Understanding Burrows Delta and its variants

Stefan Evert

16 January 2015

Some preliminaries

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`

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	accuracy	adj.rand
DE | Burrows D | n=100	100	93.33	76.41
DE | Burrows D | n=1000	1000	98.67	87.85
DE | Burrows D | n=5000	5000	96.00	79.14
EN | Burrows D | n=100	100	77.33	53.08
EN | Burrows D | n=1000	1000	94.67	82.20
EN | Burrows D | n=5000	5000	94.67	81.22
FR | Burrows D | n=100	100	86.67	71.10
FR | Burrows D | n=1000	1000	93.33	81.22
FR | Burrows D | n=5000	5000	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	accuracy	adj.rand
DE | Quadratic D | n=100	100	93.33	76.41
DE | Quadratic D | n=1000	1000	94.67	78.65
DE | Quadratic D | n=5000	5000	85.33	64.07
EN | Quadratic D | n=100	100	78.67	46.24
EN | Quadratic D | n=1000	1000	89.33	66.79
EN | Quadratic D | n=5000	5000	89.33	47.44
FR | Quadratic D | n=100	100	89.33	78.64
FR | Quadratic D | n=1000	1000	93.33	80.46
FR | Quadratic D | n=5000	5000	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	accuracy	adj.rand
DE | Cosine D | n=100	100	89.33	75.35
DE | Cosine D | n=1000	1000	98.67	96.60
DE | Cosine D | n=5000	5000	96.00	96.60
EN | Cosine D | n=100	100	84.00	64.41
EN | Cosine D | n=1000	1000	98.67	96.60
EN | Cosine D | n=5000	5000	94.67	93.24
FR | Cosine D | n=100	100	93.33	82.52
FR | Cosine D | n=1000	1000	97.33	93.24
FR | Cosine D | n=5000	5000	93.33	93.24

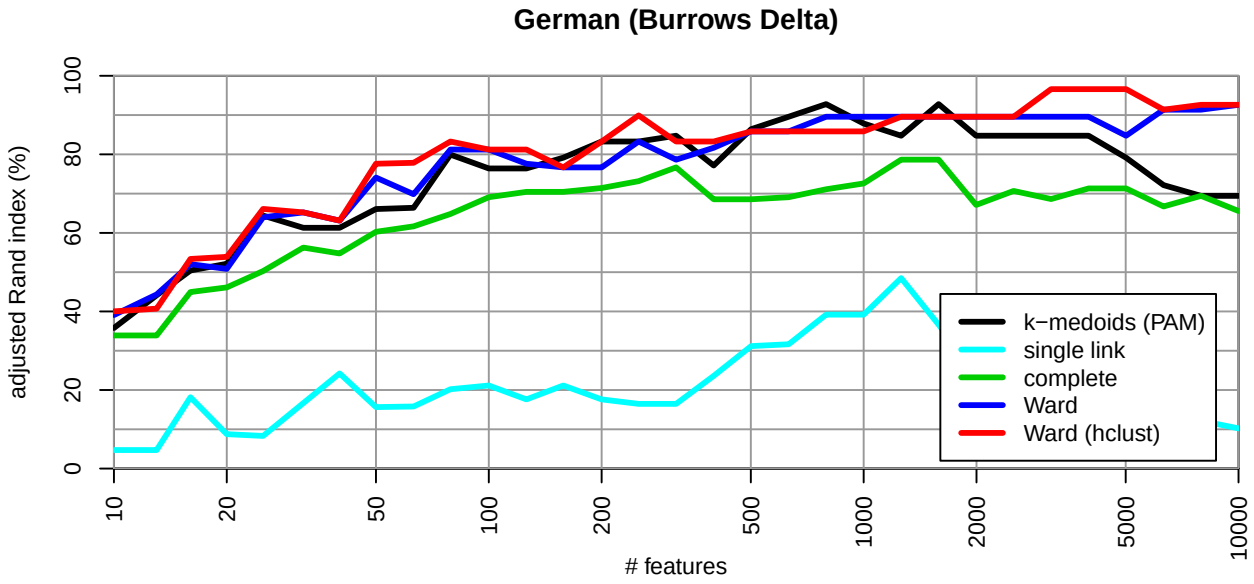
Setup for plots

```
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")
}
```

Clustering Methods

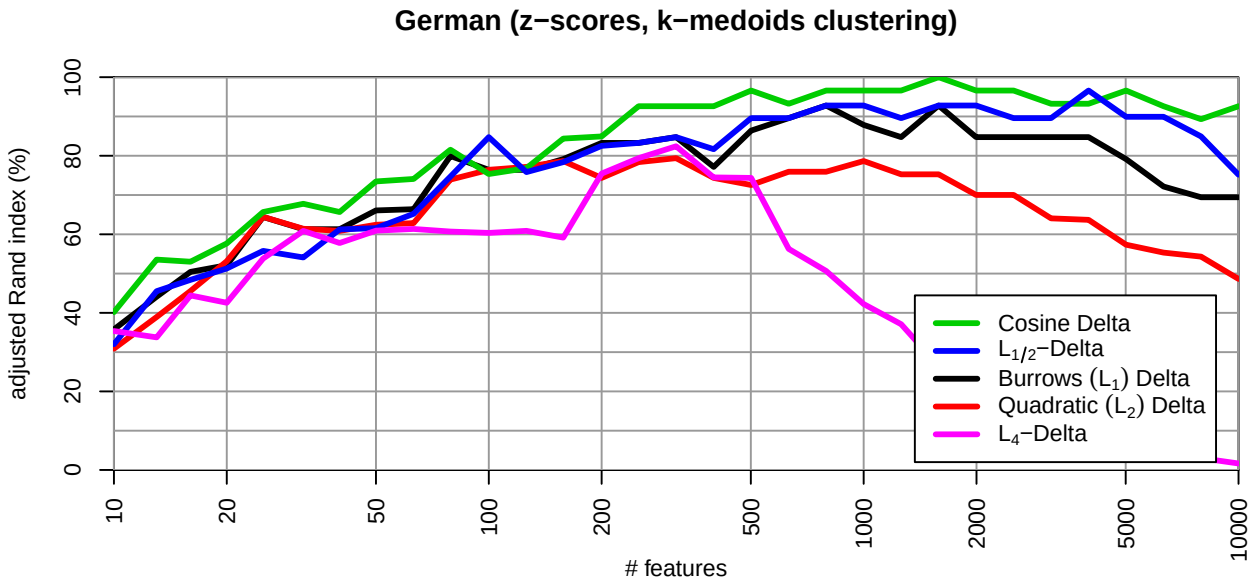
Apart from the Delta distance measure itself, the clustering algorithm seems to have a substantial impact on classification accuracy / ARI. Let us start with a comparison of different clustering methods for Burrows Delta Δ_B:

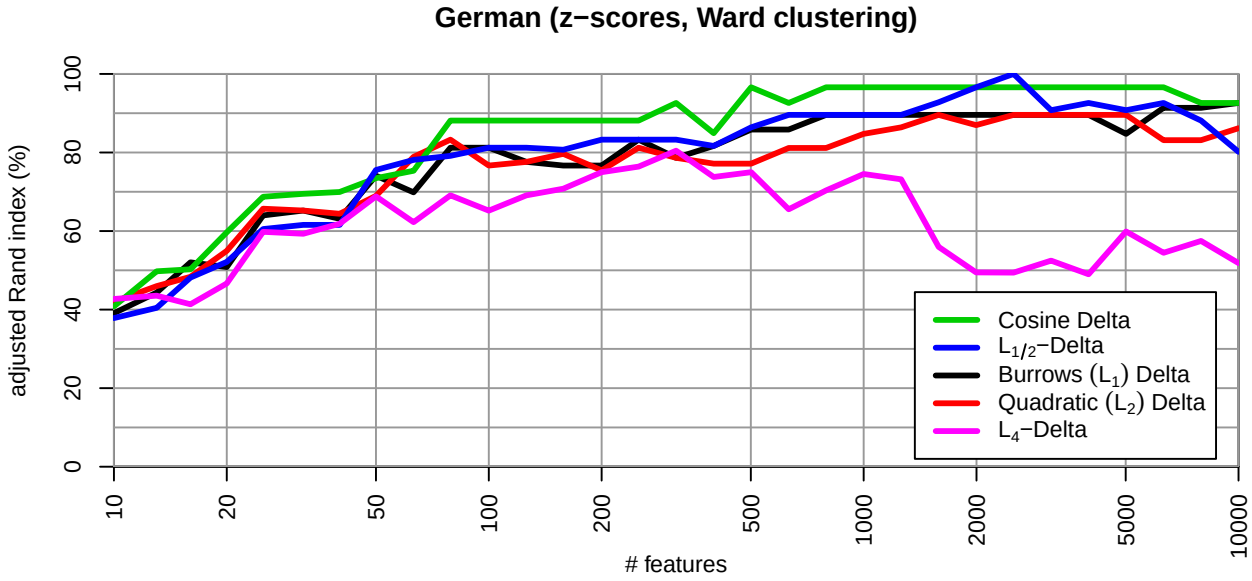
```
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)", main="German (Burrows Delta)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="pam")$adj.rand, lwd=3, col=1)
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="single")$adj.rand, lwd=3, col=5)
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="complete")$adj.rand, lwd=3, col=3)
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="ward")$adj.rand, lwd=3, col=4)
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="hclust")$adj.rand, lwd=3, col=2)
legend("bottomright", inset=.02, bg="white", lwd=3, col=c(1,5,3,4,2),
       legend=c("k-medoids (PAM)", "single link", "complete", "Ward", "Ward (hclust)"))
```



The enormous influence of clustering methods on attribution quality suggests a more complex interplay between the structure of inter-text distances induced by some Delta measure and the clustering algorithm. It would be too simplistic to equate good performance in the PAM-based evaluation with a better measure of stylistic similarity. Many further experiments seem to be required in order to understand why and how clusters are formed from text distances.

