FeatureForge: A Novel Tool for Visually Supported Feature Engineering and Corpus Revision

Florian Heimerl[1] *Charles Jochim*[2] *Steffen Koch*[1] *Thomas Ertl*[1]
(1) Institute for Visualization and Interactive Systems (VIS), University of Stuttgart
(2) Institute for Natural Language Processing (IMS), University of Stuttgart
`firstname.lastname@[vis|ims].uni-stuttgart.de`

Abstract
In many fields of NLP, supervised machine learning methods reach the best performance results. Apart from creating new classification models, there are two possibilities to improve classification performance: (i) improve the comprehensiveness of feature representations of linguistic instances, and (ii) improve the quality of the training gold standard. While researchers in some fields can rely on standard corpora and feature sets, others have to create their own domain specific corpus and feature representations. The same is true for practitioners developing NLP-based applications. We present a software prototype that uses interactive visualization to support researchers and practitioners in two aspects: (i) spot problems with the feature set and define new features to improve classification performance, and (ii) find groups of instances hard to label or that get systematically mislabeled by annotators to revise the annotation guidelines.

Title and Abstract in German

FeatureForge: Ein neues Werkzeug für visuell unterstütze Merkmalsoptimierung und Korpusüberarbeitung

In vielen Bereichen der maschinellen Sprachverarbeitung erzielen überwachte Lernmethoden die besten Ergebnisse. Neben dem Entwurf neuer Klassifikationsmodelle gibt es zwei Möglichkeiten die Klassifikationsleistung bestehender Modelle zu verbessern: (i) durch Verbessern des Informationsgehalts der Merkmalsrepräsentationen und (ii) durch Verbessern der Qualität des Goldstandards der Trainingsdaten. Während manche Forscher auf Standardkorpora und -merkmale zurückgreifen können, müssen andere eigene domänenspezifische Korpora und Merkmalsrepräsentationen erstellen. Gleiches gilt für NLP-basierte Anwendungen in der Praxis. Wir stellen einen Softwareprototyp vor, der Forscher und Entwickler mit interaktiver Visualisierung auf zwei Arten unterstützt: (i) beim Auffinden von durch die Merkmalsmenge erzeugten Problemen und der Definition von neuen Merkmalen zur Verbesserung der Klassifikationsleistung und (ii) bei der Überarbeitung der Annotationsrichtlinien durch Auffinden von Instanzengruppen die Problemfälle beim Labeln darstellen oder von Annotatoren systematisch falsch gelabelt werden.

Keywords: Feature Engineering, Corpus Annotation, Interactive Visualization, Machine Learning, Citation Classification.

Keywords in L_2: Feature Engineering, Korpusannotation, Interaktive Visualisierung, Maschinelles Lernen, Zitationsklassifikation

