# Text Classification for Organizational Researchers

*A Tutorial*

Kobayashi, V.B.; Mol, S.T.; Berkers, H.A.; Kismihók, G.; Den Hartog, D.N.

*Published in:*
Organizational Research Methods

*DOI:*
10.1177/1094428117719322

*Article*

Organizational Research Methods
2018, Vol. 21(3) 766-799

DOI: 10.1177/1094428117719322
journals.sagepub.com/home/orm
SAGE

# Text Classification for Organizational Researchers: A Tutorial

**Vladimer B. Kobayashi[1], Stefan T. Mol[1], Hannah A. Berkers[1], Gábor Kismihók[1] and Deanne N. Den Hartog[1]**

**Abstract**

Organizations are increasingly interested in classifying texts or parts thereof into categories, as this enables more effective use of their information. Manual procedures for text classification work well for up to a few hundred documents. However, when the number of documents is larger, manual procedures become laborious, time-consuming, and potentially unreliable. Techniques from text mining facilitate the automatic assignment of text strings to categories, making classification expedient, fast, and reliable, which creates potential for its application in organizational research. The purpose of this article is to familiarize organizational researchers with text mining techniques from machine learning and statistics. We describe the text classification process in several roughly sequential steps, namely training data preparation, preprocessing, transformation, application of classification techniques, and validation, and provide concrete recommendations at each step. To help researchers develop their own text classifiers, the R code associated with each step is presented in a tutorial. The tutorial draws from our own work on job vacancy mining. We end the article by discussing how researchers can validate a text classification model and the associated output.

**Keywords**

text classification, text mining, random forest, support vector machines, naive Bayes

Text data are pervasive in organizations. Digitization (Cardie & Wilkerson, 2008) and the ease of creating online information (e.g., e-mail messages; Berry & Castellanos, 2008) contributes to the vast quantities of text generated each day. Embedded in these texts is information that may improve our understanding of organizational processes. Thus, organizational researchers increasingly seek

[1]Leadership and Management Group, Amsterdam Business School, University of Amsterdam, Amsterdam, The Netherlands

**Corresponding Author:**
Stefan T. Mol, Leadership and Management Group, Amsterdam Business School, University of Amsterdam, Valckenierstraat 59, 1018 XE Amsterdam, The Netherlands.
Email: s.t.mol@uva.nl

ways to organize, classify, label, and extract opinions, experiences, and sentiments from text (Pang & Lee, 2008; Wiebe, Wilson, & Cardie, 2005). Up until recently, the majority of text analyses in organizations relied on time consuming and labor-intensive manual procedures, which are impractical and less effective for voluminous collections of *documents* especially when resources are limited (Kobayashi et al., 2018). Hence, automatic (or computer-assisted) strategies are increasingly employed to accelerate the analysis of text (Berry & Castellanos, 2008).

Similar to content analysis (Duriau, Reger, & Pfarrer, 2007; Hsieh & Shannon, 2005; Scharkow, 2013) and template analysis (Brooks, McCluskey, Turley, & King, 2015), a common objective of text analysis is to assign text to predefined categories. Manually assigning large collections of text to categories is costly and may become inaccurate and unreliable due to cognitive overload. Furthermore, idiosyncrasies among human coders may creep into the labeling process resulting in coding errors. One workaround is to code only part of the *corpus* as opposed to coding all documents. However, this comes at the expense of possibly omitting relevant information, which may lead to bias and a degradation of the internal and external validity of the findings. Another option is to hire multiple human coders, but this adds cost (e.g., cost of hiring and training coders) and effort pertaining to determining interrater reliability and consensus seeking (Sheng, Provost, & Ipeirotis, 2008). A final (and more affordable) option is to solicit the help of the public to label text, for instance through the Amazon Mechanical Turk platform (Buhrmester, Kwang, & Gosling, 2011). However, this may be effective only in labeling objective information (e.g., names of people, events, etc.) since it is often difficult to establish consistency on subjective labels (e.g., sentiments; Wiebe, Wilson, Bruce, Bell, & Martin, 2004). Hence, automatic text analysis procedures that reliably, efficiently, and effectively assign text elements to classes are both necessary and advantageous especially in dealing with a massive corpus of text.

This article focuses on automatic text classification for several reasons. First, although text classification (henceforth TC) has been applied in various fields, such as in political science (Atteveldt, Kleinnijenhuis, Ruigrok, & Schlobach, 2008; B. Yu, Kaufmann, & Diermeier, 2008), occupational fraud (Holton, 2009), law (Gonçalves & Quaresma, 2005), finance (Chan & Chong, 2017; Chan & Franklin, 2011; Kloptchenko et al., 2004), and personality research (Shen, Brdiczka, & Liu, 2013), so far its uptake in organizational research is limited. Second, the use of TC is economical both in terms of time and cost (Duriau et al., 2007). Third, many of the techniques that have been developed in TC, such as sentiment analysis (Pang & Lee, 2008), genre classification (Finn & Kushmerick, 2006), and sentence classification (Khoo, Marom, & Albrecht, 2006) seem particularly well suited to address contemporary organizational research questions. Fourth, the acceptance and broader use of TC within the organizational research community can stimulate the development of novel TC techniques.

Tutorials or review-tutorials on TC that have been published so far (Harish, Guru, & Manjunath, 2010; Li & Jain, 1998; Sebastiani, 2002) were targeted mainly toward researchers in the field of machine learning and data mining. This has resulted in a skewed focus on technical and methodological details. In this article our goal is to balance the discussion among techniques, theoretical concepts, and validity concerns to increase the accessibility of TC to organizational researchers.

Below we first discuss the TC process, by pointing out key concerns and providing concrete recommendations at each step. Previous studies are cited to enrich the discussion and to illustrate different use cases. The second part is a hands-on tutorial using part of our own work as a running example. We applied TC to automatically extract nursing job tasks from nursing vacancies to augment nursing job analysis (Kobayashi, Mol, Kismihók, & Hesterberg, 2016). The findings from this study were used in the EU-funded Pro-Nursing (http://pro-nursing.eu) project which aimed to understand, among others, how nursing tasks are embedded in the nursing process. We also address validity assessment because the ability to demonstrate the validity of TC outcomes will likely be critical to its uptake by organizational researchers. Thus, we discuss and illustrate how to establish

validity for TC outcomes. Specifically, we address assessing the predictive validity of the classifier and triangulating the output of the classification with other data sources (e.g., expert input and output from alternative analyses).

## Text Classification

TC is defined as the automatic assignment of text to one or more predefined classes (Li & Jain, 1998; Sebastiani, 2002). Formally, the task of TC is stated as follows. Given a set of text and a set of categories, construct a model of the form: $Y = f(X, \theta) + \varepsilon$ from a set of documents with known categories. In the preceding formula, $X$ is a suitably chosen text representation (e.g., a vector), $\theta$ is the set of unknown parameters associated with the function $f$ (also known as the *classifier* or *classification model*) that need to be estimated using the training data, and $\varepsilon$ is the error of the classification. The error is added to account for the fact that $f$ is just an approximation to the true but unknown function $h$ such that $Y = h(X)$. Hence, the smaller $\varepsilon$ is, the more *effective* the classifier $f$ is. The $Y$ term usually takes numerical values indicating the membership of text to a particular category. For example, when there are only 2 categories, such as in classifying the polarity of relations between political actors and issues, as either positive or negative (Atteveldt et al., 2008), $Y$ can take the values of $+1$ and $-1$, respectively signifying positive and negative sentiment. We further discuss how to deal with each part of the formula, such as how to choose $X$ and $f$, below. Once the classification model has been constructed it is then used to predict the category of new text (Aggarwal & Zhai, 2012).

An ideal classifier would mimic how humans process and deduce meaning from text. However, there are still many challenges before this becomes reality. Natural languages contain high-level semantics and abstract concepts (Harish et al., 2010; Popping, 2012) that are difficult to articulate in computer language. For instance, the meaning of a word may change depending on the context in which it is used (Landauer, Foltz, & Laham, 1998). Also, lexical, syntactic, and structural ambiguities in text are continuing challenges that would need to be addressed (Hindle & Rooth, 1993; Popping, 2012). Another issue is dealing with typographical errors or misspellings, abbreviations, and new lexicons. Strategies for dealing with ambiguities all need to be explicated during classifier development. Before a classifier is deployed it thus needs several rounds of training, testing, fine-tuning (of parameters), and repeated evaluation until acceptable levels of performance and validity are reached. The resulting classifier is expected to approximate the performance of human experts in classification tasks (Cardie & Wilkerson, 2008), but for a large corpus its advantage is that it will be able to do so in a faster, cheaper, and more reliable manner.

## TC: The Process

The TC process consists of six interrelated steps, namely (a) text preprocessing, (b) text representation or transformation, (c) dimensionality reduction, (d) selection and application of classification techniques, (e) classifier evaluation, and (f) classifier validation. As with any research activity, before starting the TC process, we begin by formulating the research question and identifying text of interest. Here, we assume that classes are predefined and that the researcher has access to, or can gather, documents with known classes, that is, the *training data*. For example, in a study about identifying disgruntled employee communications, researchers used posts from intracompany discussion groups. Subsequently, using criteria on employee disgruntlement, two people manually classified 80 messages into either disgruntled or nondisgruntled communication (Holton, 2009). Another study focused on the detection of personality of users from their email messages. Researchers first administered a 120-item questionnaire to 486 users to identify their personalities after which their email messages over a 12-month period were collected (Shen et al., 2013). Compared to the

study on disgruntlement, it is more straightforward to label the associated text in this latter study because the labels are based on the questionnaire. Researchers are often faced with the decision of how many documents to label, an issue we will return to in the "Other TC issues" section below. Once the training dataset has been compiled, the next step is to preprocess the documents.

### *Text Preprocessing for Classification*

The purpose of preprocessing is to remove irrelevant bits of text as these may obscure meaningful patterns and lead to poor classification performance and redundancy in the analysis (Uysal & Gunal, 2014). During preprocessing we first apply *tokenization* to separate individual terms. Terms may be words, punctuation marks, numbers, tags, and other symbols (e.g., an emoticon). In written English, terms are usually separated by spaces.

Punctuations and numbers, if deemed irrelevant to the classification task at hand are removed, although in some cases these may be informative and thus retained (exclamation marks or emoticons, for instance, may be indicative of sentiment). Dictionaries or lexicons are used to apply spelling correction, and to resolve typos, and abbreviations. Words that are known to have low information content such as conjunctions and prepositions are typically deleted. These words are called *stopwords* (Fox, 1992), examples of pre-identified stopwords in the English language are "and," "the," and "of" (see http://www.ranks.nl/stopwords for different lists stopwords in various languages). When the case of the letters is irrelevant it is advisable to transform all upper case letters into lower case.

During preprocessing *stemming*, which is defined as the process of obtaining the base or stem form of words (Frakes, 1992; Porter, 1980), is also commonly applied. A key assumption in stemming is that words that have similar root forms are identical in meaning. Stemming is performed by removing suffixes that may not correspond to an actual base form of the word (Willett, 2006). For example, the words *calculate*, *calculating*, *calculated* will be rewritten to *calculate* although the actual base form is *calculate* (Toman, Tesar, & Jezek, 2006). If one wants to recover the actual base form then one can use *lemmatization* instead of stemming. However, lemmatization is more challenging than stemming (Toman et al., 2006) and the added complexity of applying lemmatization may offset its benefits. Both lemmatization and stemming leads to a loss of inflection information in words (e.g., tense, gender, and voice). Inflection information may be important in some applications, such as in identifying the sentiment of product reviews, since as it turns out, most negative reviews are written in the past tense (Dave, Lawrence, & Pennock, 2003). Stemming and lemmatization are part of a broad class of preprocessing techniques called *normalization* (Dave et al., 2003; Toman et al., 2006). The aim of normalization is to merge terms that express the same idea or concept under a single code called a *template*. For example, another normalization strategy is to use the template POST_CODE to replace all occurrences of postcodes in a collection of documents. This can be useful when it is important to consider if a document does or does not contain a postcode (i.e., contains an address), but the actual postcode is irrelevant.

A practical question is: what preprocessing techniques to apply for a given text? The answer is largely determined by the nature of text (e.g., language and genre), the problem that we want to address, and the application domain (Uysal & Gunal, 2014). Any given preprocessing procedure may be useful for a specific domain of application or language but not for others. Several empirical studies demonstrated the effect of preprocessing on classification performance. For example, stemming in the Turkish language does not seem to make a difference in classification performance when the size of the training data set is large (Torunoğlu, Çakırman, Ganiz, Akyokuş, & Gürbüz, 2011). In some applications stemming even appears to degrade classification performance, particularly in the English and Czech languages (Toman et al., 2006). In the classification of English online news, the impact of both stemming and stopword removal is negligible (Song, Liu, & Yang, 2005). In general,

the classification of English and Czech documents benefits from stopword removal but may suffer from word normalization (Toman et al., 2006). For the Arabic language, certain classifiers benefit from stemming (Kanaan, Al-Shalabi, Ghwanmeh, & Al-Ma'adeed, 2009). In spam email filtering, some words typically seen as stopwords (e.g., "however" or "therefore") were found to be particularly rare in spam email, hence these should not be removed for this reason (Méndez, Iglesias, Fdez-Riverola, Díaz, & Corchado, 2006).