This observation also puts the amazing effect of vector normalization in a new light: Ward clustering seems to be fairly robust w/o normalization, even for Δ_Q (cf. two plots below). Perhaps the normalization effect we're trying to understand merely adjusts for a weakness of certain clustering algorithms?

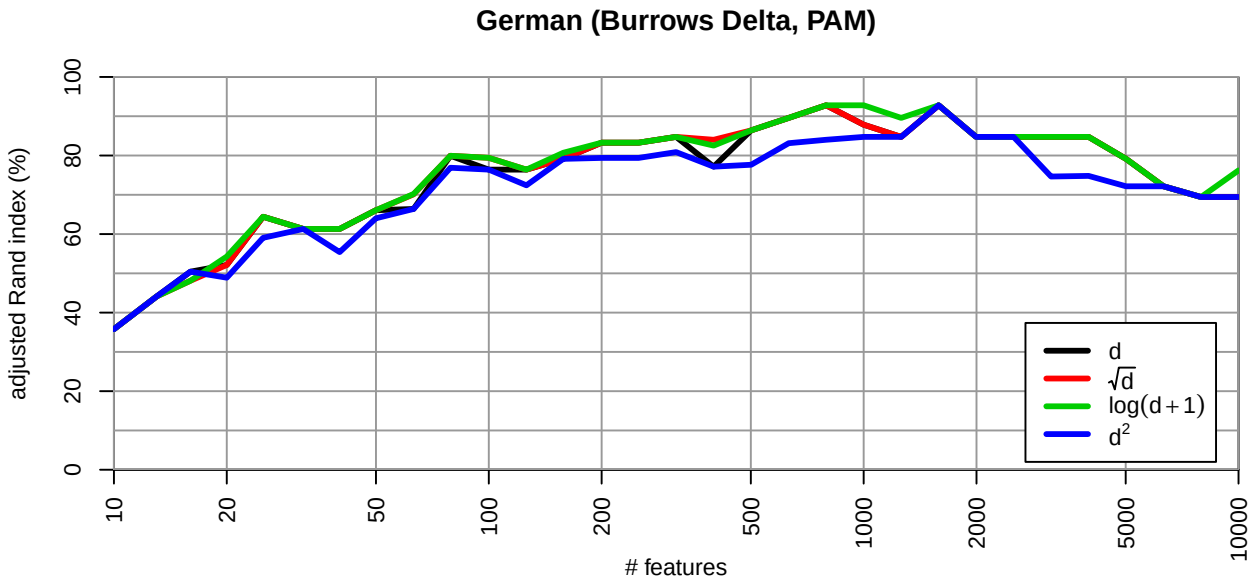




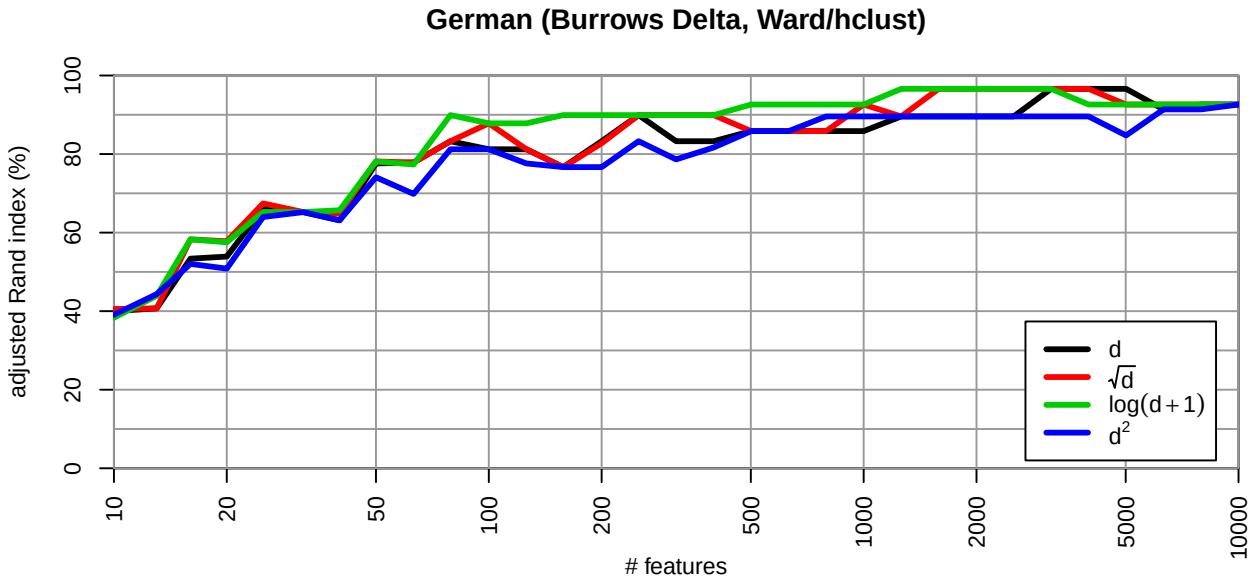
An easy explanation would be that the clustering algorithms react differently to the scaling of outlier distances, which increase much more quickly for Δ_Q than Δ_B, for instance. This effect could be counteracted by transforming the distance values using a suitable concave function (so that the metric property is preserved). As the plot below shows, neither reducing nor exaggerating outlier distances seems to have a substantial effect on authorship attribution quality:

```
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)", main="German (Burrows Delta, PAM)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="pam", trans=NULL)$adj.rand, lwd=3, 
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="pam", trans=sqrt)$adj.rand, lwd=3, 
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="pam", trans=function(x) log(x+1))$a
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="pam", trans=function(x) x*x)$adj.ra
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:4,
```

```
       legend=expression(d, sqrt(d), log(d+1), d^2))
```




```
plot(1, 100, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
     xlab="# features", ylab="adjusted Rand index (%)", main="German (Burrows Delta, Ward/hclust)",
     xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
draw.grid()
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="hclust", trans=NULL)$adj.rand, lwd=
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="hclust", trans=sqrt)$adj.rand, lwd=
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="hclust", trans=function(x) log(x+1)
lines(n.vals, evaluate(zDE, goldDE, n=n.vals, meth="manh", clust.m="hclust", trans=function(x) x*x)$adj
legend("bottomright", inset=.02, bg="white", lwd=3, col=1:4,
       legend=expression(d, sqrt(d), log(d+1), d^2))
```

