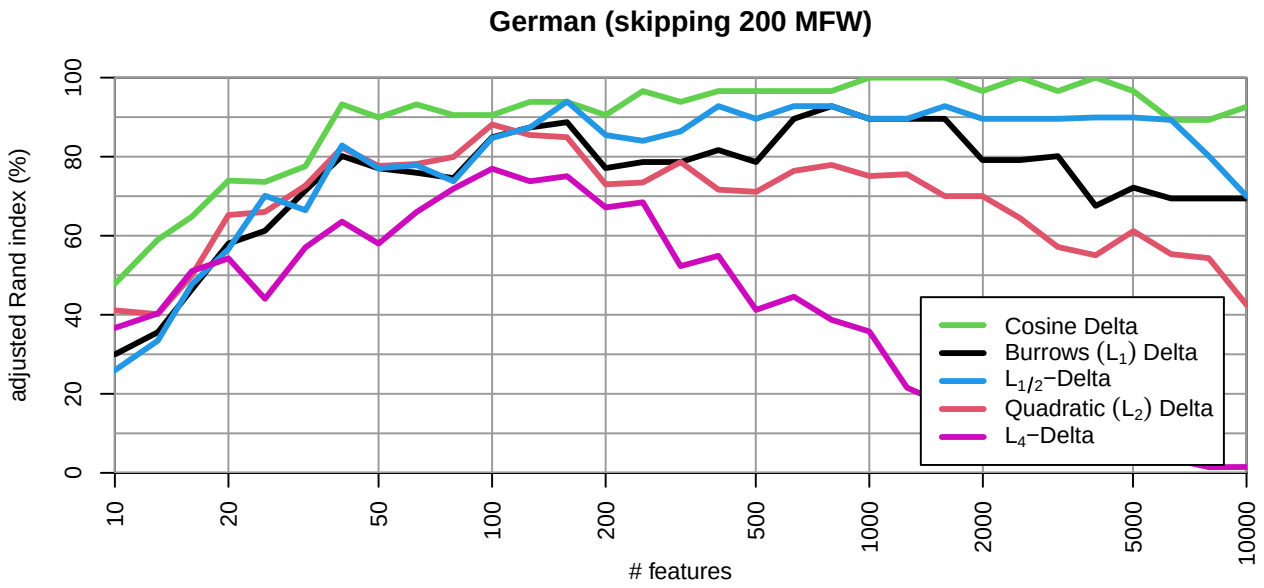
Interestingly, skipping the first 200 mfw entirely appears to give even better results for $n_w > 50$ combine with better robustness (except for L_4-Delta). This surprising observation will have to be confirmed for the other languages and should be investigated systematically (using different combination of n_w and n_{skip}).

```
tmp <- ternarize(zDE, neutral.p=1/3)[, -(1:200)]
```