*Recommendation.* In using English documents, our general recommendation is to apply word tokenization, convert upper case letters to lower case, and apply stopword removal (except for short text such as email messages and product titles; Méndez et al., 2006; H.-F. Yu, Ho, Arunachalam, Somaiya, & Lin, 2012). Since the effects of normalization have been mixed, our suggestion is to apply it only when there is no substantial degradation on classification performance, since it can increase classification efficiency by reducing the number of terms. When in doubt whether to remove numbers or punctuations (or other symbols), our advice is to retain them and apply the dimensionality reduction techniques discussed in the below section on text transformation.

## Text Transformation ($X$)

Text transformation is about representing documents so that they form a suitable input to a classification algorithm. In essence, this comprises imposing structure on a previously unstructured text. Most classification algorithms accept vectors or matrices as input. Thus the most straightforward way is to represent a document as a vector and the corpus as a matrix.

The most common way to transform text is to use the so-called *vector space model* (VSM) where documents are modeled as elements in a vector space (Raghavan & Wong, 1986; Salton, Wong, & Yang, 1975). The features in this representation are the individual terms found in the corpus. This somehow makes sense under the assumption that words are the smallest independently meaningful units of a language. The size of the vector is therefore equal to the size of the vocabulary (i.e., the set of unique terms in a corpus). Hence, we can represent document $j$ as $X_j = (\,x_j^1 \quad x_j^2 \quad \cdots \quad x_j^M\,)$ where $M$ is the size of the vocabulary, and the element $x_j^i$ is the *weight* of term $i$ in document $j$. Weights can be the count of the terms in a document ($x_j^i = TF(j,i)$) or, when using binary weighting, a 1 (presence of a term) or 0 (absence of a term). Applying the transformation to the entire corpus will lead to a *document-by-term matrix* (DTM), where the rows are the documents, the columns are the terms, and the entries are the weights of the terms in each document.

Other weighting options can be derived from basic count weighting. One can take the logarithm of the counts to dampen the effect of highly frequent terms. Here we need to add 1 to the counts so that we avoid taking the logarithm of zero counts. It is also possible to normalize with respect to document length by dividing each count by the maximum term count in a given document. This is to ensure that frequent terms in long documents are not overrepresented. Apart from the weights of the terms in each document, terms can also be weighted with respect to the corpus. Common corpus-based weights include the inverse document frequency (IDF), which assesses the specificity of terms in a corpus (Algarni & Tairan, 2014). Terms that occur in too few (large IDF) or in too many (IDF close to zero) documents have low discriminatory power and are therefore not useful for classification purposes. The formula for IDF is: $IDF(i) = \log\left(\frac{N}{df(i)}\right)$, where $df(i)$ stands for the document frequency of term $i$, that is, the number of documents containing term $i$. Document- and corpus – based weights may also be combined so that that the weights simultaneously reflect the importance of a term in a document and its specificity to the corpus. The most popular combined weight measure is the product of term frequency (TF) and the IDF ($x_j^i = TF(j,i) \times IDF(i)$) (Aizawa, 2003).

Although the VSM ignores word order information, it is popular due to its simplicity and effectiveness. Ignoring word order means losing some information regarding the semantic relationships between words. Also, words alone may not always express true atomic units of meaning. Some researchers improve the VSM by adding adjacent word pairs or trios (*bigrams* and *trigrams*) as features. For example, "new" followed by "york" becomes "new york" in a bigram. Although this incorporates some level of word order information, it also leads to feature explosion thereby increasing noise and redundancy. Also, many bigrams and trigrams do not occur often, thus their global contributions to the classification are negligible and will only contribute to sparsity and computational load. A workaround is to use only the most informative phrases (e.g., frequent phrases; Scott & Matwin, 1999). Strategies for selecting key phrases include the noun phrase (Lewis, 1992) and key phrase (Turney, 1999) extraction algorithms. However, this does add additional complexity in the analysis, which may again not result in a significant improvement in the classification. Studies have consistently shown that using bigrams only marginally improved classification performance and in some cases degraded it, whereas the use of trigrams typically yielded improvement (Dave et al., 2003; Ragas & Koster, 1998). Using syntactic phrases typically does not improve performance much compared to single term features (Moschitti & Basili, 2004; Scott & Matwin, 1999). Thus, the recommendation is to rely on single term features rather than phrases unless there is a strong rationale to use phrases.

Text transformation plays a critical role in determining classification performance. Inevitably some aspects of the text are lost in the transformation phase. Thus, when resulting classification performance is poor, we recommend that the researcher reexamines this step. For example, while term-based features are popular, if performance is poor one could also consider developing features derived from linguistic information (e.g., parts of speech) contained in text (Gonçalves & Quaresma, 2005; Kobayashi et al., 2017; Moschitti & Basili, 2004) or using consecutive characters instead of whole words (e.g., n-grams; Cavnar & Trenkle, 1994).

*Reducing dimensionality.* Even after preprocessing, transformation through VSM is still likely to result in a large feature set. Too large a number of features is undesirable because it may increase computational time and may degrade classification performance, especially when there are many redundant and noisy features (Forman, 2003; Guyon & Elisseeff, 2003; Joachims, 1998). The size of the vector and hence the size of feature set is referred to as the *dimensionality* of the VSM representation. When possible, one should reduce dimensionality either by selectively eliminating features or by creating latent features from existing ones without sacrificing classification performance (Burges, 2010; Fodor, 2002; van der Maaten, Postma, & van den Herik, 2009). A reduced feature set has advantages such as higher efficiency and in some cases, improved classification performance.

One way to eliminate features is to first assign scores to each feature and then remove features by setting a cutoff value. This is called *thresholding* (Lan, Tan, Su, & Lu, 2009; Salton & Buckley, 1988). Weights from the transformation steps are sometimes used to score features. An example is to remove rare terms, that is, terms with high IDF or low DF since they are noninformative for category prediction or not influential in global performance. In some cases, rare terms are noise terms (e.g., misspellings).

Another group of strategies to score features is to make use of class membership information in the training data. These methods are called *supervised scoring methods*. Examples of these methods are mutual information (MI), chi-squared (CHI), Gini index (GI), and information gain (IG; Yang & Pedersen, 1997). Supervised scoring methods are expected to be superior to unsupervised ones (e.g., DF), although in some cases DF thresholding has yielded performance comparable to supervised scoring methods such as CHI and GI (Yang, 1999) and even exceeded the performance of MI.

An alternative to scoring methods is to create latent orthogonal features by combining existing features. Methods that construct new features from existing ones are known as *feature transformation methods.* Techniques include principal component analysis (PCA; Sirbu et al., 2016; Zu, Ohyama, Wakabayashi, & Kimura, 2003), latent semantic analysis (LSA; Landauer et al., 1998), and nonnegative matrix factorization (Zurada, Ensari, Asl, & Chorowski, 2013). These methods construct high level features as a (non)linear combination of the original features with the property that the new features are uncorrelated. They operate on the DTM by applying a matrix factorization method. The text is scored (or projected) on the new features, or factors, and these new features are used in the subsequent analysis. LSA improves upon the VSM through its ability to detect synonymy (Landauer et al., 1998). Words that appear together and load highly on a single factor may be considered to be synonyms.

*Recommendation.* Our recommendation is start with the traditional VSM, that is, transform the documents into vectors using single terms as features. For the unsupervised scoring, compute the DF of each term and filter out terms with very low and very high DF, customarily those terms belonging to the lower 5th and upper 99th percentiles. For the supervised scoring try CHI and IG and for the feature transformation try LSA and nonnegative matrix factorization. Compare the effect on classification performance of the different feature sets generated by the methods and choose the feature set that yields the highest performance (e.g., accuracy). We also suggest to try combining scoring and transformation methods. For example, one can first run CHI and perform LSA on the terms selected by CHI. Note that the quality of the feature set (and that of the representation) is assessed based on its resulting classification performance (Forman, 2003).

For LSA and nonnegative matrix factorization, we need to decide how many dimensions to retain. For LSA, Fernandes, Artífice, and Fonseca (2017) offered this formula as a rough guide $K = N^{\left(\frac{1}{1+\frac{\log(N)}{10}}\right)}$, where $N$ is the size of the corpus, $K$ is the number of dimensions to retain and the logarithm is base 10. For example, if there are 500 documents, then retain approximately 133 latent dimensions. In the case of nonnegative matrix factorization, an upper bound for choosing K is that it must satisfy this inequality $(N + M)K < NM$, where M is the number of original features (Tsuge, Shishibori, Kuroiwa, & Kita, 2001). Hence if there are 500 documents and 1,000 terms, $K$ should not be greater than 333. Of course, one has to experiment with different sizes of dimensionality and select the size that yields the maximum performance. For example, the formula gave 133 dimensions for 500 documents but one may also try experimenting values within $\pm 30$ around 133.

## Application of TC Algorithms ($f$)

The transformed text, usually the original DTM or the dimensionality reduced DTM, serves as input to one or more classification techniques. Most techniques are from the fields of machine learning and statistics. There are three general types of techniques: (a) geometric, (b) probabilistic, and (c) logical (Flach, 2012).

Geometric algorithms assume that the documents can be represented as points in a hyperspace, the dimensions of which are the features. This means that distances between documents and lengths of the documents can be defined as well. In this representation, nearness implies similarity. An example of a geometric classifier is *K-nearest neighbors* in which classification is done by first finding the closest $K$ documents (using a distance measure) from the training data (Jiang, Pang, Wu, & Kuang, 2012) then the majority class of the $K$ closest documents is the class to which the new document is assigned. The parameter $K$ is chosen to be an odd number to prevent ties from occurring. Another geometric classifier is *support vector machines* (Joachims, 1998) in which a hyperplane is

constructed that provides the best separation among the text in each class. The hyperplane is constructed in such a way that it provides the widest separation between the two nearest observations of each class.

Probabilistic algorithms compute a joint probability distribution between the observations (e.g., documents) and their classes. Each document is assumed to be an independent random draw from this joint probability distribution. The key point in this case is to estimate the posterior probability $P(Y_m|X)$. Classification is achieved by identifying the class that yields the maximum posterior probability for a given document. The posterior probability is estimated in two ways. Either one can marginalize the joint distribution $P(X, Y_m)$, or one may compute $P(X|Y_m)$ and $P(Y_m)$ separately and apply Bayes theorem. Both *naive Bayes* (Eyheramendy, Lewis, & Madigan, 2003) and *logistic regression* (J. Zhang, Jin, Yang, & Hauptmann, 2003) are examples of probabilistic algorithms.

The third type of algorithm is the logical classifier, which accomplishes classification by means of logical rules (Dumais, Platt, Heckerman, & Sahami, 1998; Rokach & Maimon, 2005). An example of such a rule in online news categorization is: "If an article contains any of the stemmed terms "vs", "earn", "loss" and not the words "money", "market open", or "tonn" then classify the article under category "earn" (Rullo, Cumbo, & Policicchio, 2007). The rules in logical models are readable and thus facilitate revision, and, if necessary, correction of how the classification works. An example of a logical classifier is a *decision tree* (Rokach & Maimon, 2005).

Naive Bayes and support vector machines are popular choices (Ikonomakis, Kotsiantis, & Tampakas, 2005; Joachims, 1998; Li & Jain, 1998; Sebastiani, 2002). Both can efficiently deal with high dimensionality and data sparsity, though in naive Bayes appropriate smoothing will need to be applied to adjust for terms which are rare in the training data. The method of K-nearest neighbor works well when the amount of training data is large. Both logistic regression and discriminant analysis yield high performance if the features are transformed using LSA. The performance of decision trees has been unsatisfactory. A number of researchers therefore recommend the strategy of training and combining several classifiers to increase classification performance, which is known as ensemble learning (Breiman, 1996; Dietterich, 1997; Dong & Han, 2004; Polikar, 2012). This kind of classification can be achieved in three ways. The first is using a single method and training it on different subsets of the data. Examples include bagging and boosting, which both rely on resampling. Random forest is a combination of bagging and random selection of features that uses decision trees as base learners. Gradient boosted trees, a technique that combines several decision trees, has been shown to significantly increase performance as compared with that of individual decision trees (Ferreira & Figueiredo, 2012). The second is using a single method but varying the training parameters such as, for example, using different initial weights in neural networks (Kolen & Pollack, 1990). The third is using different classification techniques (naive Bayes, decision trees, or SVM; Li & Jain, 1998) and combining their predictions using, for instance, the majority vote.

*Recommendation.* Rather than using a single technique, we suggest applying different methods, by pairing different algorithms and feature sets (including those obtained from feature selection and transformation) and choosing the pair with the lowest error rate. For example, using the DTM matrix, apply SVM, naive Bayes, random forest bagging, and gradient boosted trees. When feature transformation has been applied (e.g., LSA and nonnegative matrix factorization), use logistic regression or discriminant analysis. When the training data are large (e.g., hundreds of thousands of cases), use K-nearest neighbors. Rule-based algorithms are seldom used in TC, however, if readability and efficiency are desired in a classifier, then these can be trialed as well.

| | | Predicted Classes | |
|---|---|---|---|
| | | Positive | Negative |
| True Classes | Positive | TP | FN |
| | Negative | FP | TN |

**Figure 1.** Confusion matrix as a reference to compute the evaluation measures. *Note*: FN = false negative; FP = false positive; TN = true negative; TP = true positive.