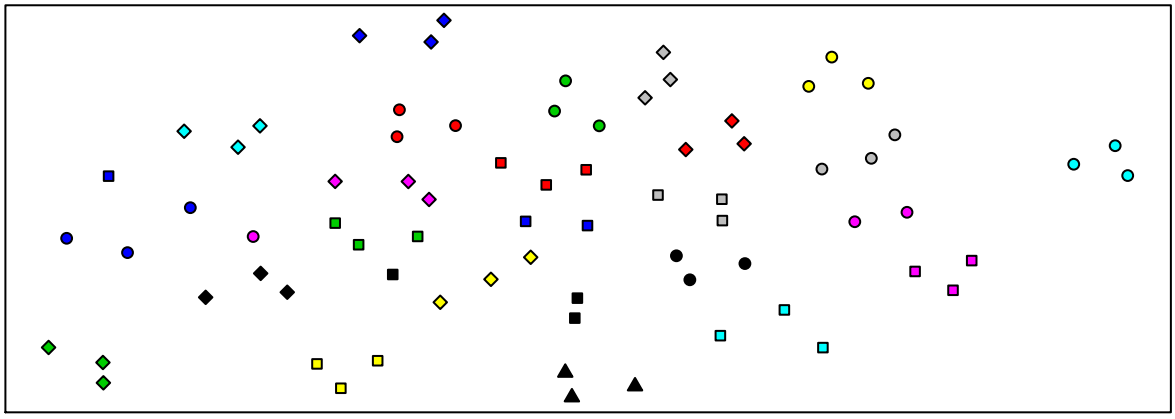


Understanding Delta

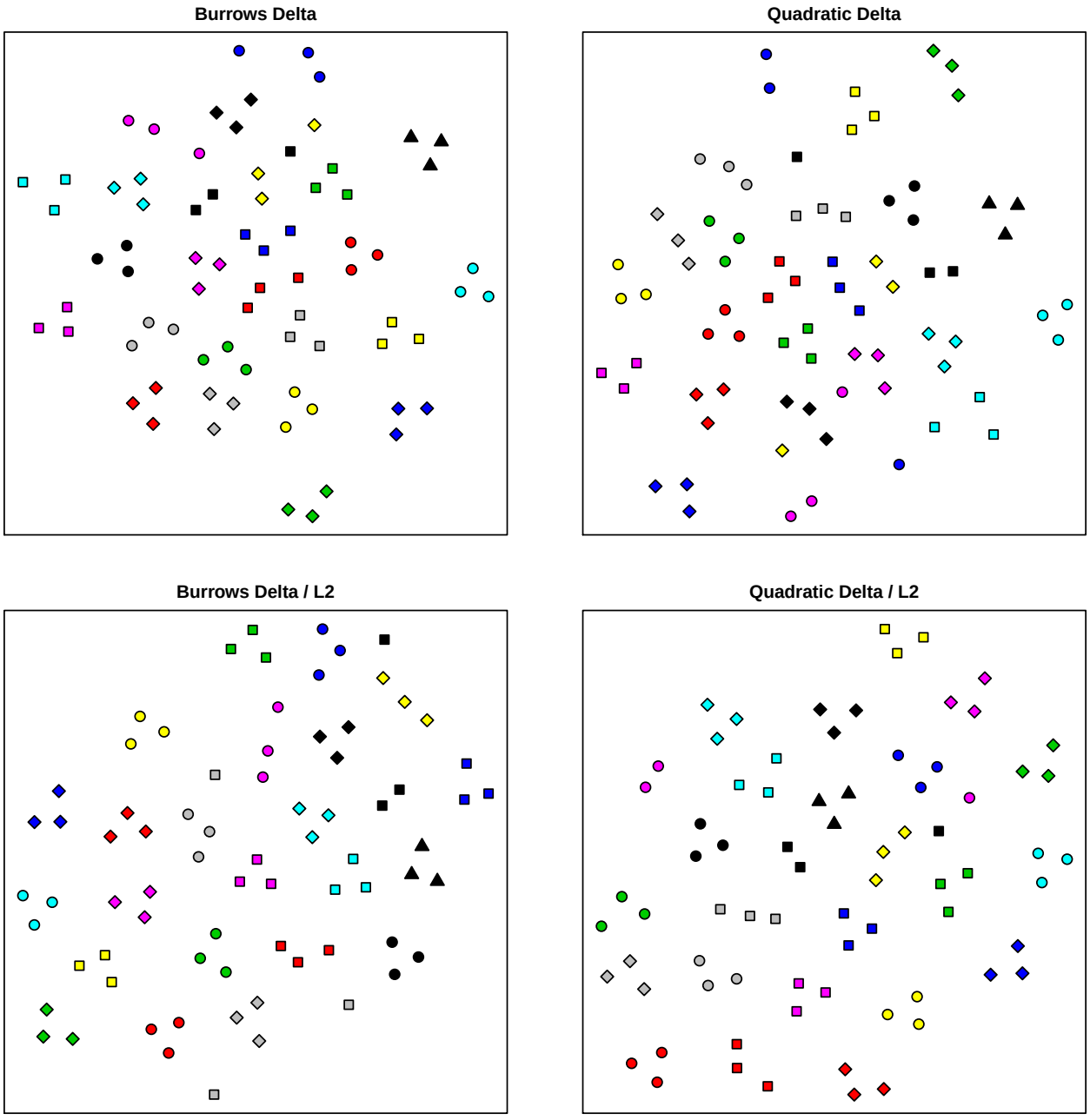
> **TODO:** control use of MDS / t-SNE with parameter; MDS is too messy to show anything, t-SNE seems far too clean; perhaps specify an MDS-based initial configuration for reproducible results? (but this would fail to re-arrange data points) or supply t-SNE embedding as initial config to MDS?

A quick first try. Use the optimal 500-dimensional vectors, and Δ_B and Δ_Q with and without L_2-normalization.

```
library(tsne)
mdsmap <- function (M, class, method="euclidean", ...) {
#  coord <- isoMDS(dist.matrix(M, method=method), trace=FALSE)$points
  coord <- suppressMessages(tsne(dist.matrix(M, method=method, as.dist=TRUE)))
  class <- as.factor(class)
  col.vals <- rep(1:8, 4)
  pch.vals <- rep(21:24, each=8)
  plot(coord, pch=pch.vals[class], bg=col.vals[class],
       xlab="", ylab="", xaxt="n", yaxt="n", ...)
}
mdsmap(zDE[,1:500], goldDE)
```



```
M <- zDE[, 1:500]
M2 <- normalize.rows(M)
par(mfrow=c(2,2), mar=c(2,2,2,2)+.1)
mdsmap(M, goldDE,  method="manhattan", cex=1.5, main="Burrows Delta")
mdsmap(M, goldDE,  method="euclidean", cex=1.5, main="Quadratic Delta")
mdsmap(M2, goldDE, method="manhattan", cex=1.5, main="Burrows Delta / L2")
mdsmap(M2, goldDE, method="euclidean", cex=1.5, main="Quadratic Delta / L2")
```




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

TODO: measure importance of features = contribution to relevant differences in distances, then visualize distribution (spike plot or importance histogram) and look at some top features

Visualizing feature vectors

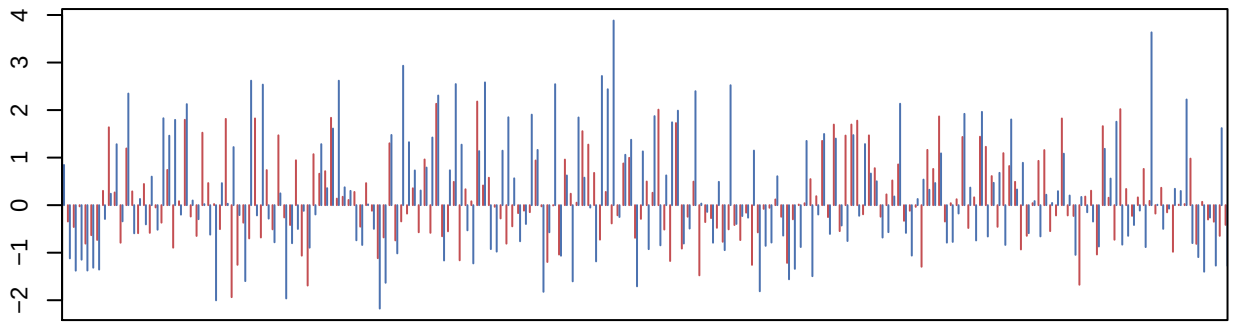
One possibility is to visualize feature vectors directly in the form of spike plots. Since it doesn't seem to be possible to display more than a few hundred dimensions, this approach is mostly useful as an illustration

of rescaling techniques. We use a colour palette based on percetual principles to ensure comparable visual contrasts between different pairs of vectors (Zeileis, Hornik, and Murrell 2009).

```
spike.pal <- rainbow_hcl(3, c=60, l=50)[c(1,3,2)] # suitable palette with clear contrasts
spike.pal <- seaborn.pal[c(3,1,2)] # or use same colours as Seaborn package for Python
spike.plot <- function (x, y, diff=FALSE, lwd=1, lwd2=1, col1=spike.pal[1], col2=spike.pal[2],
                        legend=NULL, ...) {
  n <- length(x); stopifnot(length(y) == n)
  x.vals <- 3 * (1:n)
  plot(c(x.vals, x.vals + 1), c(x, y), type=if (lwd > 0) "h" else "n",
       col=rep(c(col1, col2), each=n), lwd=lwd,
       xaxs="i", xlab="", ylab="", xaxt="n", ...)
  if (diff) segments(x.vals + .5, x, x.vals + .5, y, lwd=lwd2,
                     col=ifelse(x > y, "#00AA00", "#CC0000"))
  if (!is.null(legend)) legend("topright", inset=.02, bg="white", legend=legend,
                               col=c(col1,col2), lwd=lwd+2)
}
## spike.plot(zDE[1,1:150], zDE[2,1:150])  # for testing
```

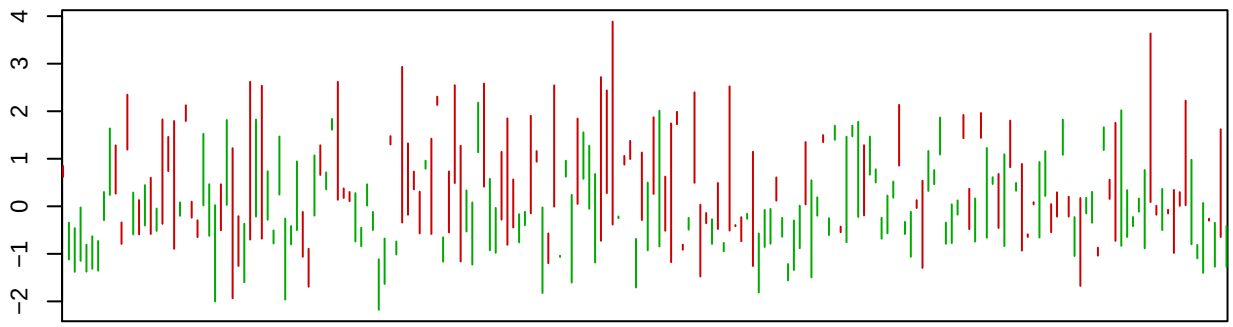
With normalization $n_w = 500$ words achieves optimal performance in all three languages. To obtain more interpretable pictures, let us look at the first 200 dimensions, which already produce a high-quality clustering. For this first test, we compare *Werther* against *Effie Briest*.

```
x <- zDE[2, 1:200]
y <- zDE[8, 1:200]
spike.plot(x, y)
```



The picture is clearer if we just show differences between the two feature vectors. Green segments indicate that *Werther* has a higher feature value than *Effi*, and red segments vice versa.

```
spike.plot(x, y, diff=TRUE, lwd=0)
```



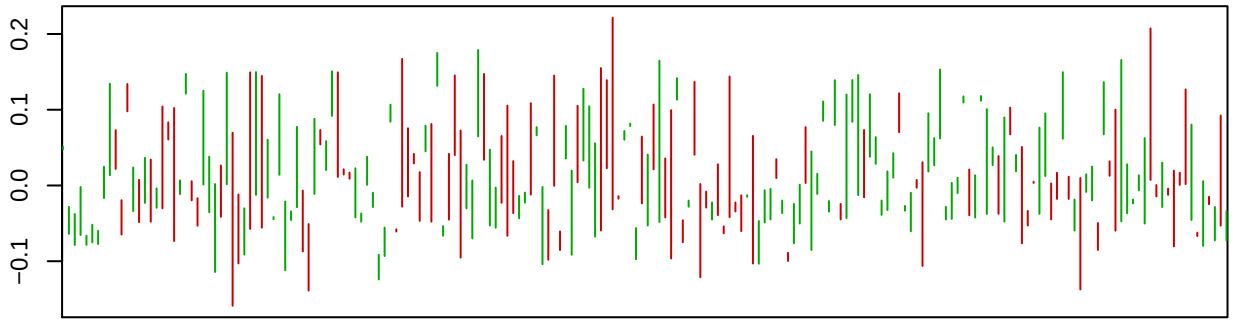
These profiles are amazingly different, but the two vectors also have markedly different lengths.

```
rowNorms(rbind(x, y))
```

```
##        x        y
## 12.19004 17.53309
```

Here is the corresponding spike plot for normalized vectors:

```
xy.norm <- normalize.rows(rbind(x, y))
spike.plot(xy.norm[1,], xy.norm[2,], diff=TRUE, lwd=0)
```



TODO: It will probably be more informative to compare the profiles of very similar texts by different authors, or to choose "interesting" text pairs from a MDS map. We should experiment with ordering the vectors, e.g. by feature informativeness, difference value, or one of the profiles.