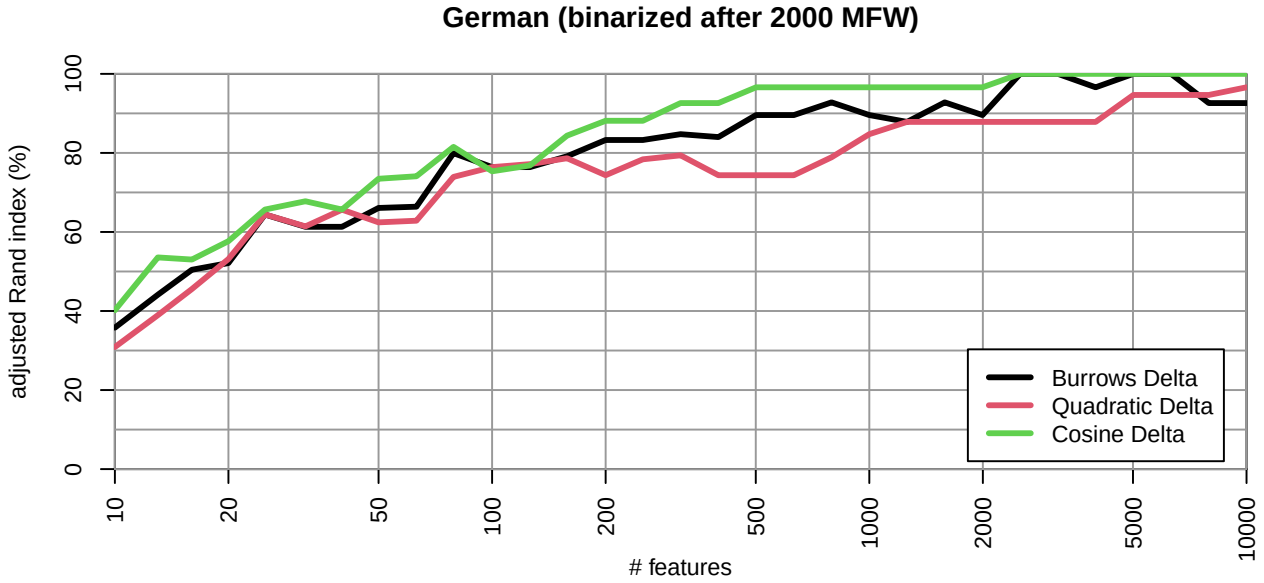


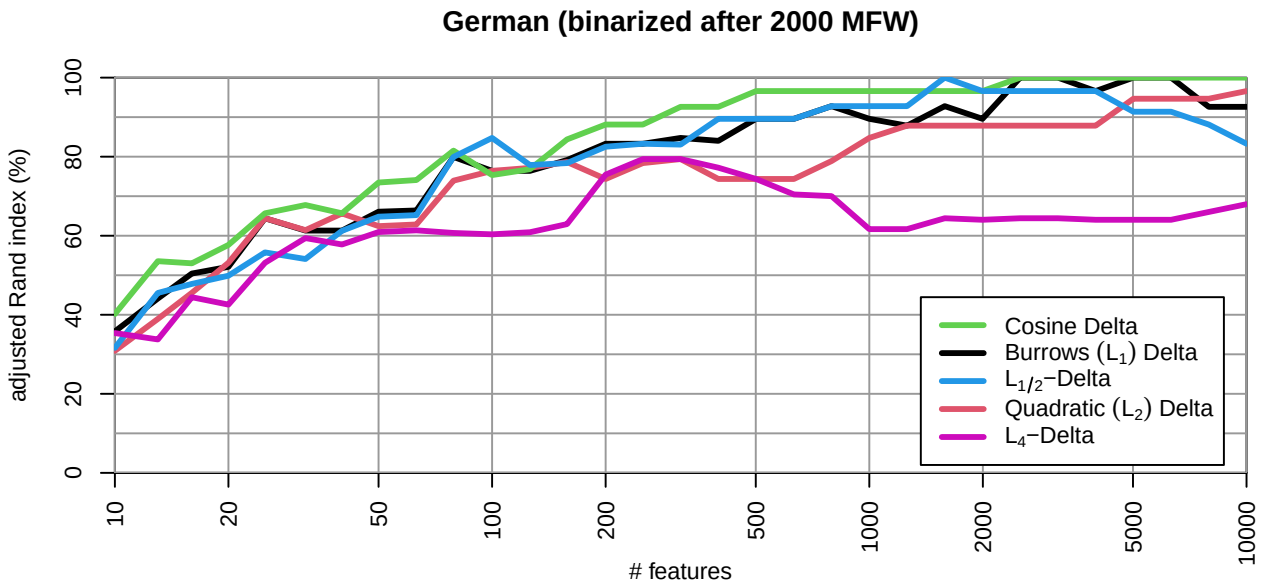
Merely skipping the first 200 mfw *without* quantization is even better for $\Delta_\angle$, but performs rather poorly for the other distance measures.



Binarization doesn't work as well for improving robustness of the measures (because it is most sensibly applied for very low-frequency features $n_w > 2000$). With a slow cross-over from z-scores to binarized features, very long feature vectors become stable, though.