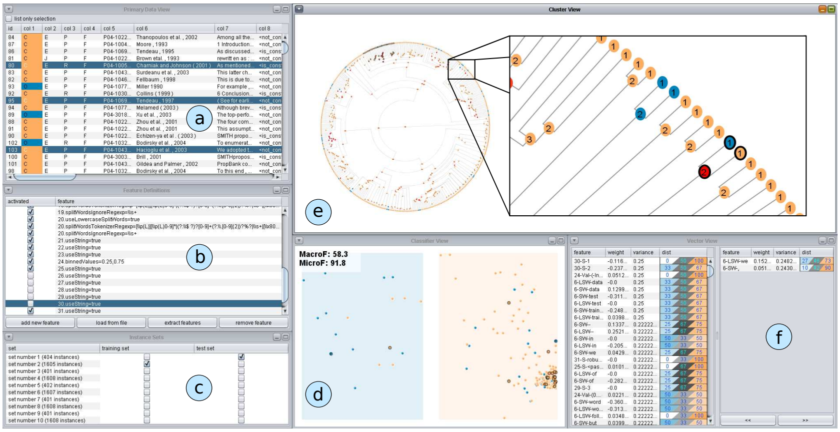
---

1 Introduction

Supervised machine learning methods provide state-of-the-art performance in many fields of NLP. Researchers who want to advance the state-of-the-art either deal with the development of new classification methods that better model the linguistic data, or improve training data by developing clean and accurately labeled corpora and feature definitions that allow for an effective vector representation of linguistic entities and provide as much discriminatory information as possible for accurate classification. Guyon and Elisseeff (2003) identify the feature definition step as the one that should be tackled first when trying to improve classification performance. In this paper, we present the prototype of an interactive software system that helps researchers with the task of spotting problems with their vector representations as well as with the gold labels of their corpora. It is based on concepts from the field of Visual Analytics, which evolved from the field of visualization and incorporates methods from data mining, data representation, and human computer interaction. Visual Analytics systems (Cook and Thomas, 2005; Keim et al., 2008) typically employ automatic data aggregation and data mining methods that enable users to retrieve useful information out of vast and often unstructured amounts of data. Data aggregation and filtering methods are used to highlight relevant aspects of the data and the data mining models and visualizations can be steered and controlled through user interaction. Such an interactive system can support feature engineers in NLP, who need to get a comprehensive overview of the linguistic data at hand in order to derive meaningful linguistic features that describe the data accurately with respect to a specific classification problem. While interacting with the tool, a sensemaking (Russell et al., 1993) loop is established and new insights into the data, the usefulness of features and the interaction between feature representations and classification algorithms are created. The presented tool combines interactive visualization techniques with unsupervised clustering methods to effectively support researchers with the task of refining vector representations by creating new features, as well as finding systematic annotation errors in the corpus. It provides information about the effectiveness and discriminatory power (cf. Somol et al., 2010) of the current feature representation by displaying the current state of the classifier in combination with a hierarchical clustering of the instance space to easily identify problematic instances. Feature engineers can examine such instances, derive new features, implement and test them. When adding a new feature, the system updates all views instantaneously and thus gives much richer feedback about their impact than just the bare performance numbers of the resulting classifier. Apart from deriving new features, mislabeled instances can help updating annotation guidelines by refining or adding annotation rules. Some of the mislabeled instances can additionally be used as examples for annotation guidelines in order to convey labeling rules more effectively.

2 The FeatureForge System

The first part of this section presents the FeatureForge desktop, depicted in Figure 1. It describes what information is displayed in its views and how it is presented to the users. The second part concentrates on interaction with the views and their interplay. FeatureForge currently integrates the feature definition language of the *ColumnDataClassifier* which is part of the Stanford Classifier suite (Manning and Klein, 2003) and makes them available to users, with the restriction of only allowing Boolean feature definitions, resulting in Boolean instance vectors. The prototype supports the column-based file format of the *ColumnDataClassifier* for the primary data of instances. Primary data is the data on which features are defined, and which has been extracted from the linguistic classification entities in a previous step. Users are able to load an arbitrary number of files containing instance sets.



list only selection

id	col 1	col 2	col 3	col 4	col 5	col 6	col 7	col 8
84	C	E	P	F	P04-1022...	Thanopoulos et al. , 2002	Among all the...	<not_con
87	C	E	P	F	P04-1004...	Moore , 1993	1 Introduction...	<not_con
86	C	E	P	F	P04-1069...	Tendeau , 1995	As discussed...	<is_cons
81	C	J	P	F	P04-1022...	Brown etal. , 1993	rewritt en as : ...	<not_con
80	C	E	R	F	P04-1005...	Charniak and Johnson (2001)	As mentioned...	<is_cons
83	C	E	P	F	P04-1043...	Surdeanu et al. , 2003	This latter ch...	<not_con
82	C	E	P	F	P04-1046...	Fellbaum , 1998	This is due to...	<not_con
93	O	E	P	F	P04-1077...	Miller 1990	For example ,...	<not_con
92	C	E	R	F	P04-1030...	Collins (1999)	6 Conclusion...	<is_cons
95	C	E	P	F	P04-1069...	Tendeau , 1997	(See for earli...	<not_con
94	C	E	P	F	P04-1077...	Melamed (2003)	Although brev...	<is_cons
89	O	E	P	F	P04-3018...	Xu et al. , 2003	The top-perfo...	<not_con
88	C	E	P	F	P04-1022...	Zhou et al. , 2001	The four com...	<not_con
91	C	E	P	F	P04-1022...	Zhou et al. , 2001	This assumpt...	<not_con
90	C	E	P	F	P04-1022...	Echizen-ya et al . (2003)	SMITH propo...	<is_cons
102	O	E	R	F	P04-1032...	Bodirsky et al. , 2004	To enumerat...	<not_con
103	C	E	P	F	P04-1043...	Hacioglu et al. , 2003	We adopted t...	<not_con
100	C	E	P	F	P04-3003...	Brill , 2001	SMITHpropos...	<is_cons
101	C	E	P	F	P04-1043...	Gildea and Palmer , 2002	PropBank co...	<not_con
98	C	E	P	F	P04-1032...	Bodirsky et al. , 2004	To this end , ...	<not_con