Spike plots as illustration of "fingerprint" profiles

Spike plots might be useful to illustrate the notion of an author's "fingerprint". As an English example, let us compare Thomas Hardy's *Tess of the d'Urbervilles* (#57) with *Far from the Madding Crowd* (#53) and Charles Dickens's *Oliver Twist* (#34).

```
spike.idx <- c(53,57,34)
spike.legend <- c("Hardy: Far from the Madding Crowd",
                  "Hardy: Tess of the d'Urbervilles",
                  "Dickens: Oliver Twist")
```

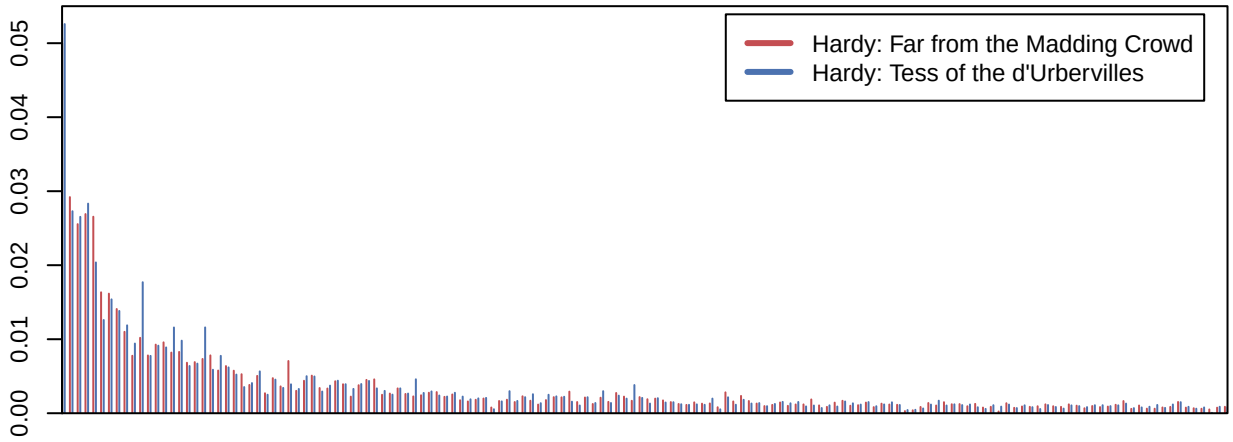
Relative frequencies

First, we look at the vectors $\mathbf{f}(D)$ of relative frequencies:

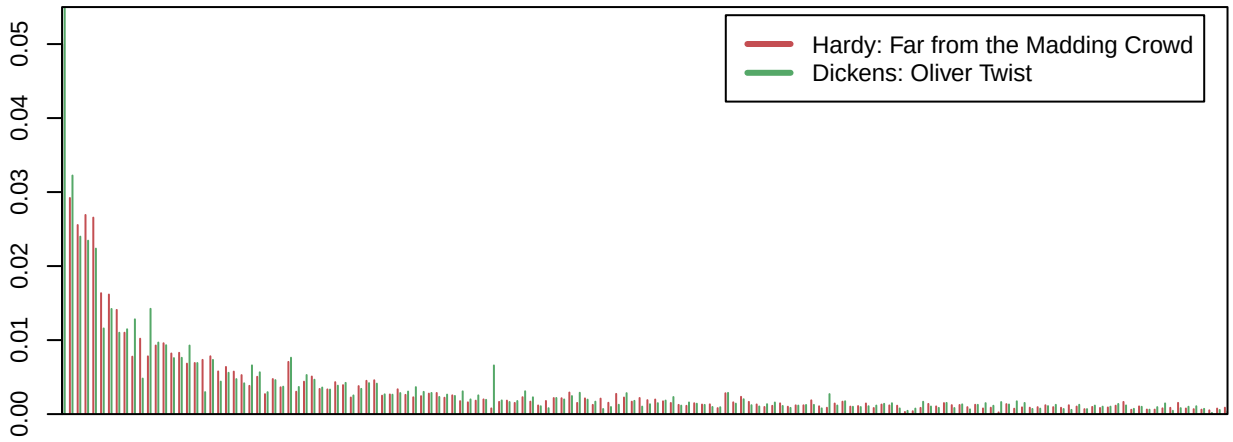
```
x <- FreqEN$S[spike.idx, ]
knitr::kable(round(x[, 1:12], 3))
```

	the	and	to	of	a	I	in	was	that	he	her	his
hardy: madding	0.051	0.029	0.026	0.027	0.027	0.016	0.016	0.014	0.011	0.008	0.010	0.008
hardy: tess	0.053	0.027	0.027	0.028	0.020	0.013	0.015	0.014	0.012	0.009	0.018	0.008
dickens: oliver	0.055	0.032	0.024	0.023	0.022	0.012	0.014	0.011	0.011	0.013	0.005	0.014

```
spike.plot(x[1, 1:150], x[2, 1:150], legend=spike.legend[1:2], yaxs="i", ylim=c(0, .055))
```




```
spike.plot(x[1, 1:150], x[3, 1:150], col2=spike.pal[3], legend=spike.legend[c(1,3)], yaxs="i", ylim=c(0
```



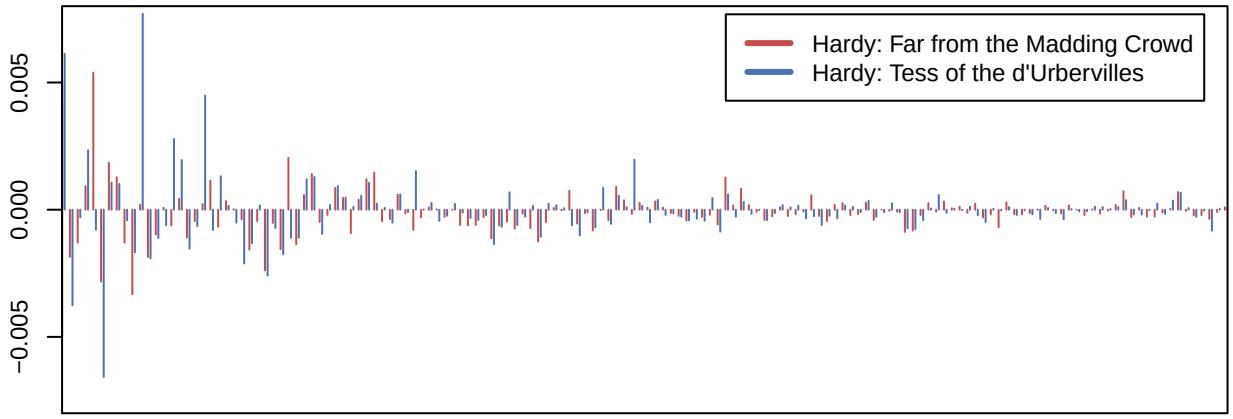
Overuse/underuse

The fingerprint (or "key profile") intuition is based on the overuse/underuse of a word by an author compared to other authors. Therefore, center the relative frequencies by subtracting the data set means, i.e. plot the values $f_i(D) - \mu_i$ for each text D:

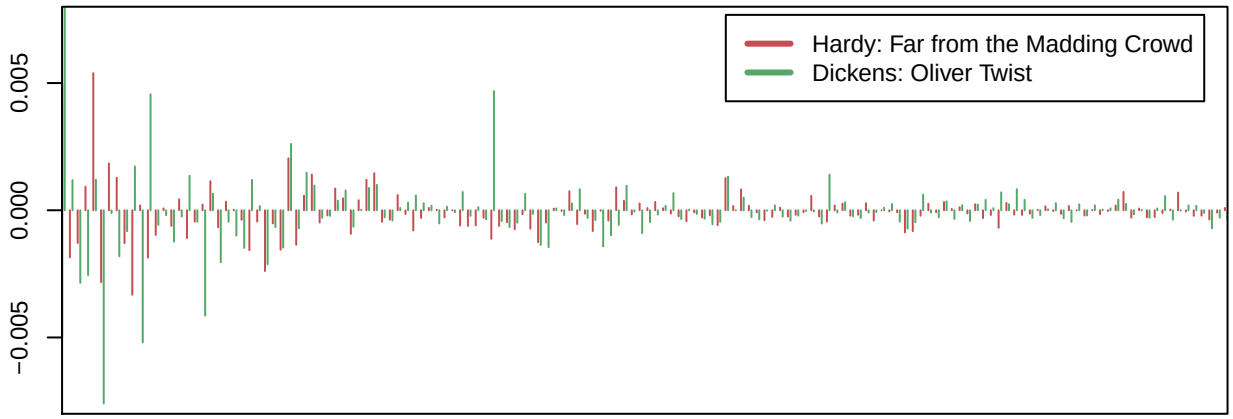
```
x <- scale(FreqEN$S, center=TRUE, scale=FALSE)[spike.idx, ]
knitr::kable(round(x[, 1:12], 3))
```

	the	and	to	of	a	I	in	was	that	he	her	his
hardy: madding	0.004	-0.002	-0.001	0.001	0.005	-0.003	0.002	0.001	-0.001	-0.003	0.000	-0.002
hardy: tess	0.006	-0.004	0.000	0.002	-0.001	-0.007	0.001	0.001	0.000	-0.002	0.008	-0.002
dickens: oliver	0.009	0.001	-0.003	-0.003	0.001	-0.008	0.000	-0.002	-0.001	0.002	-0.005	0.005

```
spike.plot(x[1, 1:150], x[2, 1:150], legend=spike.legend[1:2], yaxs="i", ylim=c(-.008, .008))
```




```
spike.plot(x[1, 1:150], x[3, 1:150], col2=spike.pal[3], legend=spike.legend[c(1,3)], yaxs="i", ylim=c(-
```