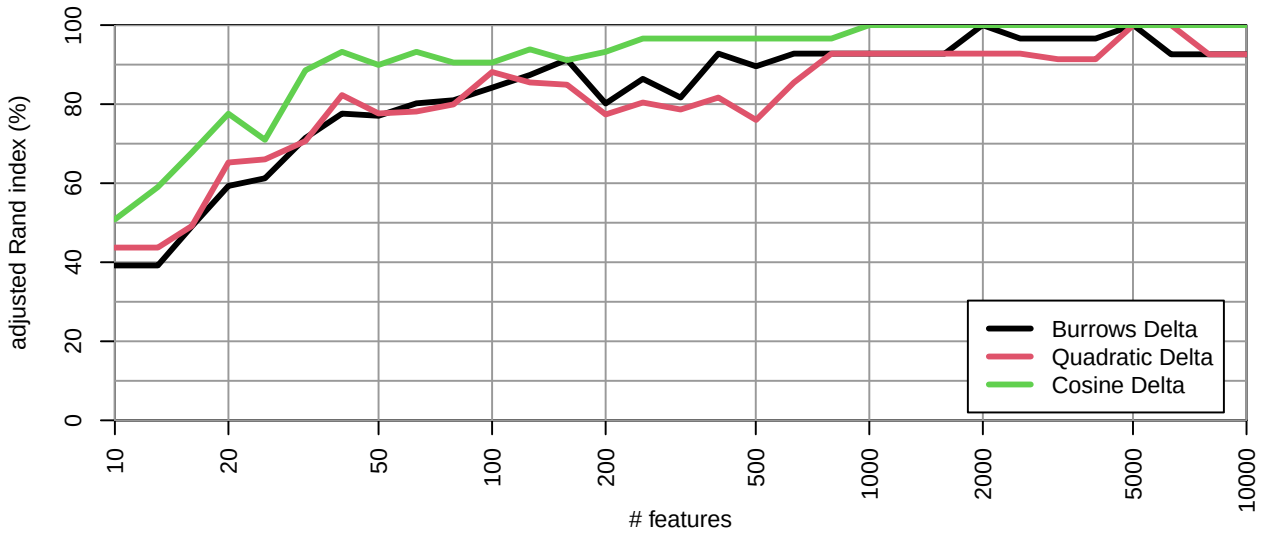
```
tmp <- binarize(zDE, crossover=2000)
```