## Evaluation Measures

Crucial to any classification task is the assessment of the performance of classifiers using evaluation measures (Powers, 2011; Yang, 1999). These measures indicate whether a classifier models the relationship between features and class membership well, and may thus be used to indicate the extent to which the classifier is able to emulate a human coder. The most straightforward evaluation measure is the accuracy measure, which is calculated as the proportion of correct classifications. Accuracy ranges from 0 to 1 (or 0 to 100 when expressed as a percentage). The higher the accuracy the better the classifier (1 corresponds to perfect classification). However, in case of imbalanced classification (i.e., when there is one class with only a few documents) and/or unequal costs of misclassification, accuracy may not be appropriate. An example is detecting career shocks (cf. Seibert, Kraimer, Holtom & Pierotti, 2013) in job forums. Since it is likely that only a small fraction of these postings pertain to career shocks (suppose .05), a classifier can still have a high accuracy (equal to .95) even if that classifier classifies all discussion as containing no career shocks content.

Alternative measures to accuracy are precision, recall, F-measure (Powers, 2011), specificity, breakeven point, and balanced accuracy (Ogura, Amano, & Kondo, 2011). In binary classification, classes are commonly referred to as positive and negative. Classifiers aim to correctly identify observations in the positive class. A summary table which can be used as a reference for computing these measures is presented in Figure 1. The entries of the table are as follows: TP stands for true positives, TN for true negatives, FP for false positives (i.e., negative cases incorrectly classified into the positive class), and FN for false negatives (i.e., positive cases incorrectly classified into the negative class). Hence the five evaluation measures are computed as follows: $precision = \frac{TP}{TP+FP}$, $recall = \frac{TP}{TP+FN}$, $specificity = \frac{TN}{TN+FP}$, $F - measure = \frac{2\times recall\times precision}{recall+precision}$, and $Bal.Accu = \left(\frac{TP}{TP+FP} + \frac{TN}{TN+FN}\right)/2$.

The breakeven point is the value at which $precision = recall$. F-measure and balanced accuracy are generally to be preferred in case of imbalanced classification, because they aggregate the more basic evaluation measures.

Evaluation measures are useful to compare the performance of several classifiers (Alpaydin, 2014). Thus, one can probe different combinations of feature sets and classification techniques to determine the best combination (i.e., the one which gives the optimal value for the evaluation measure). Apart from classification performance, one can also take the parsimony of the trained classifier into account by examining the relative size of the different feature sets, since they determine the complexity of the trained classifier. In line with Occam's razor, when two classifiers have the same classification performance, the one with the lower number of features is to be preferred (Shreve, Schneider, & Soysal, 2011).

Evaluation measures are computed from the labeled data. It is not advisable to use all labeled data to train the classifier since this might result in *overfitting* which is the case when the classifier is good at classifying the observations in the training data but performs poorly on new data. Hence, part of the labeled data should be set aside for evaluation so that we can assess the degree to which the classifier is able to predict accurately in data that were not used for training.

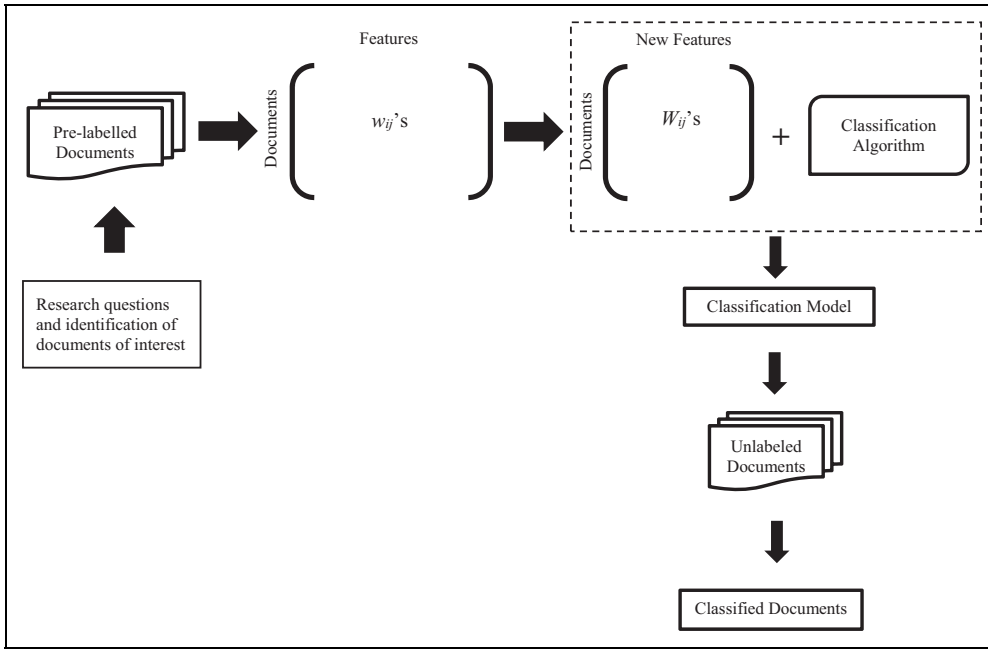



**Figure 2.** Diagrammatic depiction of the text classification process.

Cross-validation can be applied by computing not only one value for the evaluation measure but several values corresponding to different splits of the data. A systematic strategy to evaluate a classifier is to use $k$-fold cross-validation (Kohavi, 1995). This method splits the labeled dataset into $k$ parts. A classifier is trained using $k - 1$ parts and evaluated on the remaining part. This is repeated until each of the $k$ parts has been used as test data. Thus for $k$ equals 10, there are 10 partitions of the labeled data and corresponding 10 values for a given measure, the final estimate is just the average of the 10 values. Another strategy is called bootstrapping, which is accomplished by computing an average of the evaluation measures for $N$ bootstrap samples of the data (sampling with replacement).

*Recommendation.* Since accuracy may give misleading results when classes are imbalanced we recommend using measures sensitive to this, such as F-measure or balanced accuracy (Powers, 2011). For the systematic evaluation of the classifier we advise using $k$-fold cross validation and setting K to 5 or 10 when data are large, as this ensures sufficient data for the training. For smaller data sets, such as fewer than 100 documents, we suggest bootstrapping or choosing a higher K for cross-validation.

## Model Validity

Figure 2 illustrates that a classification model consists of features and the generic classification algorithm (Domingos, 2012). Thus the validity of the classification model depends both on the choice of features and the algorithm.

Many TC applications use the set of unique words as the feature set (i.e., VSM). For organizational researchers this way of specifying the initial set of features may seem counterintuitive since features are constructed in an ad hoc and inductive manner, that is, without reference to theory.

Indeed, specifying the initial set of features, scoring features, transforming features, evaluating features, and modifying the set of features in light of the evaluation constitutes a data-driven approach to feature construction and selection (Guyon, Gunn, Nikravesh, & Zadeh, 2008). The validity of the features is ultimately judged in terms of the classification performance of the resulting classification model. But this does not mean that researchers should abandon theory based approaches. If there is prior knowledge or theory that supports the choice of features then this can be incorporated (Liu & Motoda, 1998). Theory can also be used as a basis for assigning scores to features such as using theory to rank features according to importance. Our recommendation, however, would be to have theory complement, as opposed to restrict, feature construction, because powerful features (that may even be relevant to subsequent theory building and refinement) may emerge inductively.

The second component, the classification algorithm, models the relationship between features and class membership. Similar to the features, the validity of the algorithm is ultimately determined from the classification performance and is also for the most part data driven. The validity of both the features and the classification algorithm establishes the validity of the classification model.

A useful strategy to further assess the validity of the classification model is to compare the classifications made by the model with the classification of an independent (group of) human expert(s). Usually agreement between the model and the human expert(s) is quantified using measures of concordance or measures of how close the classification of the two correspond to one another (such as Cohen's kappa for interrater agreement where one "rater" is the classifier). Using expert knowledge, labels can also be checked against standards. For example, in job task extraction from a specific set of job vacancies one can check with experts or job incumbents to verify whether the extracted tasks correspond to those tasks actually carried out on the job and whether specific types of tasks are under or over represented.

Once model validity is established one may start applying the classification model to unlabeled data. However, the model will still need to be reevaluated from time to time. When the performance drops below an acceptability threshold, there are four possible solutions: (a) add more features or change existing features, (b) try other classification algorithms, (c) do both, and/or (d) collect more data or label additional observations.

## Other Issues in TC

In this section we discuss how to deal with multiclass classification, where there is an increased likelihood of classes being imbalanced, and provide some suggestions on determining training size and what to do when obtaining labeled data is both expensive and difficult.

*Multiclass classification.* Multiclass classification pertains to dealing with more than 2 categories. The preprocessing and representation parts are the same as in the binary case. The only changes are in the choices of supervised feature selection techniques, classification techniques and evaluation measures. Most supervised feature selection techniques can be easily generalized to more than 2 categories. For example, when calculating CHI, we just need to add an extra column to the two-way contingency table. Most techniques for classification we discussed previously have been extended to multiclass classification. For example, techniques suited for binary classification problems (e.g., SVM) are extended to the multiclass case by breaking the multiclass problem into several binary classification problems in either one-against all or one-against-one approaches. In the former approach we build binary classifiers by taking each category as the positive class and merging the others into the negative class. Hence, if there are $K$ categories, then we build $K$ binary classifiers. For the latter approach, we construct a binary classifier for each pair of categories resulting to $\frac{K(K-1)}{2}$ classifiers. Since several classifiers are built, and thus there are several outputs, final category

membership is obtained by choosing the category with the largest value for the decision function for the one-against-all case or by a voting approach for the one-against-one case (Hsu & Lin, 2002).

The four evaluation measures can also be extended to classifications with more than two classes by computing these measures per category, the same as in one-against-all, and averaging the results. An example is the extension of F-measure called the *macro F-measure* which is obtained by computing the F-measure of each category, and then averaging them.

*Imbalanced classification.* By and large, in binary classification, when the number of observations in one class represents less than 20% of the total number of observations then the data can be seen as imbalanced. The main danger of imbalanced classification is that we may train a classifier with a high accuracy even if it fails to correctly classify the observations in the minority class. In some cases, we are more interested in detecting the observations in the minority class. At the same time however, we also want to avoid many false detections.

Obvious fixes are to label more observations until the classes are balanced as was done by Holton (2009), or by disregarding some observations in the majority class. In cases where classification problems are inherently imbalanced and labeling additional data is costly and difficult, another approach is to oversample the minority class or to undersample the majority class during classifier training and evaluation. A strategy called the synthetic minority oversampling technique (SMOTE) is based on oversampling but instead of selecting existing observations in the minority class it creates synthetic samples to increase the number of observations in the minority class (Chawla, Bowyer, Hall, & Kegelmeyer, 2002). Preprocessing and representation remain the same as in balanced classes. The parts that make use of class membership need to be adjusted for imbalanced data.

There are options for supervised dimensionality reduction for imbalanced classification such as those provided by Ogura et al. (2011). For the choice of classification techniques, those discussed previously can be used with minor variations such as adjusting the costs of misclassification, which is known as cost-sensitive classification (Elkan, 2001). Traditional techniques apply equal costs of misclassification to all categories, whereas, for cost-sensitive classification we can assign large cost for incorrect classification of observations in the minority class. For the choice of evaluation measures, we suggest using the weighted F-measure or balanced accuracy. One last suggestion is to treat imbalanced classification as an anomaly or outlier detection problem where the observations in the minority class are the outliers (Chandola, Banerjee, & Kumar, 2009).

*Size of the training data.* A practical question that often arises is how many documents one should label to ensure a valid classifier. The size of the training dataset depends on many considerations such as the cost and limitations associated with acquiring prelabeled documents (e.g., ethical and legal impediments) and the kind of learning framework we are using. In the probably approximately correct (PAC) learning framework, which is perhaps the most popular framework for learning concepts (such as the concept of spam emails or party affiliation) training size is determined by the type of classification technique, the representation size, the maximum error rate one is willing to tolerate, and the probability of not exceeding the maximum error rate. Under the PAC learning framework, formulae have been developed to determine the lower bound for the training size, an example being the one by Goldman (2010): $\Omega\left(\frac{1}{\varepsilon}\log_2\frac{1}{\delta}+\frac{\mathrm{VCD}(C)}{\varepsilon}\right)$, where $\epsilon$ is the maximum error rate, $1-\delta$ indicates the probability that the error will not exceed $\varepsilon$, and $\mathrm{VCD}(C)$ of the classifier $C$. VCD stands for Vapnik-Chervonenski dimension of the classifier $C$ which can be interpreted as the expressive power of the classifier which depends on the representation size and the form of the classifier (e.g., axis parallel rectangle, closed sets, or half-spaces). As an illustration suppose we want to learn the concept of positive sentiment from English text, we then represent each document as a vector of 50,000 dimensions (number of commonly used English words) and our classification

technique constructs a hyperplane that separates positive and negative observations (e.g., SVM using ordinary dot product as kernel). If we want to ensure with probability 0.99 that the error rate will not exceed 0.01, then the minimum training size is: $\frac{1}{0.01}\log_2\frac{1}{0.01} + \frac{50001}{0.01} = 5000764$. This means we would need at least 5 million documents. Here we calculated the VCD of $C$ using the formula $d + 1$, where $d$ is the dimensionality of the representation, since we consider classifiers that construct hyperplane boundaries (half-space) in 50,000 dimensions. Of course, in practice dimensionality reduction can be applied still get adequate representation. If one would manage to reduce dimensionality to 200 then the lower bound for the training size is dramatically reduced to 20,765. We can tweak this lower bound by adjusting the other parameters.

Although formulae provide theoretical guarantees, determining training size is largely empirically driven and involves a good deal of training, evaluation, and validation. To give readers an idea of training sizes as typically found in practice, Table 1 provides information about the training data sizes for some existing TC studies.