activated	feature
✓	19.splitWordsIgnoreRegexp=\\s+
✓	20.useLowercaseSplitWords=true
✓	20.splitWordsTokenizerRegexp=[\\p{L}][\\p{L}0-9]*|(?:\\$?)?[0-9]+(?:\\.[0-9]{2})?%?|\\s+|[\\x80...
✓	20.splitWordsIgnoreRegexp=\\s+
✓	21.useString=true
✓	22.useString=true
✓	23.useString=true
✓	24.binnedValues=0.25,0.75
✓	25.useString=true
☐	26.useString=true
☐	27.useString=true
☐	28.useString=true
☐	29.useString=true
☐	30.useString=true
✓	31.useString=true

set	training set	test set
set number 1 (404 instances)	☐	✓
set number 2 (1605 instances)	✓	☐
set number 3 (401 instances)	☐	☐
set number 4 (1608 instances)	☐	☐
set number 5 (402 instances)	☐	☐
set number 6 (1607 instances)	☐	☐
set number 7 (401 instances)	☐	☐
set number 8 (1608 instances)	☐	☐
set number 9 (401 instances)	☐	☐
set number 10 (1608 instances)	☐	☐

MacroF: 58.3
MicroF: 91.8

feature	weight	variance	dist
30-S-1	-0.116...	0.25	0 50 100
30-S-2	-0.237...	0.25	33 50 67
24-Val-(-In...	0.0512...	0.25	0 50 100
6-LSW-data	-0.0	0.25	33 50 67
6-SW-data	0.1299...	0.25	33 50 67
6-SW-test	-0.311...	0.25	33 50 67
6-LSW-test	-0.0	0.25	33 50 67
6-SW-train...	-0.248...	0.25	33 50 67
6-LSW-trai...	0.0398...	0.25	33 50 67
6-SW--	0.1337...	0.22222...	25 67 75
6-LSW--	0.2521...	0.22222...	25 67 75
6-SW-in	-0.0	0.22222...	50 33 50
6-LSW-in	-0.205...	0.22222...	50 33 50
6-SW-we	0.0429...	0.22222...	25 67 75
31-S-robu...	-0.0	0.22222...	0 33 100
25-S-<pas...	0.0101...	0.22222...	0 67 100
6-LSW-of	-0.0	0.22222...	25 67 75
6-SW-of	-0.282...	0.22222...	25 67 75
29-S-3	-0.0	0.22222...	25 67 75
24-Val-(0....	0.0221...	0.22222...	50 33 50
6-SW-word	-0.360...	0.22222...	50 33 50
6-LSW-wo...	-0.313...	0.22222...	50 33 50
6-LSW-foll...	0.0348...	0.22222...	0 33 100
6-SW-but	0.0399...	0.22222...	50 33 50

feature	weight	variance	dist
6-LSW-we	0.152...	0.2482...	27 46 73
6-SW-,	0.051...	0.2430...	10 42 90

Figure 1: The FeatureForge desktop with (a) the Primary Data View, (b) the Feature Definition View, (c) the Instance Set View, (d) the Classifier View, (e) the Cluster View, and (f) the Vector Set View. The picture shows the cluster tree zoomed out and part of it enlarged for illustration.