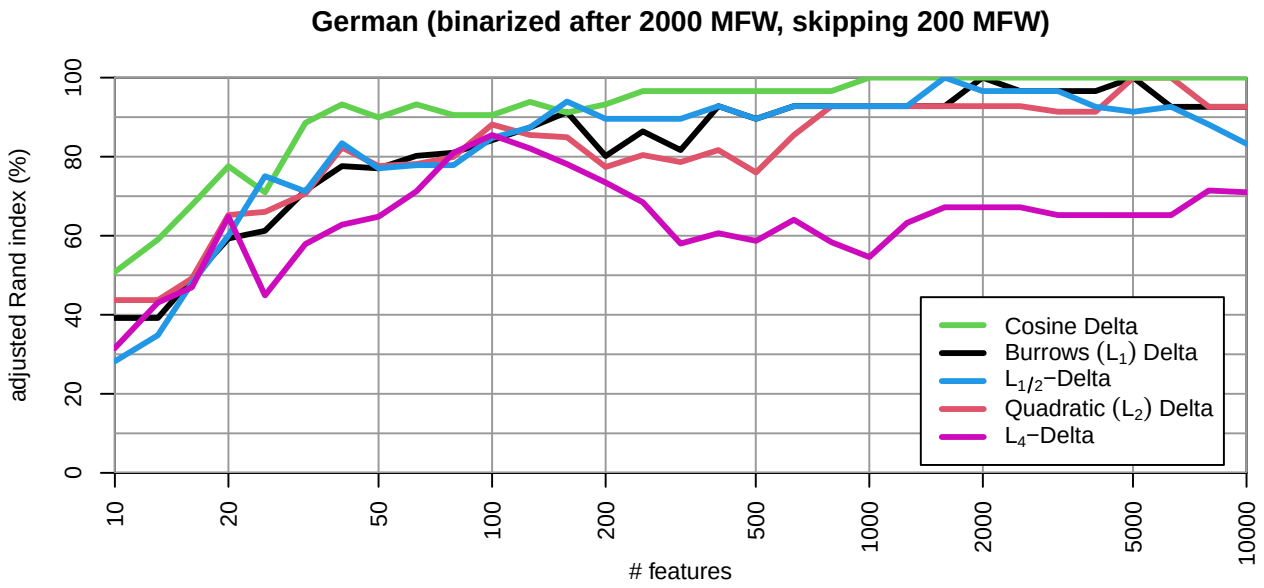




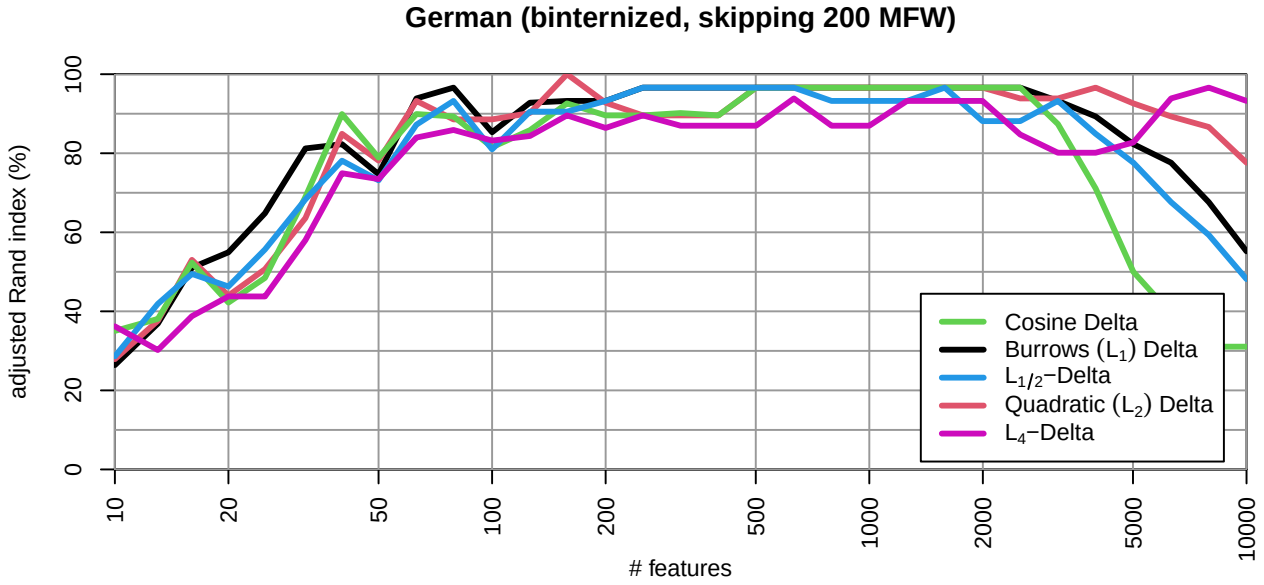
Again, skipping the first 200 mfw (here combined with the cross-over) gives even better clustering accuracy:

```
tmp <- binarize(zDE, crossover=2000)[, -(1:200)]
```





However, a theoretically appealing combination of ternarization (frequent words) and binarization (lower-frequency words where only presence and absence should be distinguished) doesn't show robust performance. Confusingly, it even appears to work better with L_2- and L_4-Delta than with normally robust measures. Further investigation will be needed in order to explain why the **average relative frequency** μ_i plays a crucial role even though it wouldn't seem to be suitable for the highly skewed and sparse distributions of lower-frequency words. One possible reason are **different text lengths** and hence different probabilities of observing a single occurrence of a word by chance – μ_i could then be seen as an approximation of the expected frequency of w_i in a given text.