*Suggestions when labeled data are scarce.* In many classification problems, labeled data are costly or difficult to obtain. Fortunately, even in this case, principled approaches can be applied. In practice, unlabeled data are plentiful and we can apply techniques to make use of the structure and patterns in the unlabeled data. This approach of using unlabeled data in classification is called *semisupervised classification* (Zhu, 2005). Various assumptions are made to make semisupervised classification feasible. Examples are the smoothness assumption which says that observations near each other are likely to share the same label, and the cluster assumption which states that if the observations form clusters then observations in the same cluster are likely to share the same label (Zhu, 2005).

Another approach is to use classification output to help us determine which observations to label. In this way, we take a targeted approach to labeling by labeling those observations which are most likely to generate better classifiers. This is called *active learning* in the machine learning literature (Settles, 2010). Active learning is made possible because some classifiers give membership probabilities or confidence rather than a single decision as to whether to assign to one class or not. For example, if a classifier assigns equal membership probabilities to all categories for a new observation then we call an expert to label the new observation. For a review of active learning techniques we refer the reader to Fu, Zhu, and Li (2013).

## Tutorial

We developed the following tutorial to provide a concrete treatment of TC. Here we demonstrate TC using actual data and codes. Our intended audience are researchers who have little or no experience with TC. This tutorial is a scaled down version of our work on using TC to automatically extract job tasks from job vacancies. Our objective is to build a classifier that automatically classifies sentences into task or nontask categories. The sentences were obtained from German language nursing job vacancies.

We set out to automate the process of classification because one can then deal with huge numbers (i.e., millions) of vacancies. The output of the text classifier can be used as input to other research or tasks such as job analysis or the development of tools to facilitate personnel decision making. We used the R software since it has many ready-to-use facilities that automate most TC procedures. We provide the R annotated scripts and data to run each procedure. Both codes and data can be downloaded as a Zip file from Github; the URL is https://github.com/vkobayashi/textclassificationtutorial. The naming of R scripts are in the following format: CodeListing (CL) <number>.R and in this tutorial we referenced them as CL <number>. Thus, CL 1 refers to the script CodeListing_1.R. Note that the CL files contain detailed descriptions of each command, and that each command should be run sequentially.

**Table 1.** Training Sizes, Number of Categories, Evaluation Measures, and Evaluation Procedures Used in Various Text Classification Studies.

| Citations | Subject Matter | Training Size | Number of Categories | Evaluation Measure | Evaluation Procedure |
|---|---|---|---|---|---|
| Phan, Nguyen, & Horiguchi, 2008 | Domain disambiguation for web search results | 12,340 | 8 | Accuracy | Fivefold cross validation (CV) |
| Moschitti & Basili, 2004; Phan et al., 2008 | Disease classification for medical abstracts | 28,145 | 5 | Accuracy | Fivefold CV |
| Vo & Ock, 2015 | Titles of scientific documents | 8,100 | 6 | Accuracy & F-measure | Fivefold CV |
| Khoo et al., 2006 | Response emails of operators to customers | 1,486 | 14 | Accuracy & F-measure | Tenfold CV |
| Chen, Huang, Tian, & Qu, 2009; Li & Jain, 1998; Moschitti & Basili, 2004; Ogura et al., 2011; Scott & Matwin, 1999; Song et al., 2005; Toman et al., 2006; Torunoğlu et al., 2011; Uysal & Gunal, 2014; Yang & Pedersen, 1997; W. Zhang, Yoshida, & Tang, 2008 | News items | 764-19,997 | 4-93 | Accuracy, micro averaged breakeven points, F-measure, recall, precision, breakeven point | 1 training and 2 test, single train-test, 20 splits with intercorpus evaluation, fourfold CV, tenfold CV, 20 splits |
| Atteveldt et al., 2008 | Sentiment analysis of actor-issue relationship | 5,348 | 2 | F-measure | Did not mention |
| Dave et al., 2003 | Product reviews sentiments | 31,574 | 7 | Accuracy | 2 test |
| Scott & Matwin, 1999 | Song lyrics | 6,499 | 33 | Micro-averaged break even points | Single-train test |
| Chen et al., 2009 | Chinese news texts | 2,816 | 10 | Accuracy & F-measure | Single-train test |
| Kanaan et al., 2009 | Arabic news documents | 1,445 | 9 | F-measure | Fourfold CV |
| Ragas & Koster, 1998 | Dutch documents | 1,436 | 4 | Accuracy | Single train-test |
| Méndez et al., 2006; Panigrahi, 2012; Uysal & Gunal, 2014; Youn & McLeod, 2007 | Emails | 400-9,332 | 2 | F-measure | Single train-test |

*(continued)*

**Table 1.** (continued)

| Citations | Subject Matter | Training Size | Number of Categories | Evaluation Measure | Evaluation Procedure |
|---|---|---|---|---|---|
| Torunoğlu et al., 2011; Uysal & Gunal, 2014 | Turkish news items | 1,150-99,021 | 5-10 | F-measure | |
| Moschitti & Basili, 2004 | Italian news items | 16,000 | 8 | F-measure & breakeven point | 20 splits with intercorpus validation |
| Toman et al., 2006 | Czech news items | 8,000 | 5 | F-measure | Fourfold CV |
| Thelwall, Buckley, Paltoglou, Cai, & Kappas, 2010 | Cyberspace comment sentiment analysis | 1,041 | 5 | Accuracy | Tenfold CV |
| Holton, 2009 | Disgruntled employee communications | 80 | 2 | Accuracy | Single train—test varying proportion |
| Shen et al., 2013 | Personality from emails (this is a multilabel classification problem) | 114,907 | 3 categories per personality | Accuracy | Tenfold CV and single train-test |

All the scripts were tested and are expected to work on any computer (PC or Mac) with R, RStudio, and the required libraries installed. However, basic knowledge including how to start R, open R projects, run R commands, and install packages in RStudio are needed to run and understand the codes. For those new to R we recommend following an introductory R tutorial (see, for example, DataCamp [www.datacamp.com/courses/free-introduction-to-r] or tutorialspoint [www.tutorialspoint.com/index.htm] for free R tutorials).

This tutorial covers each of the previously enumerated TC steps in sequence. For each step we first explain the input, elaborate the process, and provide the output, which is often the input for the subsequent step. Table 2 provides a summary of the input, process, and output for each step in this tutorial. Finally, after downloading the codes and data, open the text_classification_tutorial.Rproj file. The reader should then run the codes for every step as we go along, so as to be able to examine the input and the corresponding output.

## Preparing Text

The input for this step consists of the raw German job vacancies. These vacancies were obtained from Monsterboard (www.monsterboard.nl). Since the vacancies are webpages, they are in hypertext markup language (HTML), the standard markup language for representing content in web documents (Graham, 1995). Apart from the relevant text (i.e., content), raw HTML pages also contain elements used for layout. Therefore, a technique known as HTML parsing is used to separate the content from the layout.

In R, parsing HTML pages can be done using the **XML** package. This package contains two functions, namely, `htmlTreeParse()` that parses HTML documents and `xpathSApply()` that extracts specific content from parsed HTML documents. CL 1 (see the annotations in the file for further details as to what each command does), installs and loads the **XML** package, and applies the `htmlTreeParse()` and `xpathSApply()` functions. In addition, the contents of the HTML file *sample_nursing_vacancy.html* in the folder *data* are imported as a string object and stored in the variable `htmlfile`. Subsequently, this variable is provided as an argument to the `htmlTreeParse()` function. The parsed content is then stored in the variable `rawpagehtml`, which in turn is the `doc` argument to the `xpathSApply()` function which searches for the tags in the text that we are interested in. In our case this text can be found in the `div` tag of the class `content`. Tags are keywords surrounded by angle brackets (e.g., `<div>` and `</div>`). The `xmlValue` in the `xpathSApply()` function means that we are obtaining the content of the HTML element between the corresponding tags. Finally the `writeLines()` function writes the text content to a text file named *sample_nursing_vacancy.txt* (in the folder parsed).

To extract text from several HTML files, the codes in CL 1 are put in a loop in CL 2. The function `htmlfileparser()` in CL 2 accepts two arguments and applies the procedures in CL 1 to each HTML file in a particular folder. The first argument is the name of the folder and the second argument is the name of the destination folder where the extracted text content is to be written. Supposing these html files are in the folder *vacancypages* and the extracted text content is to be saved in the folder *parsedvacancies,* these are the arguments we provide to `htmlfileparser()`. Expectedly, the number of text files generated corresponds to the number of HTML files, provided that all HTML files are well-defined (e.g., correct formatting). The text files comprise the output for this step.

## Preprocessing Text

The preprocessing step consists of two stages. The first identifies sentences in the vacancies, since the sentence is our unit of analysis, and the second applies text preprocessing operations on the sentences. We used sentences as our unit of analysis since our assumption is that the sentence is the

**Table 2.** Text Classification Based on the Input-Process-Output Approach.

| | Text Preprocessing | | | | | | |
|---|---|---|---|---|---|---|---|
| | Text Preparation | Text Cleaning | Text Transformation | Dimensionality Reduction | Classification | Evaluation | Validation |
| Input | Raw html files | Output from text preparation | Output from text cleaning | Document-by-term matrix | Output from dimensionality reduction | Classification model, test data, and an evaluation measure | Classification from the model |
| Process | Parsing, sentence segmentation | Punctuation, number, and stopword removal, lower case transformation | Word tokenization, constructing the document-by-term matrix where the words are the features and the entries are raw frequencies of the words in each document | Latent semantic analysis and/or supervised scoring methods for feature selection | Apply classification algorithms such as naive Bayes, support vector machines, or random forest | Classify the documents in test data and compare with the actual labels; calculate the value of the evaluation measure | Compute classification performance using an independent validation data set or compare the classification to the classification of domain experts |
| Output | Raw text file (one sentence per line) | Raw text file sentences where all letters are in lower cases and without punctuation, number and stopwords | Document-by-term matrix | Matrix where the columns are the new set of features or the reduced document-by-term matrix | Classification model | Value for the evaluation | Measure of agreement (one can quantify the agreement through the use existing evaluation measure) |

right resolution level to detect job task information. We did not use the vacancy as our unit of analysis since a vacancy may contain more than one task. In fact if we chose to treat the vacancy as the unit of analysis it would still be important to identify which of the sentences contain task information. Another reason to select sentence as the unit of analysis is to minimize variance in document length. Input for the first stage are the text files generated from the previous step, and the output sentences from this stage serve as input to the second stage. CL 3 contains functions that can detect sentences from the parsed HTML file in the previous section (i.e., *sample_nursing_vacancy.html*).

The code loads the `openNLP` package. This package contains functions that run many popular natural language processing (NLP) routines including a sentence segmentation algorithm for the German language. Although the German sentence segmenter in `openNLP` generally works well, at times it may fail. Examining such failures in the output can provide ideas for the inclusion of new arguments in the algorithm. For example, if the segmenter encounters the words bzw. and inkl. (which are abbreviations of "or" and "including" respectively in German) then the algorithm will treat the next word as the start of a new sentence. This is because the algorithm has a rule that when there is a space after a period the next word is the start of a new sentence. To adjust for these and other similar cases, we created a wrapper function named `sent_tokens()`. Another function `sent_split()`searches for matches of the provided pattern within the string and when a match is found it separates the two sentences at this match. For example, some vacancies use bullet points or symbols such as "|" to enumerate tasks or responsibilities. To separate these items we supply the symbols as arguments to the function. Finally, once the sentences are identified the code writes the sentences to a text file where one line corresponds to one sentence.

For multiple text files, the codes should again be run in a loop. One large text file will then be generated containing the sentences from all parsed vacancies. Since we put all sentences from all vacancies in a single file, we attached the names of the corresponding text vacancy files to the sentences to facilitate tracing back the source vacancy of each sentence. Thus, the resulting text file containing the sentences has two columns: the first column contains the file names of the vacancies from which the sentences in the second column were derived.

After applying sentence segmentation on the parsed vacancy in *sample_nursing_vacancy.txt,* the sentences are written to the file *sentencelines_nursing_vacancy.txt* located in the folder *sentences_from_sample_vacancy*. The next task is to import the sentences into R so that additional preprocessing (e.g., text cleaning) can be performed. Other preprocessing steps that may be applied are lower case transformation, punctuation removal, number removal, stopword removal, and stemming. For this we use the `tm` package in R. This package automatically applies word tokenization, so we do not need to create separate commands for that.

The sentences are imported as a data frame in R (see CL 4). Since the sentence is our unit of analysis, hereafter we refer to these sentences as documents. The first column is temporarily ignored since it contains only the names of the vacancy files. Since the sentences are now stored in a vector (in the second column of the data frame), the VectorSource() function is used. The source determines where to find the documents. In this case the documents are in mysentences[,2]. If the documents are stored in another source, for example in a directory rather than in a vector, one can use DirSource(). For a list of supported sources, invoke the function getSources(). Once the source has been set, the next step is to create a corpus from this source using the VCorpus() function. In the tm package, corpus is the main structure for managing documents. Several preprocessing procedures can be applied to the documents once collected in the corpus. Many popular preprocessing procedures are available in this package. Apart from the existing procedures, users can also specify their own via user-defined functions. The procedures we applied are encapsulated in the `transformCorpus()`function. They include number, punctuation, and extra whitespace removal, and lower case conversion. We did not apply stemming since previous work recommends not to use

stemming for short documents (H.-F. Yu et al., 2012). The output consists of the cleaned sentences in the corpus with numbers, punctuation, and superfluous whitespaces removed.

