KeLP: a Kernel-based Learning Platform for Natural Language Processing

Simone Filice[†], **Giuseppe Castellucci**[‡], **Danilo Croce**[⋆], **Roberto Basili**[⋆]
(†) Dept. of Civil Engineering and Computer Science Engineering
(‡) Dept. of Electronic Engineering
(⋆) Dept. of Enterprise Engineering
University of Roma, Tor Vergata, Italy
{filice,croce,basili}@info.uniroma2.it; castellucci@ing.uniroma2.it

Abstract

Kernel-based learning algorithms have been shown to achieve state-of-the-art results in many Natural Language Processing (NLP) tasks. We present KELP, a Java framework that supports the implementation of both kernel-based learning algorithms and kernel functions over generic data representation, e.g. vectorial data or discrete structures. The framework has been designed to decouple kernel functions and learning algorithms: once a new kernel function has been implemented it can be adopted in all the available kernel-machine algorithms. The platform includes different Online and Batch Learning algorithms for Classification, Regression and Clustering, as well as several Kernel functions, ranging from vector-based to structural kernels. This paper will show the main aspects of the framework by applying it to different NLP tasks.

1 Introduction

Most of the existing Machine Learning (ML) platforms assume that instances are represented as vectors in a feature space, e.g. (Joachims, 1999; Hall et al., 2009; Chang and Lin, 2011), that must be defined beforehand. In Natural Language Processing (NLP) the definition of a feature space often requires a complex feature engineering phase. Let us consider any NLP task in which syntactic information is crucial, e.g. *Boundary Detection* in Semantic Role Labeling (Carreras and Màrquez, 2005). Understanding which syntactic patterns should be captured is non-trivial and usually the resulting feature vector model is a poor approximation. Instead, a more natural approach is operating directly with the parse tree of sentences. Kernel methods (Shawe-Taylor and Cristianini, 2004) provide an efficient and effective solution, allowing to represent data at a more abstract level, while their computation still looks at the informative properties of them. For instance, Tree Kernels (Collins and Duffy, 2001) take in input two syntactic parse trees, and compute a similarity measure by looking at the shared sub-structures.

In this paper, KELP, a Java kernel based learning platform is presented. It supports the implementation of Kernel-based learning algorithms, as well as kernel functions over generic data representations, e.g. vectorial data or discrete structures, such as trees and sequences. The framework has been designed to decouple data structures, kernel functions and learning algorithms in order to maximize the re-use of existing functionalities: as an example, a new kernel can be included inheriting existing algorithms and vice versa. KELP supports XML and JSON serialization of kernel functions and algorithms, enabling the agile definition of kernel-based learning systems without writing additional lines of code. KELP can effectively tackle a wide variety of learning problems. In particular, in this paper we will show how vectorial and structured data can be exploited by KELP in three NLP tasks: *Twitter Sentiment Analysis*, *Text Categorization* and *Question Classification*.

2 Framework Overview

KELP is a machine learning library completely written in Java. The Java language has been chosen in order to be compatible with many Java NLP/IR tools that are developed by the community, such as Stanford CoreNLP[1], OpenNLP[2] or Lucene[3]. KELP is released as open source software under the Apache 2.0 license and the source code is available on github[4]. Furthermore it can be imported via Maven. A detailed documentation of KELP with helpful examples and use cases is available on the website of the Semantic Analytics Group[5] of the University of Roma, Tor Vergata.

In this Section, a closer look at the implementation of different kinds of data representations, kernel functions and kernel-based learning algorithms is provided.

2.1 Data Representations

KELP supports both vectorial and structured data to model learning instances. For example, `SparseVector` can host a Bag-of-Words model, while `DenseVector` can represent data derived from low dimensional embeddings. `TreeRepresentation` can model a parse tree and `SequenceRepresentation` can be adopted to represent sequences of characters or sequences of words. Moreover, the platform enables the definition of more complex forms of data such as pairs, which are useful in modeling those problems where instances can be naturally represented as pairs of texts, such as question and answer in Q/A re-ranking (Severyn and Moschitti, 2013), text and hypothesis in textual entailment (Zanzotto et al., 2009) or sentence pairs in paraphrasing detection (Filice et al., 2015).

2.2 Kernels

Many ML algorithms rely on the notion of similarity between examples. Kernel methods (Shawe-Taylor and Cristianini, 2004) leverage on the so-called kernel functions, which compute the similarity between instances in an implicit high-dimensional feature space without explicitly computing the coordinates of the data in that space. The kernel operation is often cheaper from a computational perspective and specific kernels have been defined for sequences, graphs, trees, texts, images, as well as vectors.