2.1 Views and Visualizations

The Primary Data View (a) displays a table containing the instances from the loaded instance sets. Alternatively, by selecting the checkbox on top, the list of instances displayed can be restricted to the currently selected instances of other views. The column that contains the gold labels is highlighted by color. Users can select colors for each class in the dataset. The current prototype supports only data with binary labels (we chose the colors blue and ocher for them in Figure 1).

The Feature Definition View (b) shows a list of the currently defined features and allows the user to add, remove, activate and deactivate features at any time. Feature definitions can also be loaded from file. After users have modified the list of feature definitions in this view, the feature representations for the currently loaded instance sets are updated by clicking the *extract features* button.

The Instance Set View (c) holds a list of all the instance sets loaded. One training and one test set have to be selected from all available sets. The training set is used to train a classification model with one of the learning algorithms integrated into FeatureForge, while the test set is used for classifier evaluation and as a basis for the definition of new features.

The Classifier View (d) visualizes the classification model with the test set to give users an impression of the classifier's state and performance. Currently the classification methods integrated in the FeatureForge prototype are the *linear* and the *logistic* classifiers from the Stanford suite, and linear support vector machines (LibLinear, Fan et al., 2008). The decision boundary of a model is depicted as a white line separating the two instance half spaces. The instances of the test set are mapped on the 2D plane of the Classifier View. For the arrangement along the horizontal axis the classifier confidence (either geometric distance from decision boundary or probability estimate) is used. The position along the vertical axis is determined by the first principal component of the 50 most uncertain instances on each side of the decision boundary. We restrict the PCA to a subset of test instances to keep computational expense at a feasible level and guarantee interactivity. We chose examples with high uncertainty for this to keep the fidelity of their 2D representation highest. The vertical position of all other instances is determined by $v(d_i) = \frac{\sum_{d \in U_i} e(d_i,d)^{-1} \cdot v(d)}{\sum_{d \in U_i} e(d_i,d)^{-1}}$ where $v(d_i)$ gives the vertical position of an instance d_i, U_i are the ten instances from the PCA set closest to d_i, and e gives the Euclidean distance in the original vector space (cf. Heimerl et al., 2012). The Classifier View colors the instances from the test set according to their gold label to allow easy identification of misclassified instances. This view also includes a performance panel that evaluates micro- and macro-F scores (upper left corner).

The Cluster View (e) displays a hierarchical clustering of the test set computed by agglomorative clustering based on Euclidean distance between the instances. We use Ward's linkage (Ward, 1963) as the cluster similarity measure, which at each iteration joins the two clusters resulting in the minimal increase in the residual sum of squares with respect to the cluster centroids. A frequently used visualization method for hierarchical clusters are dendrograms (e.g. in Manning et al., 2008; Seo and Shneiderman, 2002). The Cluster View in the FeatureForge prototype uses a radial dendrogram to visualize the clustering of the test instances. The advantage of the radial layout is that it provides more space for the growing number of nodes towards the perimeter of the circle thus providing a better overview of the clustering tree. Instances with identical feature representations are joined before the clustering and form one leaf node in the tree. Each node is labeled with the number of instances that the respective cluster contains. The color of the nodes is chosen according to the signed homogeneity value, which we define as $(\frac{2 \cdot c_1}{c_1 + c_2} - 1)$, where c_1 and c_2 are the number of instances of the two classes in the cluster. The color of the node is the color of c_1 if signed homogeneity is $+1$, and the color of c_2 if its value is -1. If the homogeneity value reaches 0, the node color is red. For values in between ± 1 and 0, the color is interpolated between red and the color of c_1 or the color of c_2, respectively.