## Text Transformation

CodeListing 5 details the structural transformation of the documents. The input in this step is the output from the preceding step (i.e., the cleaned sentences in the training data). To quantify text characteristics, we use the VSM because this is the simplest and perhaps most straightforward approach to quantify text and thus forms an appropriate starting point in the application of TC (Frakes & Baeza-Yates, 1992; Salton et al., 1975). For this transformation, the `DocumentTermMatrix()` of the `tm` package has a function that may be used to build features based on the individual words in the corpus.

The `DocumentTermMatrix()` function transforms a corpus into a matrix where the rows are the documents, the columns are features, and the entries are the weights of the features in each document. The default behavior of the `DocumentTermMatrix()` function is to ignore terms with less than 3 characters. Hence, it is possible that some rows consist entirely of 0's because after preprocessing it may be the case that in some sentences all remaining terms have less than 3 characters. The output in this step is the constructed DTM. This matrix is then used as a basis for further analysis. We can further manipulate the DTM, for instance by adjusting the weights.

We mentioned previously that for word features one can use raw counts as weights. The idea of using raw counts is that the higher the count of a term in a document the more important it is in that document. The `DocumentTermmatrix()` function uses the raw count as the default weighting option. One can specify other weights through the `weighting` option of the `control` argument. To take into account documents sizes, for example, we can apply a normalization to the weights although in this case it is not an issue because sentences are short.

Let us assign a "weight" to a feature that reflects its importance with respect to the entire corpus using the DF. Another useful feature of DF is that it provides us with an idea of what the corpus is about. For our example the word with the highest DF (excluding stopwords) is `plege` (which translates to "care") which makes sense because nursing is about the provision of care. Terms that are extremely common are not useful for classification.

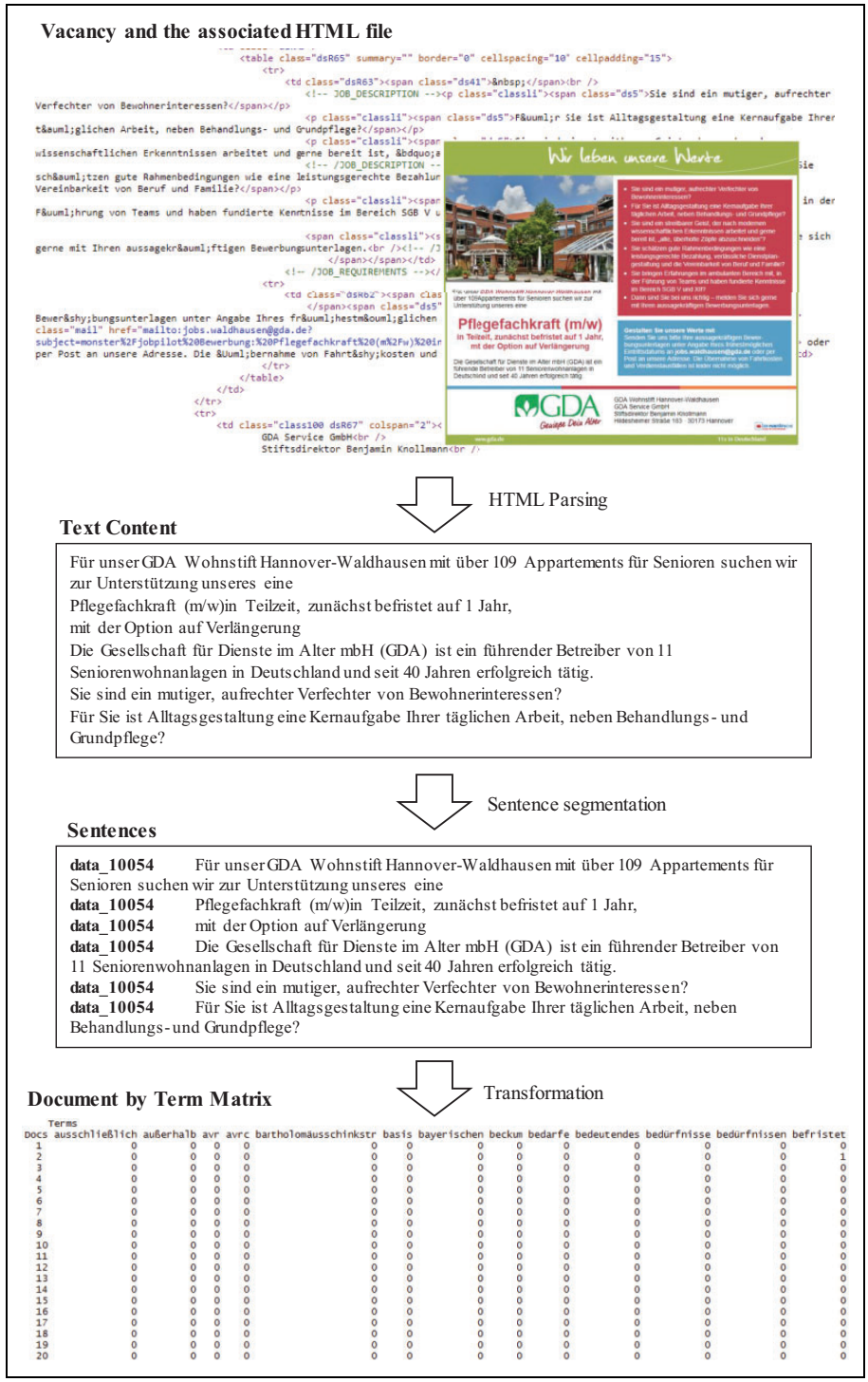
Another common text analysis strategy is to find keywords in documents. The keywords may be used as a heuristic to determine the most likely topic in each document. For this we can use the TF-IDF measure. The keyword for each document is the word with the maximum TF-IDF weight (ties are resolved through random selection). The codes in CL 6 compute the keyword for each document. For example, the German keyword for Document 4 is `aufgabenschwerpunkte` which translates in English to "task focal points."

The final DTM can be used as input to dimensionality reduction techniques or directly to the classification algorithms. The process from text preprocessing to text transformation culminated in the DTM that is depicted in Figure 3.

## Dimensionality Reduction

Before running classification algorithms on the data, we first investigate which among the features are likely most useful for classification. Since the initial features were selected in an ad hoc manner, that is, without reference to specific background knowledge or theory, it may be possible that some of the features are irrelevant. In this case, we applied dimensionality reduction to the DTM.

LSA is a commonly applied to reduce the size of the feature set (Landauer et al., 1998). The output of LSA yields new dimensions which reveal underlying patterns in the original features. The new features can be interpreted as new terms that summarize the contextual similarity of the original




| Docs | ausschließlich | außerhalb | avr | avrc | bartholomäusschinkstr | basis | bayerischen | beckum | bedarfe | bedeutendes | bedürfnisse | bedürfnissen | befristet |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 3.** Illustration of text preprocessing from raw HTML file to document-by-term matrix.

terms. Thus, LSA partly addresses issues of synonymy and in some circumstances, polysemy (i.e., when a single meaning of a word is used predominantly in a corpus). In R, the `lsa` package contains a function that runs LSA.

To illustrate LSA we need additional vacancies. For illustrative purposes we used 11 job vacancies (see the *parsedvacancies* folder). We applied sentence segmentation to all the vacancies and obtained a text file containing 425 sentences that were extracted from the 11 vacancies (see the *sentences_from_vacancies* folder). After applying preprocessing and transforming the sentences in *sentenceVacancies.txt* into a DTM, we obtained 1079 features and retained 422 sentences. We selected all sentences and ran LSA on the transposed DTM (i.e., the term-by-document matrix; see CL7). We applied normalization to the term frequencies to minimize the effect of longer sentences.

Documents and terms are projected onto the constructed LSA space in the `projdocterms` matrix. The entries in this matrix are readjustments of the original entries in the term-by-document matrix. The readjustments take into account patterns of co-occurrence between terms. Hence, terms which often occur together will roughly have the same values in documents where they are expected to appear. We can apply the cosine measure to identify similar terms. Similarity is interpreted in terms of having the same pattern of occurrence. For example, terms which have the same pattern of occurrence with `sicherstellung` can be found by running the corresponding commands in CL 8.

The German word `sicherstellung` (which means "to guarantee" or "to make sure" in English) is found to be contextually similar to `patientenversorgunng` (patient care) and `reibungslosen` (smooth or trouble-free) because these two words appeared together with `sicherstellung` (to guarantee) in the selected documents. Another interesting property of LSA is that it can uncover similarity between two terms even though the two terms may never be found to co-occur in a single document. Consider for example the word `koordinierung` (coordination), we find that `kooperative` (cooperative) is a term with which it is associated even though there is not one document in the corpus in which the two terms co-occur. This happens because both terms are found to co-occur with `zusammenarbeit` (collaboration), thus when either one of the terms occurs then LSA expects that the other should also be present. This is the way LSA addresses the issue of synonymy and polysemy. One can also find the correlation among documents and among terms by running the corresponding commands in CL 8.

Since our aim is to reduce dimensionality, we project the documents to the new dimensions. This is accomplished through the corresponding codes in CL 8. From the LSA, we obtain a total of 107 new dimensions from the original 1,079 features. It is usually not easy to attach natural language interpretations to the new dimensions. In some scenarios, we can interpret the new dimension by examining the scaled coefficients of the terms on the new dimensions (much like in PCA). Terms with higher loadings on a dimension have a greater impact on that dimension. Figure 4 visualizes the terms with high numerical coefficients on the first 6 LSA dimensions (see CL 8 for the relevant code). Here we distinguish between terms found to occur in a task sentence (red) or not (blue). In this way, an indication is provided of which dimensions are indicative for each class (note that distinguishing between tasks and nontasks requires the training data, which is discussed in greater detail below).

Another approach that we could try is to downsize the feature set by eliminating those features that are not (or less) relevant. Such techniques are collectively called filter methods (Guyon & Elisseeff, 2003). They work by assigning scores to features and setting a threshold whereby features having scores below the threshold are filtered out. Both the DF and IDF can be used as scoring methods. However, one main disadvantage of DF and IDF is that they do not use class membership information in the training data. Including class membership (i.e., through supervised scoring methods) ought to be preferred, as it capitalizes on the discriminatory potential of features (Lan et al., 2009).

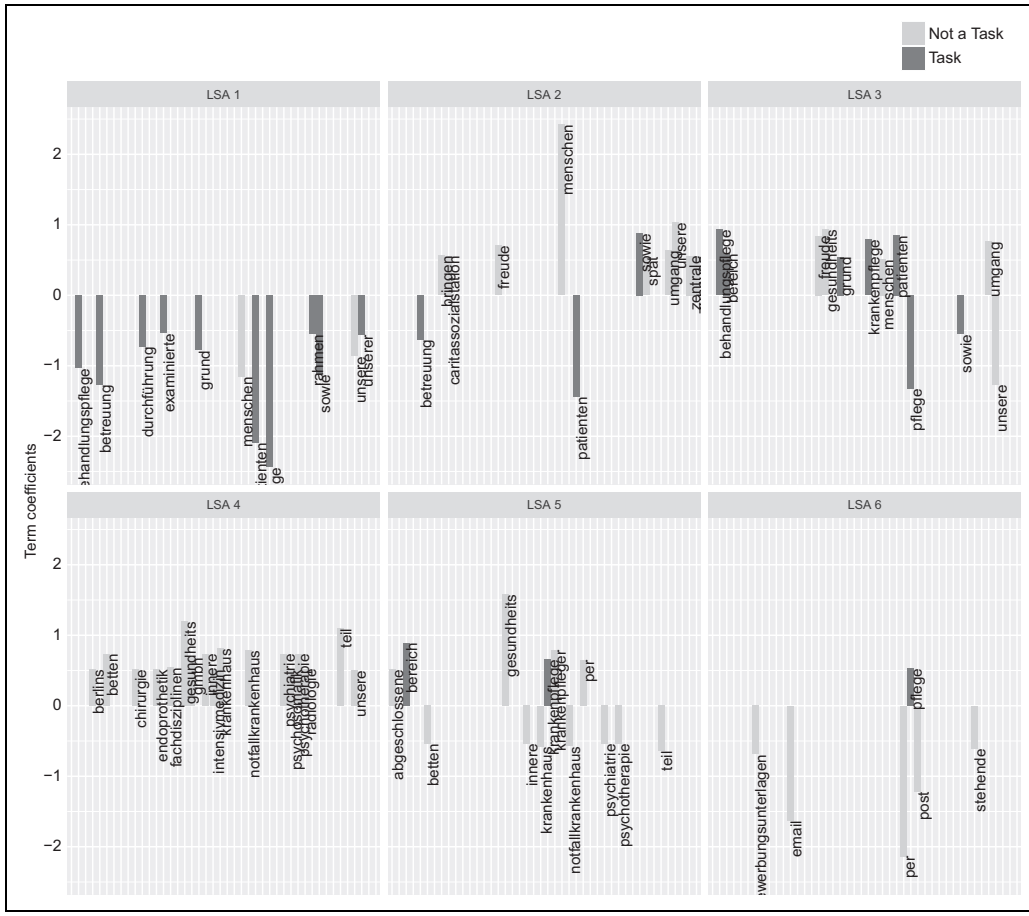



**Figure 4.** Loadings of the terms on the first 6 LSA dimensions using 422 sentences from 11 vacancies.

For supervised scoring methods, we need to rely on the labels of the training data. In this example, the labels are whether a sentence expresses task information (1) or not (0). These labels were obtained by having experts manually label each sentence. For our example, experts manually assigned labels to the 425 sentences. We applied three scoring methods, namely, Information Gain, Gain ratio, and Symmetric Uncertainty (see CL 12). Due to the limited number of labeled documents, these scoring methods yielded less than optimal results. However, they still managed to detect one feature that may be useful for identifying the class of task sentences, that is, the word `zusammenarbeit` (collaboration), as this word most often occurred in task sentences. The output from this step is a column-reduced matrix that is either the reduced version of the DTM or the matrix with the new dimensions. In our example we applied LSA and the output is a matrix in which the columns are the LSA dimensions.