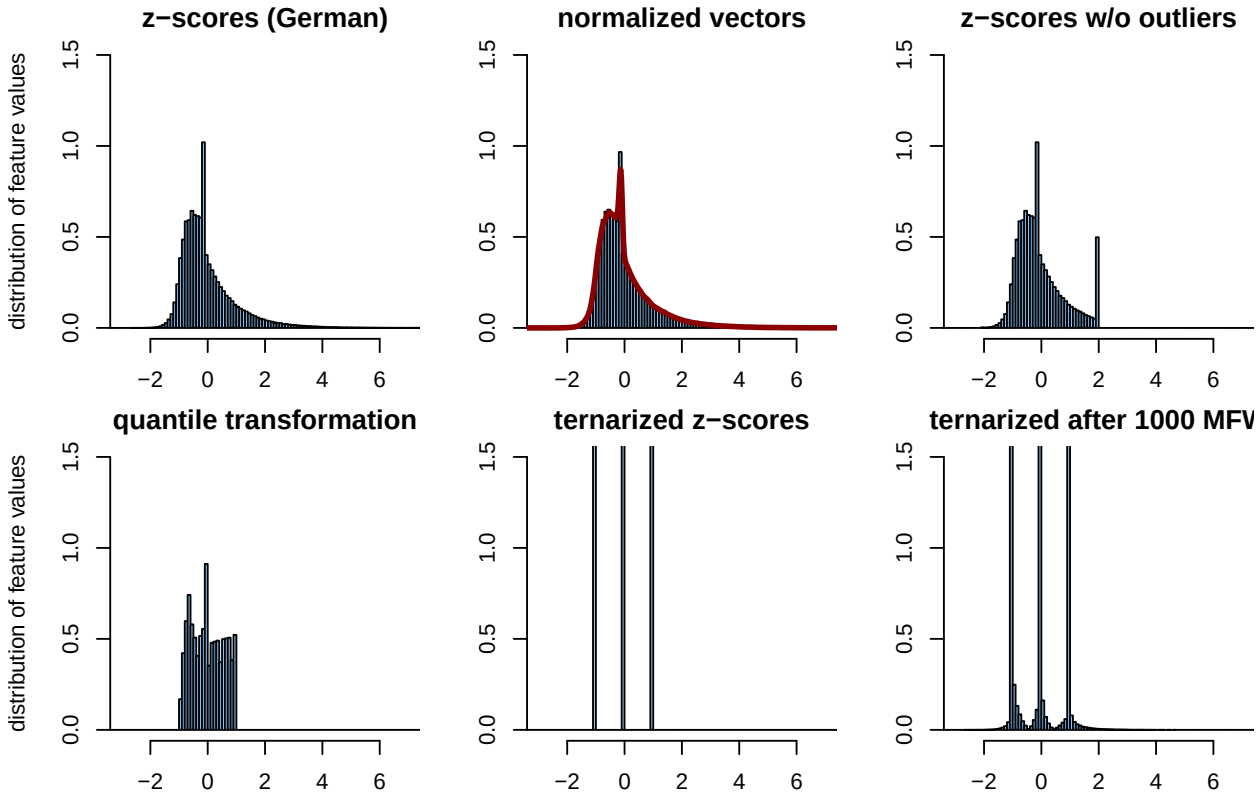


2.6 Looking for causes: outliers or profiles?

How much do the various adjustments help to reduce outlier values? Do approaches that cut off outliers perform better than other adjustments? The plot below shows histogram of feature scores for $n_w = 5,000$ mfw pooled across all 75 texts; z-scores with cutoff are clamped to the range $-2 \leq z \leq 2$.

```
x.brk <- seq(-4, 12, .1)
z <- zDE[, 1:5000]
```

```
z.norm <- scale(as.vector(normalize.rows(z))) # rescale normalized vectors
z.clamp <- clamp(z, -2, 2)
z.tern <- ternarize(z, neutral.p=1/3)
z.mix <- ternarize(z, neutral.p=1/3, crossover=1000)
z.quant <- quantile.score(z)
par(mfrow=c(2,3), mar=c(2,4,2,0), cex=1)
hist(z, breaks=x.brk, freq=FALSE, col="#88CCFF", xlim=c(-3,7), ylim=c(0, 1.5),
     main="z-scores (German)", xlab="", ylab="distribution of feature values")
hist(z.norm, breaks=x.brk, freq=FALSE, col="#88CCFF", xlim=c(-3,7), ylim=c(0, 1.5),
     main="normalized vectors", xlab="", ylab="")
lines(density(z), col="darkred", lwd=3)
hist(z.clamp, breaks=x.brk, freq=FALSE, col="#88CCFF", xlim=c(-3,7), ylim=c(0, 1.5),
     main="z-scores w/o outliers", xlab="", ylab="")
hist(z.quant, breaks=x.brk, freq=FALSE, col="#88CCFF", xlim=c(-3,7), ylim=c(0, 1.5),
     main="quantile transformation", xlab="", ylab="distribution of feature values")
hist(z.tern, breaks=x.brk, freq=FALSE, col="#88CCFF", xlim=c(-3,7), ylim=c(0, 1.5),
     main="ternarized z-scores", xlab="", ylab="")
hist(z.mix, breaks=x.brk, freq=FALSE, col="#88CCFF", xlim=c(-3,7), ylim=c(0, 1.5),
     main="ternarized after 1000 MFW", xlab="", ylab="")
```




```
par(mfrow=c(1,1))
```

The top left panel shows that the overall distribution of z-scores has a fairly typical bell-curve shape, even though the first 5,000 mfw already include many sparsely distributed content words. The distribution is moderately skewed to the right with a few substantial outliers ($z > 8$). The only conspicuous feature is a spike at or near $z = 0$, presumably from matrix cells with $f_i = 0$.

When z-scores are clamped to the range $[-2, 2]$ in the top right panel, outliers with $|z| > 2$ are effectively removed from the data set. Note that this only applies to features with above-average frequency $z > 2$, as