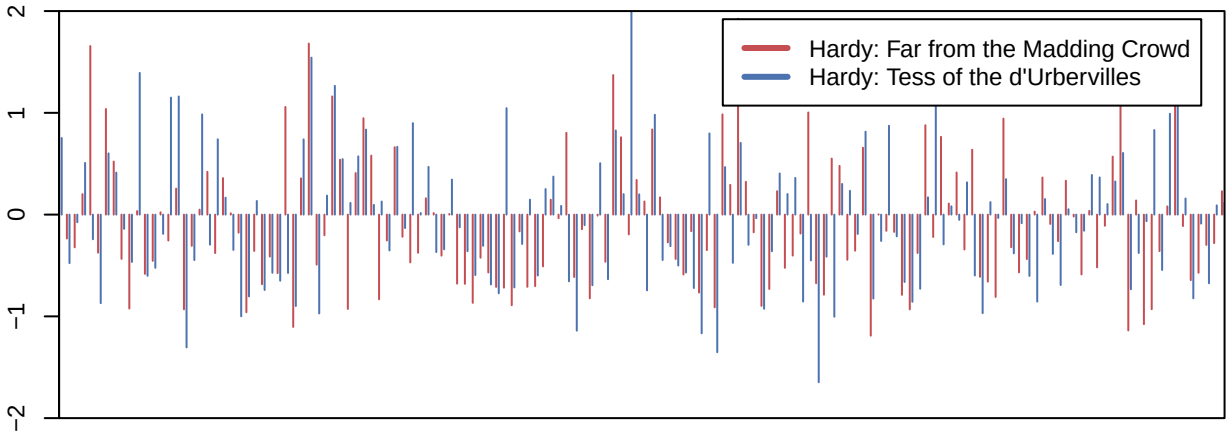
z-Scores give equal weight to all features

But the fingerprints only become clearly visible after standardization to z-scores $\mathbf{z}(D)$:

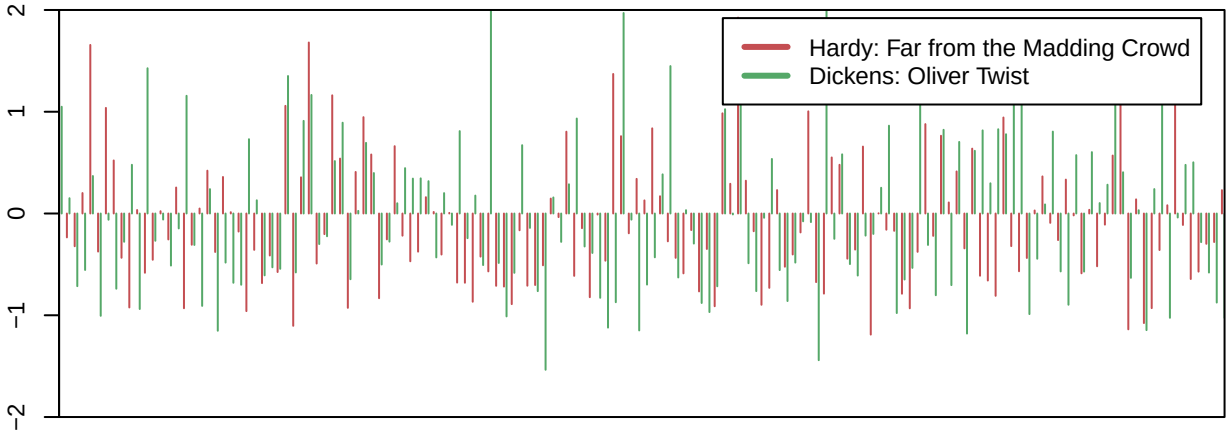
```
x <- zEN[spike.idx, ]
knitr::kable(round(x[, 1:12], 2))
```

	the	and	to	of	a	I	in	was	that	he	her	his
hardy: madding	0.53	-0.23	-0.32	0.20	1.66	-0.37	1.04	0.52	-0.44	-0.92	0.03	-0.58
hardy: tess	0.75	-0.48	-0.08	0.51	-0.24	-0.87	0.60	0.41	-0.14	-0.47	1.39	-0.60
dickens: oliver	1.05	0.15	-0.71	-0.56	0.37	-1.01	-0.06	-0.74	-0.28	0.48	-0.94	1.43

```
spike.plot(x[1, 1:150], x[2, 1:150], legend=spike.legend[1:2], yaxs="i", ylim=c(-2, 2))
```




```
spike.plot(x[1, 1:150], x[3, 1:150], col2=spike.pal[3], legend=spike.legend[c(1,3)], yaxs="i", ylim=c(-1
```



In case people in the audience have insufficient visual acuity to perceive the very obvious difference between the two pairs of profiles immediately, we confirm numerically that the profiles of the two Hardy novels are much more similar than Hardy vs. Dickens.

```
round(dist.matrix(zEN[spike.idx, 1:150], method="manhattan", as.dist=TRUE), 2)
```

```
##                  hardy: madding hardy: tess
## hardy: tess               74.56
## dickens: oliver          101.91       118.28
```

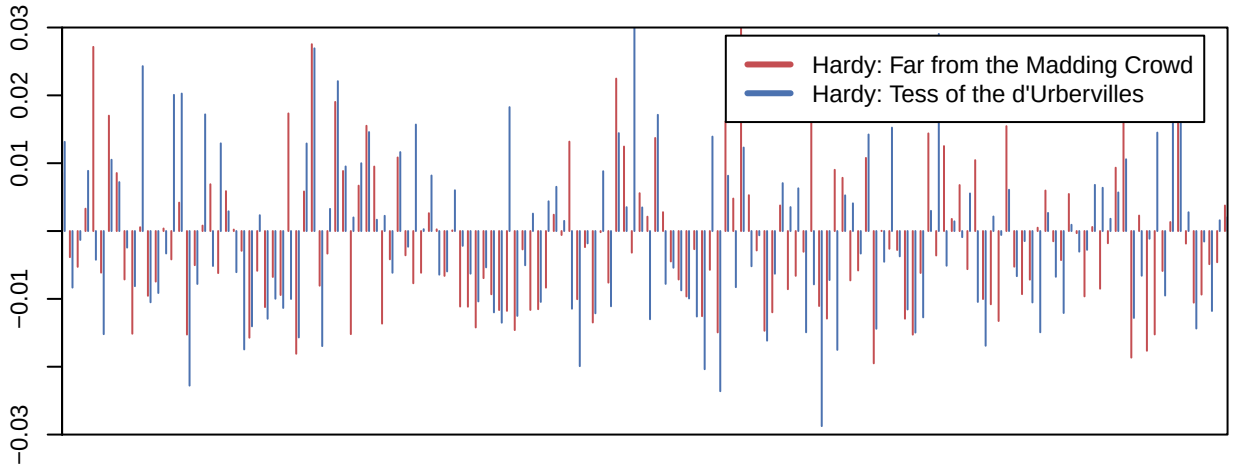
The effect of normalization

Now we can visualize the effect of vector normalization and other transformations.

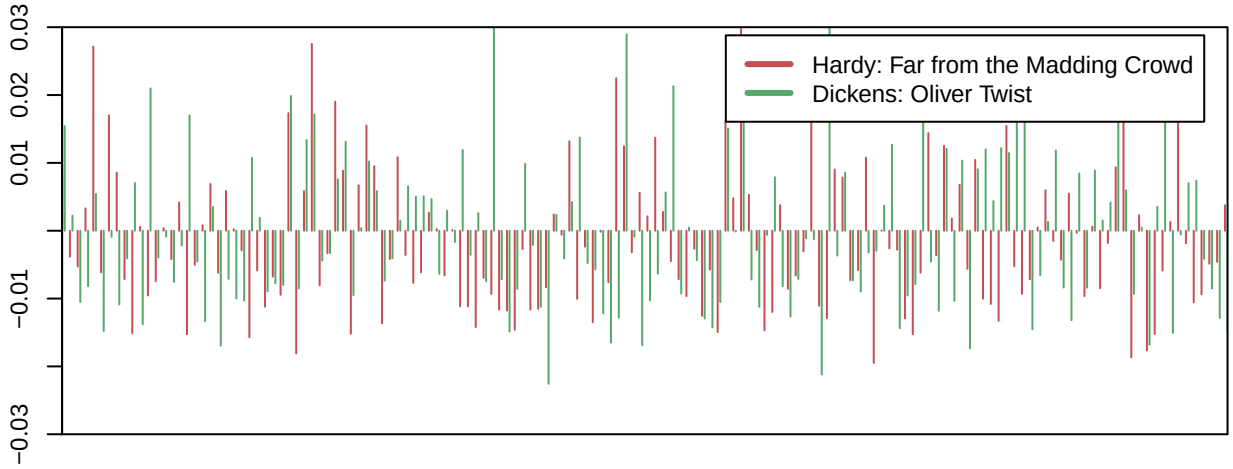
```
x <- normalize.rows(zEN[spike.idx, 1:5000])[, 1:150]
knitr::kable(round(x[, 1:12], 2))
```

	the	and	to	of	a	I	in	was	that	he	her	his
hardy: madding	0.01	0.00	-0.01	0.00	0.03	-0.01	0.02	0.01	-0.01	-0.02	0.00	-0.01
hardy: tess	0.01	-0.01	0.00	0.01	0.00	-0.02	0.01	0.01	0.00	-0.01	0.02	-0.01
dickens: oliver	0.02	0.00	-0.01	-0.01	0.01	-0.01	0.00	-0.01	0.00	0.01	-0.01	0.02

```
spike.plot(x[1, ], x[2, ], legend=spike.legend[1:2], yaxs="i", ylim=c(-.03, .03))
```




```
spike.plot(x[1, ], x[3, ], col2=spike.pal[3], legend=spike.legend[c(1,3)], yaxs="i", ylim=c(-.03, .03))
```