## *Classification*

The reduced matrix from the preceding section can be used as input for classification algorithms. The output from this step is a classification model which we can then use to automatically classify sentences in new vacancies. We have mentioned earlier that reducing dimensionality is an

empirically driven decision rather than one which is guided by specific rules of thumb. Thus, we will test whether the new dimensions lead to improvement in performance as compared to the original set by running separate classification algorithms, namely support vector machines (SVMs), naive Bayes, and random forest, on each set. These three have been shown to work well on text data (Dong & Han, 2004; Eyheramendy et al., 2003; Joachims, 1998).

Accuracy is not a good performance metric in this case since the proportion of task sentences in our example data is low (less than 10%). The baseline accuracy (computed from the model which assigns all sentences to the dominant class), would be 90% which is high, and thus difficult to improve upon. More suitable performance metrics are the F-measure (Ogura et al., 2011; Powers, 2011) and balanced accuracy (Brodersen, Ong, Stephan, & Buhmann, 2010). We use these two measures here since the main focus is on the correct classification of task sentences and we also want to control for misclassifications (nontask sentences put into the task class or task sentences put into the nontask class).

In assessing the generalizability of the classifiers, we employed 10 times 10 fold cross-validation. We repeated 10 fold cross-validation 10 times because of the limited training data. We use one part of the data to train a classifier and test its performance by applying the classifier on the remaining part and computing the F-measure and balanced accuracy. For the 10 times 10 fold cross-validation, we performed 100 runs for each classifier using the reduced and original feature sets. Hence, for the example we ran about 600 trainings since we trained 6 classifiers in total. All performance results reported are computed using the test sets (see CL 10).

From the results we see how classification performance varies across the choice of features, classification algorithms, and evaluation measures. Figure 5 presents the results of the cross-validation. Based on the F-measure, random forest yielded the best performance using the LSA reduced feature set. The highest F-measure obtained is 1.00 and the highest average F-measure is 0.40 both from random forest. SVM and naive Bayes have roughly the same performance. This suggests that among the three classifiers random forest is the best classifier using the LSA reduced feature set, and F-measure as the evaluation metric. If we favor the correct detection of task sentences and we want a relatively small dimensionality then random forest should thus be favored over the other methods. For the case of using the original features, SVM and random forest exhibit comparable performance. Hence, when using F-measure and the original feature set, either SVM or random forest would be the preferred classifier. The low values of the F-measures can be accounted for by the limited amount of training data. For each fold, we found that there are about 3-4 task sentences, thus a single misclassification of a task sentence leads to sizeable reduction in precision and recall which in turn results in a low F-measure value.

When balanced accuracy is the evaluation measure, SVM and random forest consistently yield similar performance when using either the LSA reduced feature set or the original feature set, although, random forest yielded a slightly higher performance compared to SVM using the LSA reduced features set. This seems to suggest that for balanced accuracy and employing the original features, one can choose between SVM and random forest, and if one decides to use the LSA feature set then random forest is to be preferred. Moreover, notice that the numerical value for balanced accuracy is higher than F-measure. Balanced accuracy can be increased by the accuracy of the dominant class, in this case the nontask class.

This classification example reveals the many issues that one may face in building a suitable classification model. First is the central role of features in classification. Second is how to model the relationship between the features and the class membership. Third is the crucial role of choosing an appropriate evaluation measure or performance metric. This choice should be guided by the nature of the problem, the objectives of the study, and the amount of error we are willing to tolerate. In our example, we assign equal importance to both classes, and we therefore have slight preference for balanced accuracy. In applications where the misclassification cost for the positive class is greater

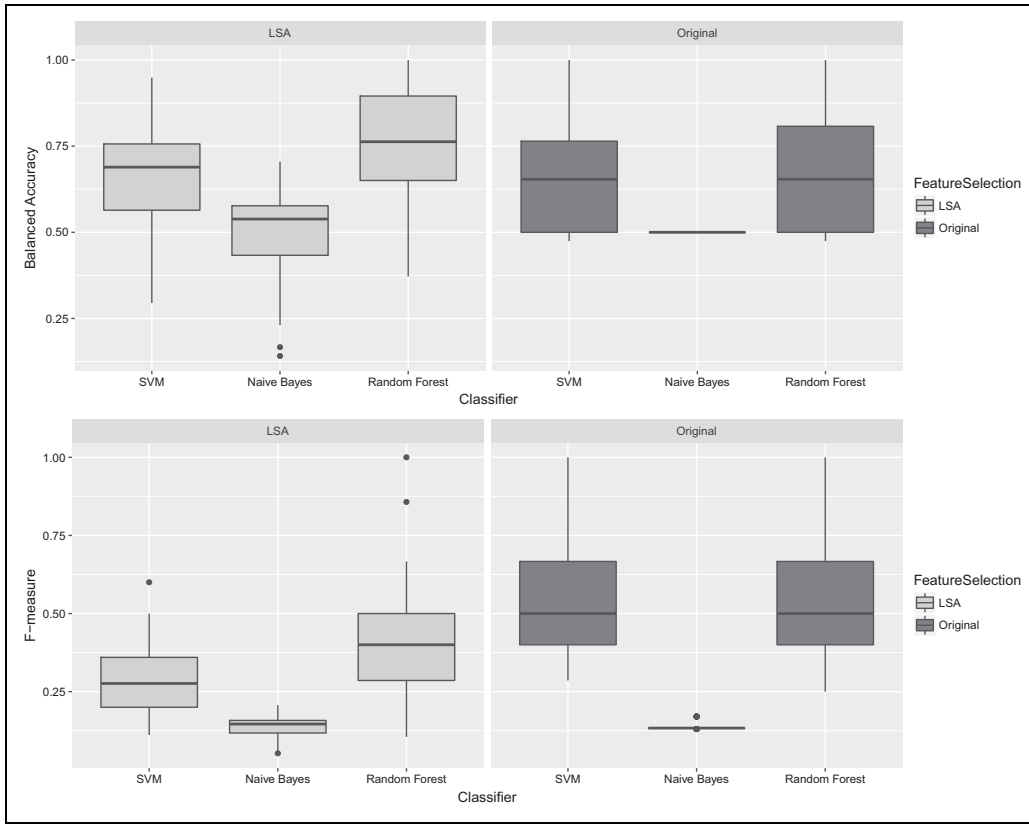



**Figure 5.** Comparison of classification performance among three classifiers and between the term-based and LSA-based features.

than that for the other class, the F-measure may be preferred. For a discussion of alternative evaluation measures see Powers (2011).

Other issues include the question of how to set a cutoff value for the evaluation measure to judge whether a model is good enough. A related question is how much training data are needed for the classification model to generalize well (i.e., how to avoid overfitting). These questions are best answered empirically through systematic model evaluation, such as by trying different training sizes and varying the threshold, and then observing the effect on classifier performance. One strategy is to treat this as a factorial experiment where the choices of training size and evaluation measures are considered as factor combinations. In addition, one has to perform repeated evaluation (e.g., cross-validation) and validation. Aside from modeling issues there are also practical concerns such as the cost of acquiring training data and the interpretability of the resulting model. Classification models with high predictive performance are not always the ones that yield the greatest insight. Insofar as the algorithm is to be used to support decision making, the onus is on the researcher to be able to explain and justify its workings.

## *Classification for Job Information Extraction*

For our work on job task information extraction three people hand labeled a total of 2,072 out of 60,000 sentences. It took a total of 3 days to label, verify and relabel 2,072 sentences. From this total,

132 sentences were identified as task sentences (note that the task sentences were not unique). The proportion of task sentences in vacancy texts was only 6%. This means that the resulting training data are imbalanced. This is because not all tasks that are part of a particular job will be written in the vacancies, likely only the essential and more general ones. This partly explains their low proportion.

Since labeling additional sentences will be costly and time-consuming we employed a semisupervised learning approach called label propagation (Zhu & Ghahramani, 2002). For the transformation and dimensionality reduction we respectively constructed the DTM and applied LSA. Once additional task sentences were obtained via semisupervised learning we ran three classification algorithms, namely, SVM, random forest, and naive Bayes. Instead of choosing a single classifier we combined the predictions of the three in a simple majority vote. For the evaluation measure we used the Recall measure since we wanted to obtain as many task sentences as possible. Cross-validation was used to assess the generalization property of the model. The application of classification resulted to identification of 1,179 new task sentences. We further clustered these sentences to obtain unique nursing tasks since some sentences pointed to the same tasks.

### *Model Reliability and Validity*

We set out to build a classification model that can extract sentences containing nursing tasks from job vacancies. Naturally, a subsequent step is to determine whether the extracted task sentences correspond to real tasks performed by nurses. An approach to establish construct validity is to use an independent source to examine the validity of the classification. Independent means that the source should be blind from the data collection activity, initial labeling procedure, and model building process. Moreover, in case ratings are obtained, these should be provided by subject matter experts (SMEs), that is, individuals who have specialist knowledge about the application domain. If found to be sufficiently valid the extracted sentences containing job tasks may then be used for other purposes such as in job analysis, identifying training needs or developing selection instruments.

We enlisted the help of SMEs, presented them the task sentences predicted by the text classifier, and asked them to check whether the sentences are actual nursing tasks or not so as to be able to compute the precision measure. Specifically, we compute precision as the ratio of the number of sentence tasks confirmed as actual nursing tasks to the total number of sentences tasks predicted by the model. We reran the classification algorithm in light of the input from the experts. The input is data containing the correct label of sentences which were misclassified by the classifier. We performed this in several iterations until there was no significant improvement in precision. This is necessarily an asymmetric approach since we use the expert knowledge as the "ground truth."

A more elaborate approach would be to compare the extracted tasks from vacancies to tasks collected using a more traditional job analysis method, namely a task inventory. The task inventory would consist of interviews and observations with SMEs to collect a list of tasks performed by nurses. Based on this comparison, a percentage of tasks would be found in both lists, a percentage of unique tasks would only be found in the task inventory, and a percentage of unique tasks would only be found in the online vacancies. A high correspondence between the list of tasks collected by text mining and the list of tasks collected in the task inventory (which would be considered to accurately reflect the nursing job) could be taken as evidence for convergent validity. Conversely, one could establish discriminant validity, or a very low correspondence with so-called bogus tasks that are completely unrelated to the nursing job.

We apply the less elaborate approach by first training a classification model, making predictions using the model, and presenting the task sentences to an SME. The expert judged whether the sentences are actual nursing tasks or not. The precision measure was used to give an indication of the validity of the model. The first round of validation resulted in a precision of 65% (precision range: 0% to 100%) and we found out that some of the initial labels we assigned did not match the

**Table 3.** Basic Care and Medical Care Core Nursing Tasks Extracted From Nursing Vacancies by Applying Text Classification.

| Task | German Translation | Task Cluster |
|---|---|---|
| Monitoring the patients' therapy | Überwachung der Therapie des Patienten | Basic care |
| Caring for the elderly | Pflege von älteren Menschen | Basic care |
| Providing basic or general care | Durchführung der Allgemeinen Pflege | Basic care |
| Providing palliative care | Durchführung von Palliativpflege | Basic care |
| Caring for mentally ill patients | Pflege von psychisch kranken Menschen | Basic care |
| Caring for children | Pflege von Kindern | Basic care |
| Assisting at intake of food | Hilfe bei der Nahrungsaufnahme | Basic care |
| Supporting of rehabilitation | Unterstützung der Rehabilitation | Basic care |
| Providing holistic care | Durchführung ganzheitlicher Pflege | Basic care |
| Accompanying patients | Begleitung von Patienten | Basic care |
| Assisting at surgical interventions | Assistenz bei operativen Eingriffen | Medical care |
| Doing laboratory tests | Durchführung von Labortests | Medical care |
| Participating in resuscitations | Beteiligung an Reanimationsmaßnahmen | Medical care |
| Conducting ECG | Durchführung von EKG | Medical care |
| Collecting blood | Durchführung der Blutabnahme | Medical care |
| Preparing and administer intravenous drugs | Vorbereitung und Verabreichung von intravenösen Medikamenten | Medical care |
| Assisting at diagnostic interventions | Assistenz bei diagnostischen Maßnahmen | Medical care |
| Operating the technical equipment | Bedienung der technischen Geräteschaften | Medical care |
| Assisting at endoscopic tests | Assistenz bei endoskopischen Maßnahmen | Medical care |
| Assisting at examination | Assistenz bei Untersuchungen | Medical care |

labels provided by the independent expert (that is some of the labels in the initial labels were judged to be erroneous by the expert). In light of this, we adjusted the labels and conducted a second round of validation in which precision increased to 89%. This indicates that we gained classification validity in the classification model. A total of 91 core tasks were validated. Table 3 contains validated tasks under the basic care and medical care clusters. In practice, it is difficult to obtain 100% precision since forcing a model to give high precision comes at the expense of sacrificing its recall. High precision and low recall imply the possibility that many task sentences will be dismissed though we can put more confidence on the sentences that are labeled as a task. As a last note, TC models are seldom static, that is, as new documents arrive, we have to continually assess the performance of the model on new observations and adjust our model if there is significant degradation in performance.