there do not seem to be large negative values $z < -2$ in the data set. Clustering quality is improved to some extent by this operation, but much less than by normalization or when a more aggressive threshold (e.g. $|z| > 1$) is used. In the most extreme case, z-scores can be ternarized (bottom right panel).

The bottom left panel shows the effect of vector normalization on the feature distribution (where feature values have been rescaled after normalization to be on the same scale as the other histograms). The red line indicates the distribution of z-scores before normalization, showing that normalization has only a minimal effect: it neither corrects the skew of the distribution nor does it remove outlier values. Nonetheless, normalized vectors produce excellent and robust clustering quality.

This leads to the conclusion that it is not the sensitivity of Δ_Q towards outlier values which accounts for its poor performance and robustness. Especially the ternarized features suggest an interpretation in terms of a profile of positive and negative deviations, where the degree of deviation – i.e. how much an author's fingerprint is expressed in a given text – is a confounding effect.

References

Argamon, Shlomo. 2008. "Interpreting Burrows's Delta: Geometric and Probabilistic Foundations." *Literary and Linguistic Computing* 23 (2): 131–47. https://doi.org/10.1093/llc/fqn003.

Burrows, John. 2002. "'Delta': A Measure of Stylistic Difference and a Guide to Likely Authorship." *Literary and Linguistic Computing* 17 (3): 267–87. https://doi.org/10.1093/llc/17.3.267.

Jannidis, Fotis, Steffen Pielström, Christof Schöch, and Thorsten Vitt. 2015. "Improving Burrows' Delta. An Empirical Evaluation of Text Distance Measures." In *Proceedings of the Digital Humanities Conference 2015*. Sydney, Australia.

Shapiro, S. S., and M. B. Wilk. 1965. "An Analysis of Variance Test for Normality (Complete Samples)." *Biometrika* 52 (3/4): 591–611.

Shapiro, S. S., M. B. Wilk, and H. J. Chen. 1968. "A Comparative Study of Various Tests for Normality." *Journal of the American Statistical Association* 63: 1343–72.