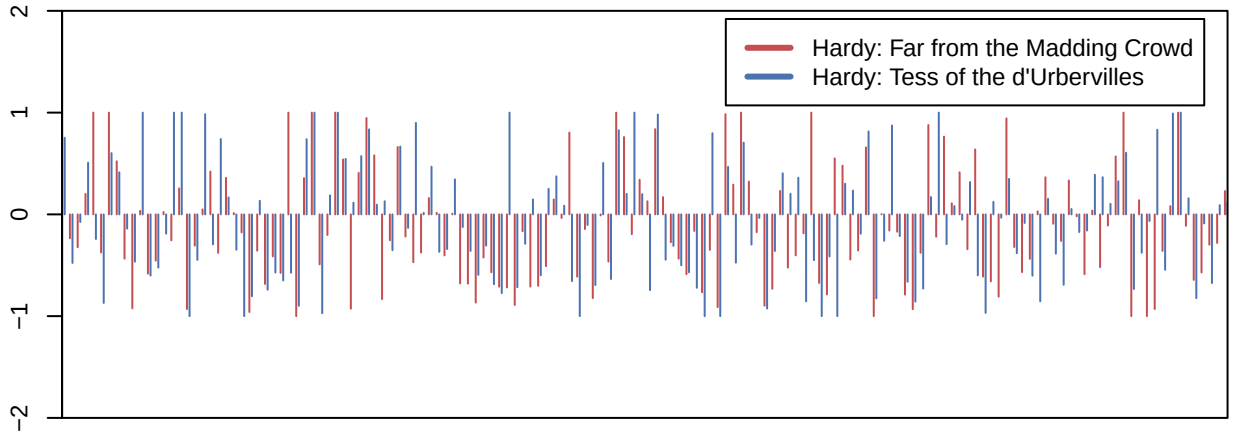
Clamping outlier values

For the illustration, we clamp z-scores to the range $[-1, +1]$ so that effect becomes more visible (y-axis only shows range $[-2, +2]$).

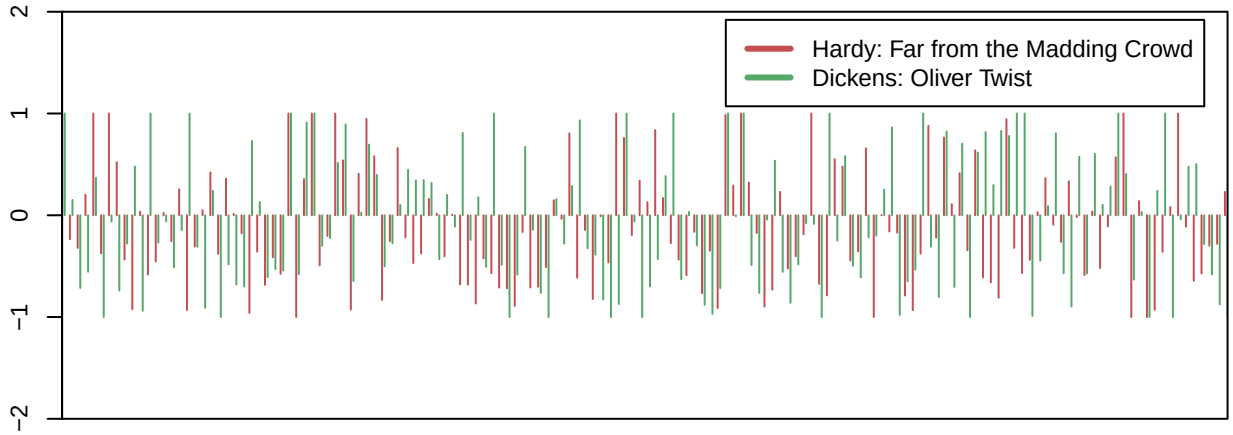
```
clamp <- function (x, min=-1, max=1) {
  pmax(pmin(x, max), min)
}
x <- clamp(zEN[spike.idx, ])
knitr::kable(round(x[, 1:12], 2))
```

	the	and	to	of	a	I	in	was	that	he	her	his
hardy: madding	0.53	-0.23	-0.32	0.20	1.00	-0.37	1.00	0.52	-0.44	-0.92	0.03	-0.58
hardy: tess	0.75	-0.48	-0.08	0.51	-0.24	-0.87	0.60	0.41	-0.14	-0.47	1.00	-0.60
dickens: oliver	1.00	0.15	-0.71	-0.56	0.37	-1.00	-0.06	-0.74	-0.28	0.48	-0.94	1.00

```
spike.plot(x[1, 1:150], x[2, 1:150], legend=spike.legend[1:2], yaxs="i", ylim=c(-2, 2))
```




```
spike.plot(x[1, 1:150], x[3, 1:150], col2=spike.pal[3], legend=spike.legend[c(1,3)], yaxs="i", ylim=c(-
```



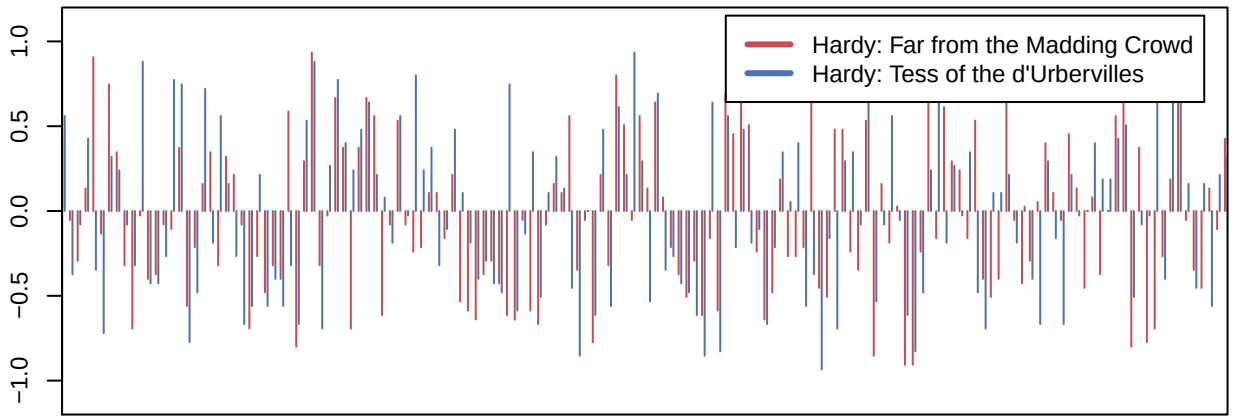
Quantile transformation

Another way of giving truly equal weight to all features – even those with highly skewed distributions – is to transform the relative frequencies to empirical quantiles (across all texts in the collection).

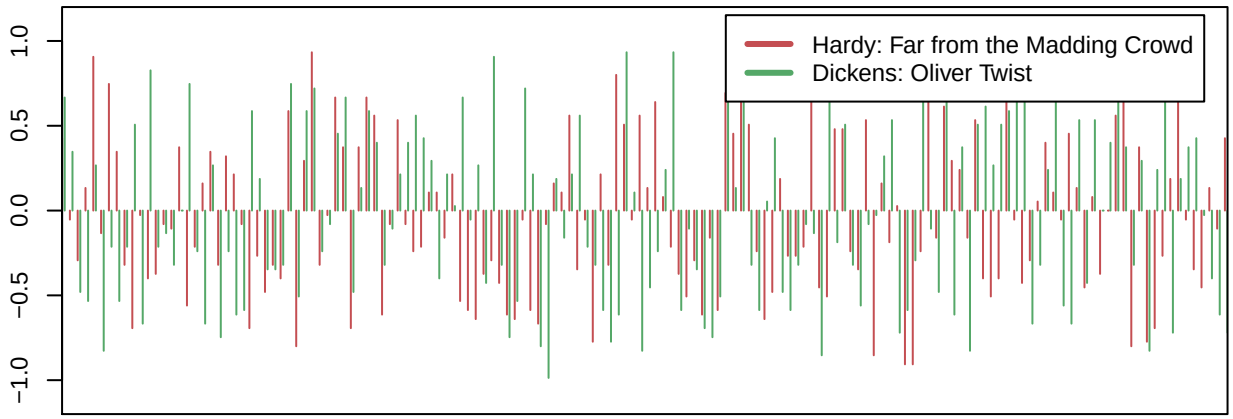
```
quantile.score <- function(x) {
  if (is.matrix(x)) {
    apply(x, 2, quantile.score) # apply to columns of matrix
  } else {
    2 * ((rank(x, ties="average") - 0.5) / length(x)) - 1
  }
}
x <- quantile.score(zEN[, 1:150])[spike.idx, ] # limit expensive quantile transform to relevant columns
knitr::kable(round(x[, 1:12], 2))
```

	the	and	to	of	a	I	in	was	that	he	her	his
hardy: madding	0.43	-0.05	-0.29	0.13	0.91	-0.13	0.75	0.35	-0.32	-0.69	-0.03	-0.40
hardy: tess	0.56	-0.37	-0.08	0.43	-0.35	-0.72	0.32	0.24	-0.08	-0.32	0.88	-0.43
dickens: oliver	0.67	0.35	-0.48	-0.53	0.27	-0.83	-0.21	-0.53	-0.21	0.51	-0.67	0.83

```
spike.plot(x[1, ], x[2, ], legend=spike.legend[1:2], yaxs="i", ylim=c(-1.2, 1.2))
```




```
spike.plot(x[1, ], x[3, ], col2=spike.pal[3], legend=spike.legend[c(1,3)], yaxs="i", ylim=c(-1.2, 1.2))
```