## Conclusion

This article provided an overview of TC and a tutorial on how to conduct actual TC on the problem of job task information extraction from vacancies. We discussed and demonstrated the different steps in TC and highlighted issues surrounding the choices of features, classification algorithms, and evaluation metrics. We also outlined ways to evaluate and validate the resulting classification models and prediction from these models. TC is an empirical enterprise where experimentation with choices of representation, dimensionality reduction, and classification techniques is standard practice. By building several classifiers and comparing them, the final classifier is chosen based on repeated evaluation and validation. Thus TC is not a linear process; one has to revisit each step iteratively to examine how choices in each step affects succeeding steps. Moreover, classifiers evolve in the presence of new data. TC is a wide research field and there are many other techniques

that were not covered here. An exciting new area is the application of deep learning techniques for text understanding (for more on this we refer the reader to Maas et al., 2011; Mikolov, Chen, Corrado, & Dean, 2013; X. Zhang & LeCun, 2015).

TC models are often descriptive as opposed to explanatory in nature, in the sense that they capture the pattern of features and inductively relate these to class membership (Bird, Klein, & Loper, 2009). This contrasts with explanatory models whose aim is to explain why the pattern in features leads to the prediction of a class. Nevertheless, the descriptive work can be of use for further theory building too as the knowledge of patterns can be used as a basis for the development of explanatory models. For example, in the part about feature selection we found out that the word sicherstellung (to guarantee or to make sure) is useful in detecting sentences containing nursing tasks. Based on this we can define the concept of "task verb," that is, a verb that is indicative of a task in the context of job vacancy. We could then compile a list of verbs that are "task verbs" and postulate that task verbs pair with noun or verb phrases to form task sentences. Further trials could then be designed to validate this concept and establish the relationship between features and patterns. In this way, we are not only detecting patterns but we also attempt to infer their properties and their relationship to class membership.

Whether a descriptive model suffices or whether an explanatory model is needed depends on the objectives of a specific study. If the objective is accurate and reliable categorization (e.g., when one is interested in using the categorized text as input to other systems) then a descriptive model will suffice although the outcomes still need to be validated. On the other hand, if the objective is to explain how patterns lead to categorization or how structure and form lead to meaning then an explanatory model is required.

In this article we tried to present TC in such a manner that organizational researchers can understand the underlying process. However, in practice, organizational researchers will often work with technical experts to make choices on the algorithms and assist in tweaking and tuning the parameters of the resulting model. The role of organizational researchers then is to provide the research questions, help select the relevant features, and provide insights in light of the classification output. These insights might lead to further investigation and ultimately to theory development and testing.

Finally, we conclude that TC offers great potential to make the conduct of text-based organizational researches fast, reliable, and effective. The utility of TC is most evident when there is a need to analyze massive text data, in fact in some cases TC is able to recover patterns that are difficult for humans to detect. Otherwise, manual qualitative text analysis procedures may suffice. As noted, the increased use of TC in organizational research will likely not only contribute to organizational research, but also to the advancement of TC research, because real problems and existing theory can further simulate the development of new techniques.

## Acknowledgments

An earlier version of this article was presented as Kobayashi, V. B., Berkers, H. A., Mol, S. T., Kismihók, G., & Den Hartog, D. N. (2015, August). Augmenting organizational research with the text mining toolkit: All aboard! In J. M. LeBreton (Chair), *Big Data: Implications for Organizational Research*. Showcase symposium at the 75th annual meeting of the Academy of Management, Vancouver, BC, Canada. The authors would like to acknowledge the anonymous reviewers at *Organizational Research Methods* and the Academy of Management for their constructive comments on earlier drafts of this article.

## Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the European Commission through the Marie-Curie Initial Training Network EDUWORKS (Grant PITN-GA-2013-608311) and by the Society of Industrial and Organizational Psychology Sidney A. Fine Grant for Research on Job Analysis, for the Big Data Based Job Analytics Project.

## References

Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. In C. C. Aggarwal & C. Zhai (Eds.), *Mining text data* (pp. 163-222). New York, NY: Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-1-4614-3223-4_6

Aizawa, A. (2003). An information-theoretic perspective of tf-idf measures. *Information Processing & Management*, *39*(1), 45-65. https://doi.org/10.1016/S0306-4573(02)00021-3

Algarni, A., & Tairan, N. (2014). Feature selection and term weighting. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)* (*Vol. 1*, pp. 336-339). Washington, DC: IEEE. https://doi.org/10.1109/WI-IAT.2014.53

Alpaydin, E. (2014). *Introduction to machine learning*. Cambridge, MA: MIT Press.

Atteveldt, W. van, Kleinnijenhuis, J., Ruigrok, N., & Schlobach, S. (2008). Good news or bad news? Conducting sentiment analysis on Dutch text to distinguish between positive and negative relations. *Journal of Information Technology & Politics*, *5*(1), 73-94. https://doi.org/10.1080/19331680802154145

Berry, M. W., & Castellanos, M. (2008). *Survey of text mining II—Clustering, classification, and retrieval*. Retrieved from http://www.springer.com/gp/book/9781848000452

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python*. Sebastopol, CA: O'Reilly Media. Retrieved from https://books.google.nl

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123-140.

Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *20th International Conference on Pattern Recognition (ICPR)* (pp. 3121-3124). Washington, DC: IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5597285

Brooks, J., McCluskey, S., Turley, E., & King, N. (2015). The utility of template analysis in qualitative psychology research. *Qualitative Research in Psychology*, *12*(2), 202-222. https://doi.org/10.1080/14780887.2014.955224

Buhrmester, M., Kwang, T., & Gosling, S. D. (2011). Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data? *Perspectives on Psychological Science*, *6*(1), 3-5. https://doi.org/10.1177/1745691610393980

Burges, C. J. (2010). *Dimension reduction: A guided tour*. Redmond, WA: Now Publishers. Retrieved from https://books.google.nl

Cardie, C., & Wilkerson, J. (2008). Text annotation for political science research. *Journal of Information Technology & Politics*, *5*(1), 1-6. https://doi.org/10.1080/19331680802149590

Cavnar, W. B., & Trenkle, J. M. (1994). *N-gram-based text categorization*. Ann Arbor, MI: Environmental Research Institute of Michigan.

Chan, S. W. K., & Chong, M. W. C. (2017). Sentiment analysis in financial texts. *Decision Support Systems*, *94*, 53-64. https://doi.org/10.1016/j.dss.2016.10.006

Chan, S. W. K., & Franklin, J. (2011). A text-based decision support system for financial sequence prediction. *Decision Support Systems*, *52*(1), 189-198. https://doi.org/10.1016/j.dss.2011.07.003

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), 15:1-15:58. https://doi.org/10.1145/1541880.1541882

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, *16*, 321-357.

Chen, J., Huang, H., Tian, S., & Qu, Y. (2009). Feature selection for text classification with naïve Bayes. *Expert Systems with Applications*, *36*(3, pt. 1), 5432-5435. https://doi.org/10.1016/j.eswa.2008.06.054

Dave, K., Lawrence, S., & Pennock, D. M. (2003). Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th International Conference on World Wide Web* (pp. 519-528). New York, NY: ACM. https://doi.org/10.1145/775152.775226

Dietterich, T. G. (1997). Machine-learning research. *AI Magazine*, *18*(4), 97.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78-87.

Dong, Y.-S., & Han, K.-S. (2004). A comparison of several ensemble methods for text categorization. In *2004 IEEE International Conference on Services Computing 2004 (SCC 2004)* (pp. 419-422). Washington, DC: IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1358033

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management* (pp. 148-155). New York, NY: ACM. Retrieved from http://dl.acm.org/citation.cfm?id=288651

Duriau, V. J., Reger, R. K., & Pfarrer, M. D. (2007). A content analysis of the content analysis literature in organization studies: Research themes, data sources, and methodological refinements. *Organizational Research Methods*, *10*(1), 5-34. https://doi.org/10.1177/1094428106289252

Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (*Vol. 2*, pp. 973-978). San Francisco, CA: Morgan Kaufmann. Retrieved from http://dl.acm.org/citation.cfm?id=1642194.1642224

Eyheramendy, S., Lewis, D. D., & Madigan, D. (2003). *On the naive Bayes model for text categorization*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.4949

Fernandes, J., Artífice, A., & Fonseca, M. J. (2017). Automatic Estimation of the LSA Dimension. *Paper presented at the International Conference of Knowledge Discovery and Information Retrieval*. Paris, France, October 2011. Retrieved from http://www.di.fc.ul.pt/~mjf/publications/2014-2010/pdf/kdir11.pdf.

Ferreira, A. J., & Figueiredo, M. A. T. (2012). Boosting algorithms: A review of methods, theory, and applications. In C. Zhang & Y. Ma (Eds.), *Ensemble machine learning* (pp. 35-85). New York, NY: Springer. https://doi.org/10.1007/978-1-4419-9326-7_2

Finn, A., & Kushmerick, N. (2006). Learning to classify documents according to genre. *Journal of the American Society for Information Science and Technology*, *57*(11), 1506-1518. https://doi.org/10.1002/asi.20427

Flach, P. (2012). *Machine learning: The art and science of algorithms that make sense of data*. New York, NY: Cambridge University Press.

Fodor, I. K. (2002). *A survey of dimension reduction techniques* (Tech. Rep. UCRL-ID-148494). Livermore, CA: Lawrence Livermore National Laboratory. Retrieved from https://e-reports-ext.llnl.gov/pdf/240921.pdf

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, *3*, 1289-1305.

Fox, C. (1992). Lexical analysis and stoplists. In W. B. Frakes & R. Baeza-Yates (Eds.), *Information retrieval: Data structures and algorithms* (pp. 102-130). Upper Saddle River, NJ: Prentice Hall. Retrieved from http://dl.acm.org/citation.cfm?id=129687.129694

Frakes, W. B. (1992). Stemming algorithms. In W. B. Frakes & R. Baeza-Yates (Eds.), *Information retrieval: Data structures and Algorithms* (pp. 131-160). Upper Saddle River, NJ: Prentice Hall. Retrieved from http://dl.acm.org/citation.cfm?id=129687.129695

Frakes, W. B., & Baeza-Yates, R. (1992). *Information retrieval: Data structures and algorithms*. Retrieved from http://www.citeulike.org/group/328/article/308697

Fu, Y., Zhu, X., & Li, B. (2013). A survey on instance selection for active learning. *Knowledge and Information Systems*, *35*(2), 249-283. https://doi.org/10.1007/s10115-012-0507-8

Goldman, S. A. (2010). Computational learning theory. In M. J. Atallah & M. Blanton (Eds.), *Algorithms and theory of computation handbook* (2nd ed., *Vol. 1*, pp. 26-26). London. UK: Chapman & Hall/CRC. Retrieved from http://dl.acm.org/citation.cfm?id=1882757.1882783

Gonçalves, T., & Quaresma, P. (2005). Is linguistic information relevant for the classification of legal texts? In *Proceedings of the 10th International Conference on Artificial Intelligence and Law* (pp. 168-176). New York, NY: ACM. https://doi.org/10.1145/1165485.1165512

Graham, I. S. (1995). *The HTML sourcebook*. New York, NY: John Wiley. Retrieved from http://dl.acm.org/citation.cfm?id=526978

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157-1182.

Guyon, I., Gunn, S., Nikravesh, M., & Zadeh, L. (2008). *Feature extraction: Foundations and applications*. New York, NY: Springer.

Harish, B. S., Guru, D. S., & Manjunath, S. (2010). Representation and classification of text documents: A brief review. *International Journal of Computer Applications*, *2*, 110-119.

Hindle, D., & Rooth, M. (1993). Structural ambiguity and lexical relations. *Computational Linguistics*, *19*(1), 103-120.

Holton, C. (2009). Identifying disgruntled employee systems fraud risk through text mining: A simple solution for a multi-billion dollar problem. *Decision Support Systems*, *46*(4), 853-864. https://doi.org/10.1016/j.dss.2008.11.013

Hsieh, H.-F., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative Health Research*, *15*(9), 1277-1288. https://doi.org/10.1177/1049732305276687

Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, *13*(2), 415-425. https://doi.org/10.1109/72.991427

Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS Transactions on Computers*, *4*(8), 966-974.

Jiang, S., Pang, G., Wu, M., & Kuang, L. (2012). An improved K-nearest-neighbor algorithm for text categorization. *Expert Systems with Applications*, *39*(1), 1503-1509.

Joachims, T. (1998). *Text categorization with support vector machines: Learning with many relevant features*. New York, NY: Springer. Retrieved from http://link.springer.com/chapter/10.1007/BFb0026683

Kanaan, G., Al-Shalabi, R., Ghwanmeh, S., & Al-Ma'adeed, H. (2009). A comparison of text-classification techniques applied to Arabic text. *Journal of the American Society for Information Science and Technology*, *60*(9), 1836-1844. https://doi.org/10.1002/asi.20832

Khoo, A., Marom, Y., & Albrecht, D. (2006). Experiments with sentence classification. In *Proceedings of the 2006 Australasian language technology workshop* (pp. 18-25). Retrieved from http://www.aclweb.org/anthology/U06-1#page=26

Kloptchenko, A., Eklund, T., Karlsson, J., Back, B., Vanharanta, H., & Visa, A. (2004). Combining data and text mining techniques for analysing financial reports. *Intelligent Systems in Accounting, Finance & Management*, *12*(1), 29-41. https://doi.org/10.1002/isaf.239

Kobayashi, V. B., Mol, S. T., Berkers, H. A., Kismihók, G., & Den Hartog, D. N. (2018). Text mining in organizational research. *Organizational Research Methods*, *21*(3), 733-765.