The Vector Set View (f) provides information about the attributes of selected instances. Each dimension[1] occupies a row of the table with the following information (in order of the columns): textual description, weights of the classification model, variance of values, occurrences of attribute. The last column is partitioned into three blocks. The middle one displays the fraction of instances that have the respective attribute, as a numerical value and with color being interpolated between white (0%) and black (100%). The left block displays the fraction of them that belong to c_1, and the right block those that belong to c_2. Both blocks are interpolated between white and the color of the respective class. By using the buttons on the bottom right, selected dimensions can be moved to the table on the right. If it contains dimensions, the left table is restricted to the set of instances that have the respective attributes, i.e. it shows the conditional distribution of attributes.

2.2 Interacting with the System

The instances in the views are highlighted (color changes to black) if hovered and selected (black edge color) if clicked by the mouse. Selection and highlight events are propagated from one to all other views. This is known as brushing and linking. The Classifier and the Cluster

[1]We use the term *feature* for the user defined units to describe the data, e.g. a bag-of-words representation for a sentence. *Dimension* describes a position of the resulting vector and *attribute* denotes the respective instance property. Examples for dimensions / attributes are e.g. single terms for a bag-of-words feature.

View also support panning and zooming. While the Primary Data View contains all instances available, the Classifier and the Cluster View only show those from the test set. They ignore highlight or selection events for other instances. If a node in the Cluster View is selected or hovered, all child nodes are highlighted and a selection event is triggered for all instances contained in the respective cluster. In all tables, columns can be sorted arbitrarily, and rows are sorted by any column when clicking on the column header.

The purpose of the Cluster View is to guide users' attention to nodes that are candidates for closer inspection. We hypothesize that instances very similar with respect to their vector representations having heterogeneous gold labels are a symptom of missing discriminatory information or a result of mislabelings. Such nodes have a high heterogeneity level (are colored red) and lie close to the perimeter of the radial tree. This means that the better the discriminatory power of the vector representation, the higher up in the cluster tree nodes with differently labeled instances are created. The heterogeneous nodes thus move higher up in the tree when improving discriminatory power of the vector representation and removing labeling errors. The Classifier View provides insight into how the classification model treats the similar yet heterogeneously labeled instances. If, e.g., the instances are scattered near the decision boundary, classification confidence for them is low which could indicate problems of data sparseness or missing discriminatory information. If they are classified with high confidence, this could be indicative of systematic labeling errors. In case they get assigned correct classes by the classifier, no intervention is necessary, but additional features could help to increase classification confidence. FeatureForge links clustering and classification on a visual and interaction level, but not at an algorithmic level. While this type of linkage is designed to help users explore the properties of a feature set and learn about its problems, we cannot guarantee that all feature or labeling problems can be identified with this approach.

The Vector Set View characterizes selected instances in terms of the defined features by showing in what dimensions they differ and how the attributes are distributed over the instances. This gives a hint about how well certain attributes are an indicator for a class in the selected set. By using the second table, interdependencies of the attributes can be explored, thus offering more fine-grained information about the distribution of attributes.

3 Case Study: Citation Classification

For our case study we concentrate on citation classification (Teufel et al., 2006), where each citation occurring in a scientific text constitutes a classification instance. We enumerate the problems we discovered during an example session and suggest solutions for those findings. The dataset comes from the ACL Anthology Reference Corpus (Bird et al., 2008) and has been annotated by NLP students. We use this corpus because we have developed it ourselves (including annotation guidelines) and are familiar with it. Citations are labeled following the classification scheme of Moravcsik and Murugesan (1975). It is comprised of four facets with two possible labels for each, resulting in four binary classification problems. For the purpose of this illustration we will focus on the facet *conceptual vs. operational*. Conceptual citations contain an idea or concept relevant to the citing paper, while operational citations contain a tool or programming library that the citing paper uses. The complete dataset contains 2009 citations, which we split into a training- and a test-set (80%/20%) on the granularity level of the documents containing the citations, resulting in 1605 training and 404 test instances. We used the Stanford logistic classifier for classification. We loaded an initial feature set containing 32 features of which some are standard features and some are self-developed. Examples of features are a bag-of-words representation for the sentence containing the citation, the position of the citation in that sentence, whether the citation is a constituent of this sentence, and whether the sentence contains any named entity that denotes an NLP tool.