Ternarization

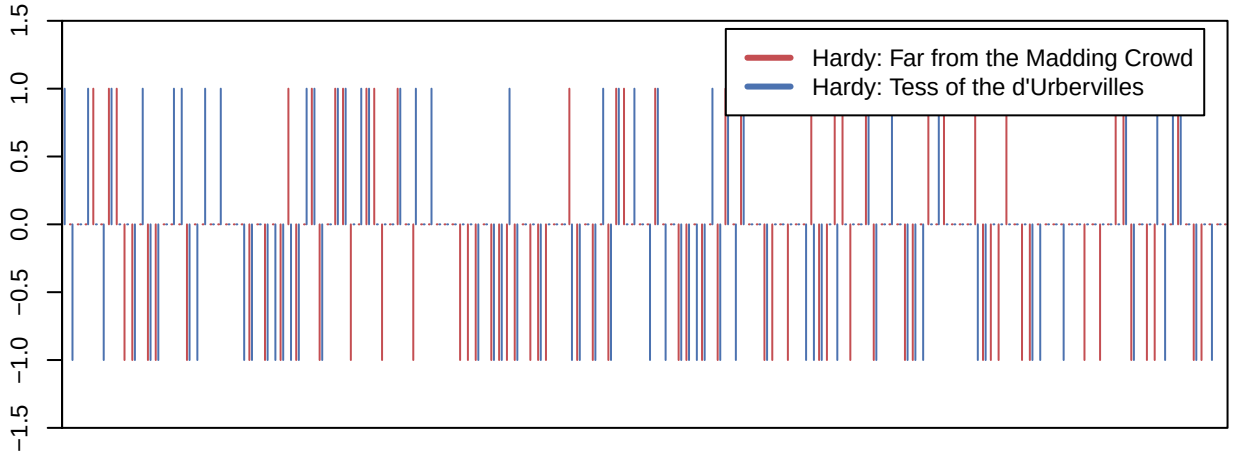
An extreme form of deskewing the distribution of z-scores is quantization the feature values into underuse, neutral and overuse categories (ternarization) or into presence vs. absence (binarization).

The default ternarization strategy sets thresholds so that a third of the data points each fall into the positive, neutral and negative category assuming a standard normal distribution.

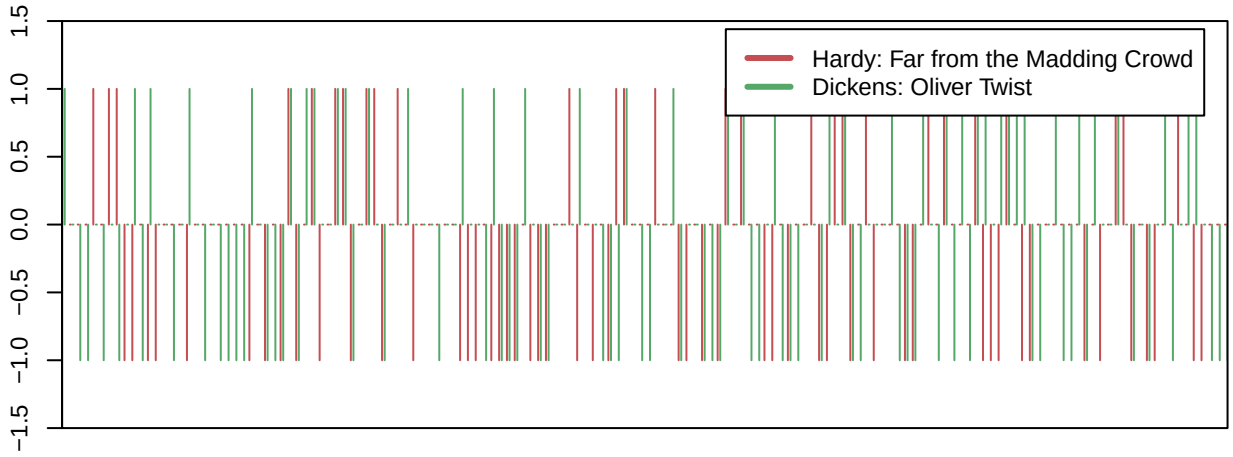
```
x <- ternarize(zEN[spike.idx, ], neutral.p=1/3)
knitr::kable(round(x[, 1:12], 2))
```

	the	and	to	of	a	I	in	was	that	he	her	his
hardy: madding	1	0	0	0	1	0	1	1	-1	-1	0	-1
hardy: tess	1	-1	0	1	0	-1	1	0	0	-1	1	-1
dickens: oliver	1	0	-1	-1	0	-1	0	-1	0	1	-1	1

```
spike.plot(x[1, 1:150], x[2, 1:150], legend=spike.legend[1:2], yaxs="i", ylim=c(-1.5, 1.5))
```




```
spike.plot(x[1, 1:150], x[3, 1:150], col2=spike.pal[3], legend=spike.legend[c(1,3)], yaxs="i", ylim=c(-
```

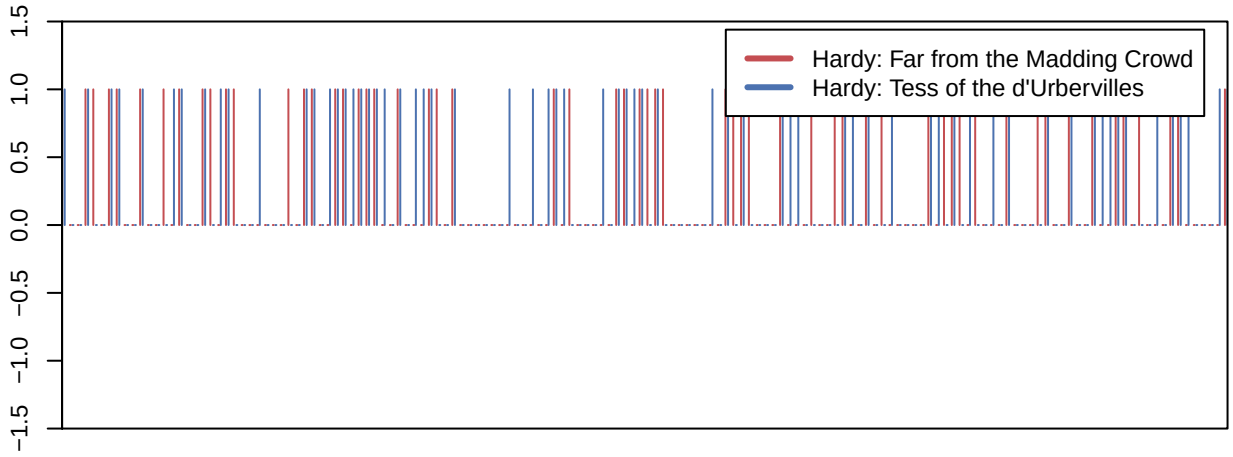


Alternatively, we can binarize the feature vectors into two categories. For low-frequency features (roughly $n_w > 1000$) this corresponds to presence vs. absence of the respective word (similar to a binarized vector space model in IR); for high-frequency features, the categories correspond to overuse ($z > 0$) vs. underuse ($z \leq 0$). Note that the visualization below only covers high-frequency words, so the spikes visible in the plot mark overuse.

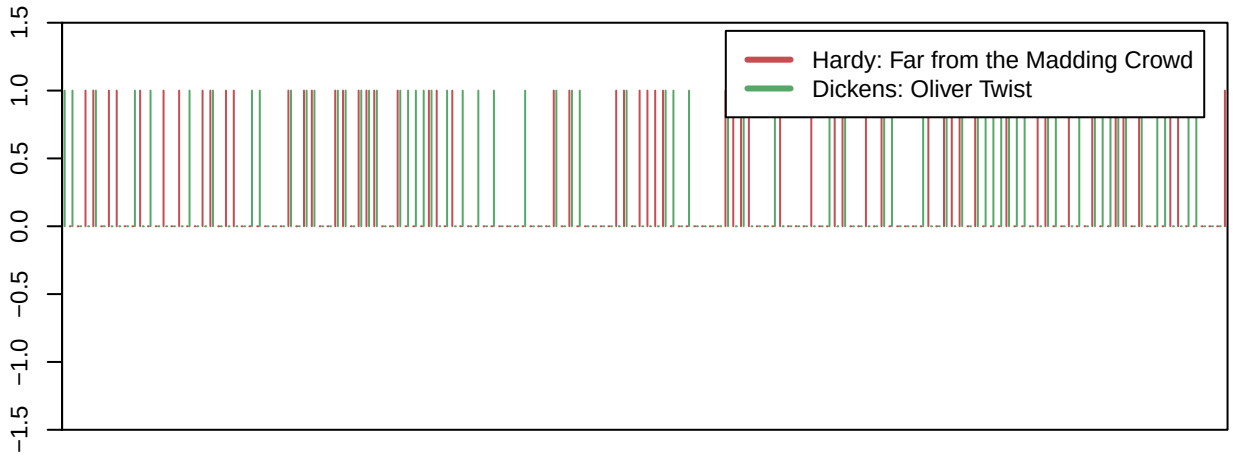
```
x <- binarize(zEN[spike.idx, ])
knitr::kable(round(x[, 1:12], 2))
```

	the	and	to	of	a	I	in	was	that	he	her	his
hardy: madding	1	0	0	1	1	0	1	1	0	0	1	0
hardy: tess	1	0	0	1	0	0	1	1	0	0	1	0
dickens: oliver	1	1	0	0	1	0	0	0	0	1	0	1

```
spike.plot(x[1, 1:150], x[2, 1:150], legend=spike.legend[1:2], yaxs="i", ylim=c(-1.5, 1.5))
```