Kobayashi, V. B., Mol, S. T., Kismihók, G., & Hesterberg, M. (2016). Automatic extraction of nursing tasks from online job vacancies. In M. Fathi, M. Khobreh, & F. Ansari (Eds.), *Professional education and training through knowledge, technology and innovation* (pp. 51-56). Siegen, Germany: University of Siegen. Retrieved from http://dokumentix.ub.uni-siegen.de/opus/volltexte/2016/1057/pdf/Professional_education_and_training.pdf#page=58

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai* (*Vol. 14*, pp. 1137-1145). Retrieved from http://frostiebek.free.fr/docs/Machine%20Learning/validation-1.pdf

Kolen, J. F., & Pollack, J. B. (1990). Back propagation is sensitive to initial conditions. In *Proceedings of the 1990 Conference on Advances in Neural Information Processing Systems 3* (pp. 860-867). San Francisco, CA: Morgan Kaufmann. Retrieved from http://dl.acm.org/citation.cfm?id=118850.119960

Lan, M., Tan, C. L., Su, J., & Lu, Y. (2009). Supervised and traditional term weighting methods for automatic text categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(4), 721-735. https://doi.org/10.1109/TPAMI.2008.110

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, *25*(2-3), 259-284. https://doi.org/10.1080/01638539809545028

Lewis, D. D. (1992). *Representation and learning in information retrieval* (Doctoral dissertation, University of Massachusetts Amherst). Retrieved from http://ciir.cs.umass.edu/pubfiles/UM-CS-1991-093.pdf

Li, Y. H., & Jain, A. K. (1998). Classification of text documents. *Computer Journal*, *41*(8), 537-546. https://doi.org/10.1093/comjnl/41.8.537

Liu, H., & Motoda, H. (1998). *Feature extraction, construction and selection: A data mining perspective*. New York, NY: Springer.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human language technologies* (*Vol. 1*, pp. 142-150). Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=2002491

Méndez, J. R., Iglesias, E. L., Fdez-Riverola, F., Díaz, F., & Corchado, J. M. (2006). Tokenising, stemming and stopword removal on anti-spam filtering domain. In R. Marín, E. Onaindía, A. Bugarín, & J. Santos (Eds.), *Current topics in artificial intelligence* (pp. 449-458). New York, NY: Springer. Retrieved from http://link.springer.com/chapter/10.1007/11881216_47

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space* (ArXiv Preprint ArXiv:1301.3781). Retrieved from http://arxiv.org/abs/1301.3781

Moschitti, A., & Basili, R. (2004). Complex linguistic features for text classification: A comprehensive study. In S. McDonald & J. Tait (Eds.), *Advances in information retrieval* (pp. 181-196). New York, NY: Springer. https://doi.org/10.1007/978-3-540-24752-4_14

Ogura, H., Amano, H., & Kondo, M. (2011). Comparison of metrics for feature selection in imbalanced text classification. *Expert Systems with Applications*, *38*(5), 4978-4989. https://doi.org/10.1016/j.eswa.2010.09.153

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*(1-2), 1-135. https://doi.org/10.1561/1500000011

Panigrahi, P. K. (2012). A comparative study of supervised machine learning techniques for spam e-mail filtering. In *2012 Fourth International Conference on Computational Intelligence and Communication Networks* (pp. 506-512). Washington, DC: IEEE. https://doi.org/10.1109/CICN.2012.14

Phan, X.-H., Nguyen, L.-M., & Horiguchi, S. (2008). Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th International Conference on World Wide Web* (pp. 91-100). New York, NY: ACM. https://doi.org/10.1145/1367497.1367510

Polikar, R. (2012). Ensemble learning. In C. Zhang & Y. Ma (Eds.), *Ensemble machine learning* (pp. 1-34). New York, NY: Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-1-4419-9326-7_1

Popping, R. (2012). Qualitative decisions in quantitative text analysis research. *Sociological Methodology*, *42*, 88-90.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130-137. https://doi.org/10.1108/eb046814

Powers, D. M. (2011). *Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation*. Retrieved from http://dspace2.flinders.edu.au/xmlui/handle/2328/27165

Ragas, H., & Koster, C. H. (1998). Four text classification algorithms compared on a Dutch corpus. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in*

*information retrieval* (pp. 369-370). New York, NY: ACM. Retrieved from http://dl.acm.org/citation.cfm?id=291059

Raghavan, V. V., & Wong, S. M. (1986). A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*, *37*(5), 279-287.

Rokach, L., & Maimon, O. (2005). Top-down induction of decision trees classifiers—A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *35*(4), 476-487. https://doi.org/10.1109/TSMCC.2004.843247

Rullo, P., Cumbo, C., & Policicchio, V. L. (2007). Learning rules with negation for text categorization. In *Proceedings of the 2007 ACM Symposium on Applied Computing* (pp. 409-416). New York, NY: ACM. https://doi.org/10.1145/1244002.1244098

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513-523.

Salton, G., Wong, A., & Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613-620.

Scharkow, M. (2013). Thematic content analysis using supervised machine learning: An empirical evaluation using German online news. *Quality & Quantity*, *47*(2), 761-773. https://doi.org/10.1007/s11135-011-9545-7

Scott, S., & Matwin, S. (1999). Feature engineering for text classification. In *ICML* (*Vol. 99*, pp. 379-388). Retrieved from http://comp.mq.edu.au/units/comp348/reading/scott99feature.pdf

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1-47.

Seibert, S. E., Kraimer, M. L., Holtom, B. C., & Pierotti, A. J. (2013). Even the best laid plans sometimes go askew: Career self-management processes, career shocks, and the decision to pursue graduate education. *Journal of Applied Psychology*, *98*(1), 169.

Settles, B. (2010). *Active learning literature survey* (Tech. Rep. 1648). Madison: University of Wisconsin–Madison.

Shen, J., Brdiczka, O., & Liu, J. (2013). Understanding email writers: Personality prediction from email messages. In S. Carberry, S. Weibelzahl, A. Micarelli, & G. Semeraro (Eds.), *User modeling, adaptation, and personalization* (pp. 318-330). New York, NY: Springer. https://doi.org/10.1007/978-3-642-38844-6_29

Sheng, V. S., Provost, F., & Ipeirotis, P. G. (2008). Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 614-622). New York, NY: ACM. https://doi.org/10.1145/1401890.1401965

Shreve, J., Schneider, H., & Soysal, O. (2011). A methodology for comparing classification methods through the assessment of model stability and validity in variable selection. *Decision Support Systems*, *52*(1), 247-257. https://doi.org/10.1016/j.dss.2011.08.001

Sirbu, D., Secui, A., Dascalu, M., Crossley, S. A., Ruseti, S., & Trausan-Matu, S. (2016). Extracting gamers' opinions from reviews. In *2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)* (pp. 227-232). https://doi.org/10.1109/SYNASC.2016.044

Song, F., Liu, S., & Yang, J. (2005). A comparative study on text representation schemes in text categorization. *Pattern Analysis and Applications*, *8*(1-2), 199-209. https://doi.org/10.1007/s10044-005-0256-3

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., & Kappas, A. (2010). Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, *61*(12), 2544-2558.

Toman, M., Tesar, R., & Jezek, K. (2006). Influence of word normalization on text classification. In *Proceedings of InSciT* (pp. 354-358). Merida, Spain. Retrieved from http://www.kiv.zcu.cz/research/groups/text/publications/inscit20060710.pdf

Torunoğlu, D., Çakırman, E., Ganiz, M. C., Akyokuş, S., & Gürbüz, M. Z. (2011). Analysis of preprocessing methods on classification of Turkish texts. In *2011 International Symposium on Innovations in Intelligent*

*Systems and Applications (INISTA)* (pp. 112-117). Washington, DC: IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5946084

Tsuge, S., Shishibori, M., Kuroiwa, S., & Kita, K. (2001). Dimensionality reduction using non-negative matrix factorization for information retrieval. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)* (*Vol. 2*, pp. 960-965). Washington, DC: IEEE. https://doi.org/10.1109/ICSMC.2001.973042

Turney, P. (1999). *Learning to extract keyphrases from text*. Retrieved from http://nparc.cisti-icist.nrc-cnrc.gc.ca/npsi/ctrl?action=rtdoc&an=8913245

Uysal, A. K., & Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, *50*(1), 104-112. https://doi.org/10.1016/j.ipm.2013.08.006

van der Maaten, L. J., Postma, E. O., & van den Herik, H. J. (2009). Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, *10*(1-41), 66-71.

Vo, D.-T., & Ock, C.-Y. (2015). Learning to classify short text from scientific documents using topic models with various types of knowledge. *Expert Systems with Applications*, *42*(3), 1684-1698. https://doi.org/10.1016/j.eswa.2014.09.031

Wiebe, J., Wilson, T., Bruce, R., Bell, M., & Martin, M. (2004). Learning subjective language. *Computational Linguistics*, *30*(3), 277-308. https://doi.org/10.1162/0891201041850885

Wiebe, J., Wilson, T., & Cardie, C. (2005). Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, *39*(2-3), 165-210. https://doi.org/10.1007/s10579-005-7880-9

Willett, P. (2006). The Porter stemming algorithm: Then and now. *Program*, *40*(3), 219-223. https://doi.org/10.1108/00330330610681295

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, *1*(1-2), 69-90.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *ICML* (*Vol. 97*, pp. 412-420). Retrieved from http://www.surdeanu.info/mihai/teaching/ista555-spring15/readings/yang97comparative.pdf

Youn, S., & McLeod, D. (2007). A comparative study for email classification. In K. Elleithy (Ed.), *Advances and innovations in systems, computing sciences and software engineering* (pp. 387-391). New York, NY: Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-1-4020-6264-3_67

Yu, B., Kaufmann, S., & Diermeier, D. (2008). Classifying party affiliation from political speech. *Journal of Information Technology & Politics*, *5*(1), 33-48. https://doi.org/10.1080/19331680802149608

Yu, H.-F., Ho, C.-H., Arunachalam, P., Somaiya, M., & Lin, C.-J. (2012). *Product title classification versus text classification*. Retrieved from http://www.csie.ntu.edu.tw/∼cjlin/papers/title.pdf

Zhang, J., Jin, R., Yang, Y., & Hauptmann, A. (2003). Modified logistic regression: An approximation to SVM and its applications in large-scale text categorization. In *ICML* (pp. 888-895). Retrieved from http://www.aaai.org/Papers/ICML/2003/ICML03-115.pdf

Zhang, W., Yoshida, T., & Tang, X. (2008). Text classification based on multi-word with support vector machine. *Knowledge-Based Systems*, *21*(8), 879-886. https://doi.org/10.1016/j.knosys.2008.03.044

Zhang, X., & LeCun, Y. (2015). *Text understanding from scratch* (ArXiv Preprint ArXiv:1502.01710). Retrieved from http://arxiv.org/abs/1502.01710

Zhu, X. (2005). *Semi-supervised learning literature survey*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.9681&rep=rep1&type=pdf

Zhu, X., & Ghahramani, Z. (2002). *Learning from labeled and unlabeled data with label propagation*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.3864&rep=rep1&type=pdf

Zu, G., Ohyama, W., Wakabayashi, T., & Kimura, F. (2003). Accuracy improvement of automatic text classification based on feature transformation. In *Proceedings of the 2003 ACM Symposium on Document Engineering* (pp. 118-120). New York, NY: ACM. https://doi.org/10.1145/958220.958242

Zurada, J. M., Ensari, T., Asl, E. H., & Chorowski, J. (2013). Nonnegative matrix factorization and its application to pattern analysis and text mining. In *2013 Federated Conference on Computer Science and Information Systems (FedCSIS)* (pp. 11-16). Washington, DC: IEEE.

## Author Biographies

**Vladimer B. Kobayashi** is a PhD student at the University of Amsterdam and on study leave from the University of the Philippines Mindanao. His current research interest is in labor market driven learning analytics. Specifically, he uses text mining techniques and machine learning to automatically extract information from job vacancies, to understand education-to-labor market transition, to study job mobility, to determine success in the labor market, and to match education and labor market needs. He plans to continue this line of research by applying his training and knowledge in industry context.

**Stefan T. Mol** is an assistant professor of organizational behavior at the Leadership and Management Section of the Amsterdam Business School of the University of Amsterdam, the Netherlands. His research interests center on a variety of topics relating to the broad fields of organizational behavior and research methods, including but not limited to person-environment fit, refugee integration in the labor market, employability, job crafting, calling, work identity, psychological contracts, expatriate management, text mining (with a focus on vacancies), learning analytics, and meta-analysis.

**Hannah A. Berkers** is a PhD candidate in organizational behavior at the Leadership and Management Section of the Amsterdam Business School of the University of Amsterdam, the Netherlands. Her research interests include work and professional identity, calling, employee well-being, meaningfulness, job crafting, and a task level perspective on work.

**Gábor Kismihók** is a postdoc in knowledge management at the Leadership and Management Section of the Amsterdam Business School of the University of Amsterdam, the Netherlands. His research is focusing on the bridge between education and the labor market, and entails topics like learning analytics, refugee integration in the labor market, and employability.

**Deanne N. Den Hartog** is a professor of organizational behavior, head of the Leadership and Management Section, and director of the Amsterdam Business School Research Institute at the University of Amsterdam, the Netherlands and visiting research professor at the University of Southern Australia. Her research interests include leadership, HRM, proactive work behavior, and trust. She has published widely on these topics and serves on several editorial boards.