By looking at the Cluster View with all initial features activated, we realized that a high number of citations in heterogeneous nodes on low variance levels occurred within the same sentence. The fact that one of the defined features was a bag-of-words representation of the sentence enclosing the citation and most of the other features also depend on the surrounding sentence resulted in this problem. The Classifier View showed that the classifier had problems separating those instances, even though we created the splits by separating the data on the granularity level of papers, thus citations with identical sentences did not occur in the training and the test set. This problem could be solved by splitting up sentences containing more than one citation as constituents. In the table of the Primary Data View, a parse tree containing the sentence of each citation is stored. It can be used to assign the largest constituent containing the respective citation as the basis for feature extraction. Unfortunately, we were not able to test it right away with the Stanford language, but we disabled the bag-of-words feature for the rest of the session.

The next finding was a cluster of four citations, of which two were labeled operational and two were labeled conceptual. By looking at the Primary Data View containing the sentences with the citations and the citation keys, we saw that one of the citations labeled operational was referring to datasets, the other one was referring to a specific type-hierarchy that the citing paper adopted from the cited paper. The citations labeled conceptual referred to aspects of two different QA systems. Although all of those labels were correct, we realized that it is generally not obvious where to draw the line between a tool borrowed and a concept adopted from another work. The next finding supported the assumption that such instances are problematic for annotators. It was a cluster with two citations, one citing a paper about a classification library and the other referring to a corpus. The citation to the classification library was correctly labeled operational, whereas the citation referring to the corpus was erroneously labeled conceptual. We will thus update the annotation guidelines with the instruction to label datasets used from other publications as operational, and emphasize that conceptual aspects of systems that are adopted by the citing paper are to be marked as conceptual. As examples we can use the citations just found.

Next, we discovered a node with three citations in the Cluster View. One was referring to a Prolog implementation and another was referring to a parser, both labeled correctly as operational. The third citation in the cluster was referring to a set of metrics and was labeled conceptual. This is also a borderline example, for which the annotation guidelines offered no clear guidance, but which should have been labeled operational. We will further refine the guidelines using this citation as an example.

The next cluster that attracted our attention contained three citations. Two of them were correctly labeled as conceptual, and were referring to the classification algorithm used in the work referred to. The third citation referred to the training set that was used in the cited paper. It was labeled correctly as operational. In the Classifier View, however, we could see that the classifier was not able to correctly separate these examples and all of them ended up on the conceptual side of the decision boundary. We realized that the annotation guidelines do not have clear instructions on how to label algorithms that the citing work builds on. They should be labeled as conceptual, and we will update the guidelines consequently. We then took a look at the main verbs of the sentences containing the citation, which were defined as features and thus visible in the Vector Set View. The two sentences that were labeled as conceptual had the main verbs ‘describe’ (with the cited classification algorithm as direct object) and ‘build [on]’ (with the classification algorithm as prepositional object), whereas the operational citation had the main verb ‘use’ (with the training set as the direct object). In the Vector Set View we saw that the classification model had weights pointing to the conceptual subspace for the two verbs ‘describe’ and ‘build’, and a weight pointing to the operational one for ‘use’, which was what we expected the classification algorithm to learn. However, we realized that single verbs were probably not features that offer the right granularity for classification, because they run into data sparseness problems. We thus expect to solve this particular classification problem by using verb classes as features. For this, we will build on verb classes from the VerbNet (Schuler, 2006) project.