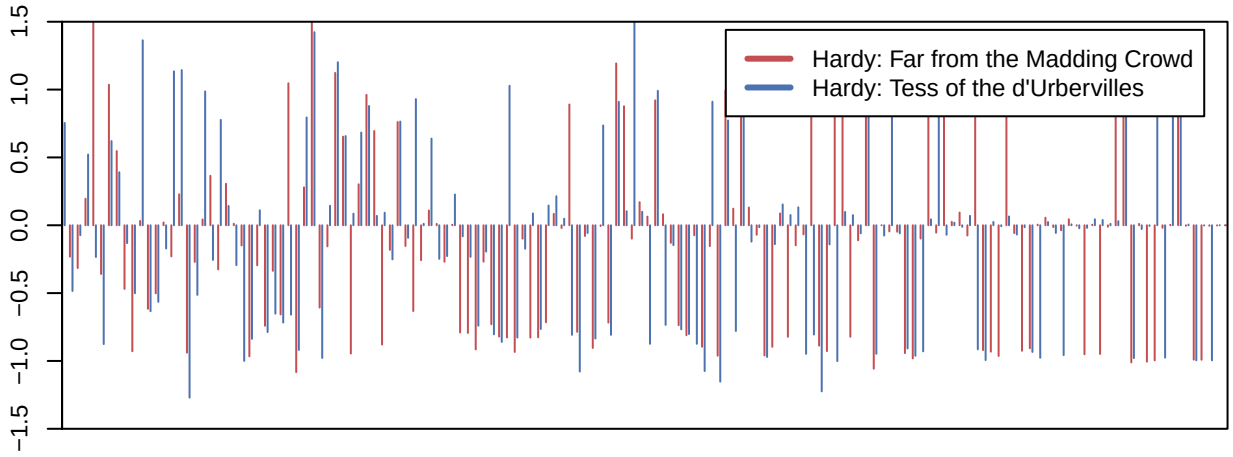



```
spike.plot(x[1, 1:150], x[3, 1:150], col2=spike.pal[3], legend=spike.legend[c(1,3)], yaxs="i", ylim=c(-
```

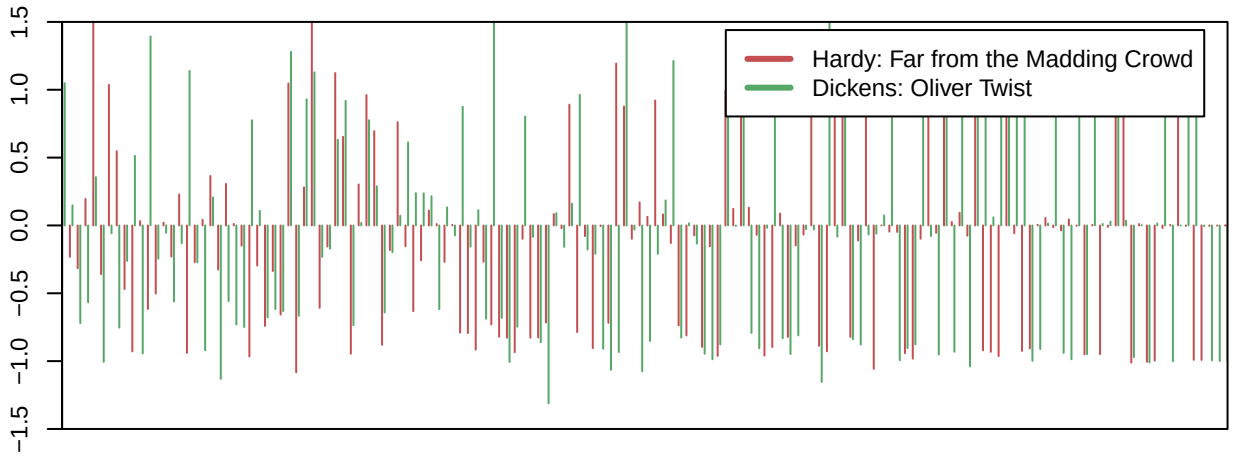


The main rationale behind binarization / ternarization is that z-scores aren't meaningful for very sparse features with a highly skewed distribution; in this case, we should only distinguish between presence and absence (or over-/underuse) rather than measure the degree of deviation. This suggests a mixed approach where MFW features transition from z-scores to ternarized values.

```
x <- ternarize(zEN[spike.idx, ], neutral.p=1/3, crossover=150)
spike.plot(x[1, 1:150], x[2, 1:150], legend=spike.legend[1:2], yaxs="i", ylim=c(-1.5, 1.5))
```

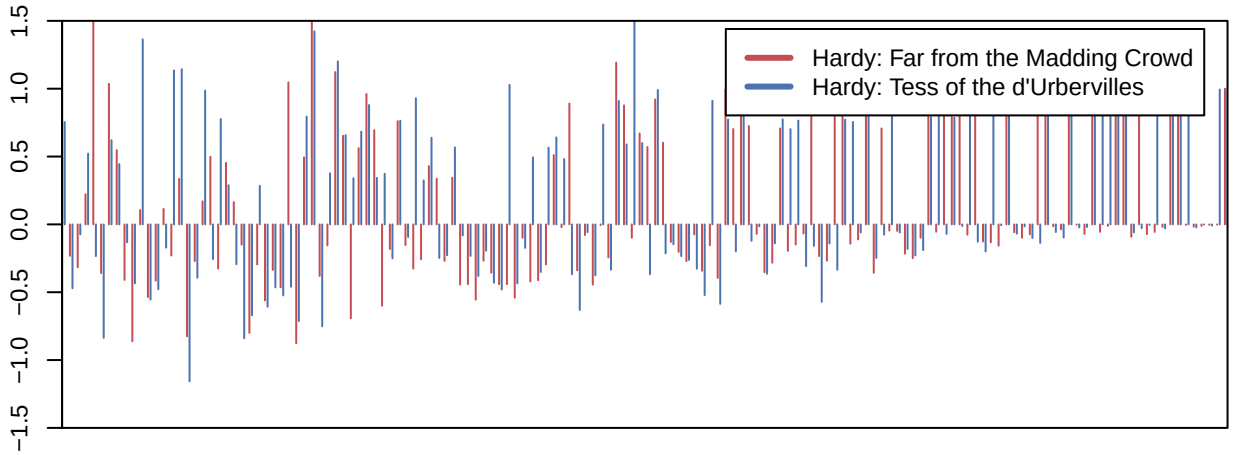



```
spike.plot(x[1, 1:150], x[3, 1:150], col2=spike.pal[3], legend=spike.legend[c(1,3)], yaxs="i", ylim=c(-
```

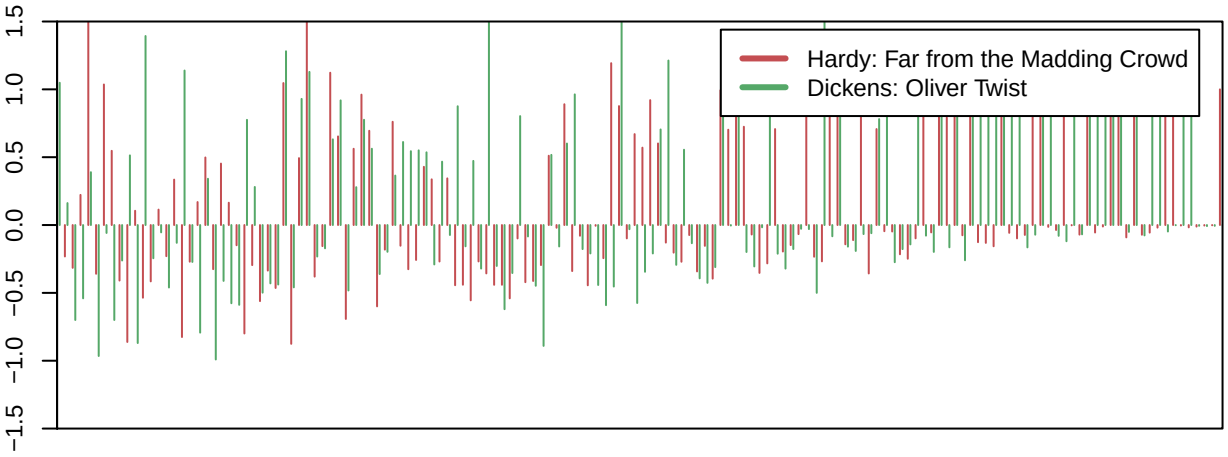


Such a mixed approach seems particularly appropriate for binarization, with a long crossover phase so that full binarization is only achieved for $n_w \gg 1000$.

```
x <- binarize(zEN[spike.idx, ], crossover=150)
spike.plot(x[1, 1:150], x[2, 1:150], legend=spike.legend[1:2], yaxs="i", ylim=c(-1.5, 1.5))
```



```
spike.plot(x[1, 1:150], x[3, 1:150], col2=spike.pal[3], legend=spike.legend[c(1,3)], yaxs="i", ylim=c(-
```



References

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.

Zeileis, Achim, Kurt Hornik, and Paul Murrell. 2009. "Escaping RGBland: Selecting Colors for Statistical Graphics." *Computational Statistics & Data Analysis* 53: 3259–70.