4 Future Work

The FeatureForge software is currently under active development. This section presents the most important improvements we plan to implement. The first one is the integration of a more powerful feature definition language. We were not able to directly implement the ideas for new features that we had while exploring our citation data in Section 3. An example of such a more powerful language is the one sketched by (Kobdani et al., 2010). Like the language for the Stanford Classifiers, it is based on primary instance data in a tabular format, but offers much more flexibility to extract and define features based on this data. Furthermore, primary data should not be static during the runtime of the tool. We plan to make this data extensible during runtime with new data columns for each instance, as well as with linguistic tools that can directly be used as a data source. We also plan to support real-valued vectors in the future. Second, we want to develop a measure for the quality of the clustering reflecting the discriminatory power of the features. There exist measures for the quality of a flat clustering (Manning et al., 2008, chapter 16.3). We plan to extend those and derive a measure for the whole clustering tree. We further plan to analyze how our new measures are linked to the classification performance. Furthermore, we aim at a tighter integration of classification and clustering. In the current version of the system this integration takes place on a visual and interaction level, but we plan to explore possibilities to take the hypothesis about the data of the classifier into account for the clustering. Third, we aim at much broader evaluation of the prototype system. Since FeatureForge is tailored to a specific user group of NLP specialists, a quantitative study to investigate the usefulness of our approach will not be possible because it will be hard to find NLP-tasks and corpora with which a large number of NLP specialists are equally familiar. Nevertheless, we plan to conduct an informal study with NLP practitioners to learn about the benefits and deficiencies of the FeatureForge system.

5 Related Work

The contributions to the ACL Feature Engineering Symposium (Ringger, 2005) show that feature engineering plays an important role in many NLP tasks. Carefully and resourcefully engineered features are able to significantly improve classification performance. NLP problems tackled by the symposium contributions include, but are not limited to, text segmentation (Kauchak and Chen, 2005), shallow semantic parsing (Moschitti et al., 2005) and recognition of temporal expressions (Adafre and de Rijke, 2005). There is evidence (Scott and Matwin, 1999) that domain-specific feature engineering is also beneficial for a task such as text classification, where the bag-of-words model is established and shows good performance in many situations. Berend and Farkas (2010) present a set of new and effective features for automatic keyphrase extraction from scientific papers. Those examples all show the importance of linguistic feature creation for statistical NLP applications. Kobdani et al. (2010) identify the creation of feature representations for linguistic instances as the most crucial and costly step in the process of building an NLP application.

There are approaches to feature space examination that build on visualization to give users insight into the usefulness and discriminatory power of feature representations. Kienreich and Seifert (2012) apply matrix reordering algorithms on document-term matrices and classify and discuss typical emerging patterns. With the help of those patterns, users are able to judge the usefulness of certain terms for the discrimination of classes. Schreck and Keim (2006) create a 2D map of an instance space produced by a feature extractor for multimedia datasets. They hypothesize that the uniformity of distances between instances in different clusters and low variance between instances in one cluster correlates with the discriminatory power of a feature representation and provide visualization techniques that help users asses both characteristics. Dolfing (2007) developed a Visual Analytics system for feature engineering for the problem of optical character recognition based on an interactive clustering algorithm that, unlike our approach, directly incorporates user judgment in its clustering decisions. The popular machine learning system WEKA (Hall et al., 2009) offers user interfaces for clustering and classification including visualization to support the user in understanding the data better. WEKA also includes methods for feature selection. Contrary to FeatureForge, however, it does not offer any linking between clustering and classification to support the feature engineering process and offers no solution for corpus improvement.

Feature engineering is related to feature selection, which comprises a number of techniques to automatically reduce the number of dimensions of instance vectors in order to either produce more compact representations, or optimize the representations for a certain classification task. Liu and Yu (2005) provide an overview.

Conclusion

We have presented FeatureForge, a prototype system that offers a fresh approach to feature engineering and corpus improvement. It uses interactive visualization to support NLP researchers and practitioners in two aspects: (i) spot problems with the feature set and define new features to improve discriminatory power and thus classification performance, and (ii) find groups of instances that are hard to label or get systematically mislabeled by the annotators and revise the annotation guidelines accordingly and add newly found instances for better illustration of hard examples. An informal evaluation on a citation corpus showed that our approaches effectively help with those two aspects. By facilitating interactive exploration of corpora with a special focus on the labels of the instances for a specific classification task and the feature representation of the linguistic classification instances, users of FeatureForge are supported during the process of designing features and in addition they can easily spot annotation problems. We expect that future extensions of our approach will support the development of new and even better performing machine-learning based NLP applications.

Acknowledgments

The work presented here is supported by the DFG as part of the Priority Program 1335 *Scalable Visual Analytics*.

References

Adafre, S. F. and de Rijke, M. (2005). Feature engineering and post-processing for temporal expression recognition using conditional random fields. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, FeatureEng'05, pages 9–16.

Berend, G. and Farkas, R. (2010). Sztergak: Feature engineering for keyphrase extraction. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval'10, pages 186–189.

Bird, S., Dale, R., Dorr, B., Gibson, B., Joseph, M., Kan, M.-Y., Lee, D., Powley, B., Radev, D., and Tan, Y. F. (2008). The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of LREC*, pages 1755–1759.

Cook, K. A. and Thomas, J. J. (2005). *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society.

Dolfing, H. (2007). A Visual Analytics Framework for Feature and Classifier Engineering. Master's thesis, Department of Computer and Information Science, University of Konstanz, Konstanz, Germany.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

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

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *SIGKDD Exploration Newsletter*, 11(1):10–18.

Heimerl, F., Koch, S., Bosch, H., and Ertl, T. (2012). Visual classifier training for text document retrieval. In *IEEE VAST'2012*.

Kauchak, D. and Chen, F. (2005). Feature-based segmentation of narrative documents. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, FeatureEng'05, pages 32–39.

Keim, D., Mansmann, F., Schneidewind, J., Thomas, J., and Ziegler, H. (2008). Visual Analytics: Scope and Challenges Visual Data Mining. In *Visual Data Mining*, volume 4404 of *Lecture Notes in Computer Science*, chapter 6, pages 76–90. Springer.

Kienreich, W. and Seifert, C. (2012). Visual Exploration of Feature-Class Matrices for Classification Problems. In *International Workshop on Visual Analytics*, EuroVA'12, pages 37–41.

Kobdani, H., Schütze, H., Burkovski, A., Kessler, W., and Heidemann, G. (2010). Relational feature engineering of natural language processing. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM'10, pages 1705–1708.

Liu, H. and Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):491–502.

Manning, C. and Klein, D. (2003). Optimization, maxent models, and conditional estimation without magic. Tutorial at HLT-NAACL 2003 and ACL 2003.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Moravcsik, M. J. and Murugesan, P. (1975). Some results on the function and quality of citations. *Social Studies of Science*, 5:86–92.

Moschitti, A., Coppola, B., Pighin, D., and Basili, R. (2005). Engineering of syntactic features for shallow semantic parsing. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, FeatureEng'05, pages 48–56.

Ringger, E., editor (2005). *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*. Assiciation for Computational Linguistics.

Russell, D. M., Stefik, M. J., Pirolli, P., and Card, S. K. (1993). The cost structure of sensemaking. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI'93, pages 269–276.

Schreck, T. and Keim, D. (2006). Visual feature space analysis for unsupervised effectiveness estimation and feature engineering. In *In IEEE International Conference on Multimedia and Expo*, ICME'06, pages 9–12.

Schuler, K. K. (2006). *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. PhD thesis, University of Pennsylvania.

Scott, S. and Matwin, S. (1999). Feature engineering for text classification. In *Proceedings of the 16th International Conference on Machine Learning*, ICML'99, pages 379–388.

Seo, J. and Shneiderman, B. (2002). Interactively exploring hierarchical clustering results. *Computer*, 35(7):80–86.

Somol, P., Novovičová, J., and Pudil, P. (2010). *Pattern Recognition Recent Advances*, chapter 4, pages 75–97. InTech.

Teufel, S., Siddharthan, A., and Tidhar, D. (2006). Automatic classification of citation function. In *Proceedings of EMNLP*, pages 103–110.

Ward, J. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244.