Kernels can be combined and composed to create richer similarity metrics, where information from different `Representations` can be exploited at the same time. This flexibility is completely supported by KELP, which is also easy to extend with new kernels. Among the currently available implementations of kernels, there are various standard kernels, such as `LinearKernel`, `PolynomialKernel` or `RbfKernel`. A large set of kernels specifically designed for NLP applications will be described in the following section.

2.2.1 Kernels for NLP

Many tasks in NLP cannot be properly tackled considering only a Bag-of-Words approach and require the exploration of deep syntactic aspects. In question classification the syntactic information is crucial has largely demonstrated in (Croce et al., 2011). In Textual Entailment Recognition or in Paraphrase Detection a pure lexical similarity between *text* and *hypothesis* cannot capture any difference between *Federer won against Nadal* and *Nadal won against Federer*. A manual definition of an artificial feature set accounting for syntax is a very expensive operation that requires a deep knowledge of the linguistic phenomena characterizing a specific task. Moreover, every task has specific patterns that must be considered, making a manual feature engineering an extremely complex and not portable operation. How can linguistic patterns characterizing a question be automatically discovered? How can linguistic rewriting rules in paraphrasing be learnt? How can semantic and syntactic relations in textual entailment be automatically captured? An elegant and efficient approach to solve NLP problems involving the usage of syntax is provided by tree kernels (Collins and Duffy, 2001). Instead of trying to design a synthetic feature space, tree kernels directly operate on the parse tree of sentences evaluating the tree fragments shared by two trees. This operation implicitly corresponds to a dot product in the feature space of all possible tree fragments. The dimensionality of such space is extremely large and operating directly on it is not viable.

Many tree kernels are implemented in KELP, and they differ by the type of tree fragment considered in the evaluation of the matching structures. In the `SubTreeKernel` (Collins and Duffy, 2001) valid fragments are subtrees (ST), i.e. any node of a tree along with all its descendants. A subset tree (SST) exploited by the `SubSetTreeKernel` is a more general structure since its leaves can be non-

[1] http://nlp.stanford.edu/software/corenlp.shtml
[2] https://opennlp.apache.org/
[3] http://lucene.apache.org/
[4] https://github.com/SAG-KeLP
[5] http://sag.art.uniroma2.it/demo-software/kelp/

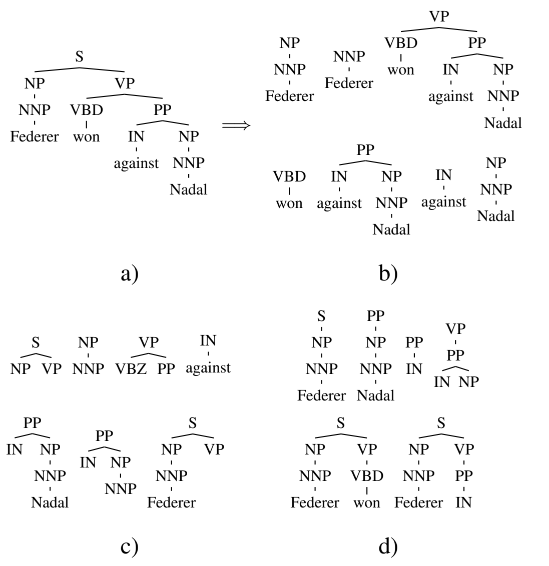

Figure 1: a) Constituent parse tree of the sentence *Federer won against Nadal*. b) some subtrees. c) some subset trees. d) some partial trees.

terminal symbols. The SSTs satisfy the constraint that grammatical rules cannot be broken. `PartialTreeKernel` (Moschitti, 2006) relaxes this constraint considering partial trees (PT), i.e. fragments generated by the application of partial production rules. Examples of different kinds of tree fragments are shown in Figure 1. The `SmoothedPartialTreeKernel` (SPTK) (Croce et al., 2011) allows to match those fragments that are not identical but that are semantically related, by relying on the similarity between lexical items, e.g. by applying a word similarity metric (e.g. WordNet or word embeddings similarities). The adopted implementation allows to easily extend the notion of similarity between nodes, enabling the implementation of more expressive kernels, as the Compositionally Smoothed Partial Tree Kernel (CSPTK) that embeds algebraic operators of Distributional Compositional Semantics (Annesi et al., 2014). Moreover, the `SequenceKernel` (Bunescu and Mooney, 2005) is included in the library, and it allows to compare two texts evaluating the number of common sub-sequences. This implicitly corresponds to operate on the space of all possible N-grams. Kernels operating over pairs, such as the `PreferenceKernel` (Shen and Joshi, 2003) for re-ranking, are also included in KELP.

2.3 Machine Learning Algorithms

In ML, a plethora of learning algorithms have been defined for different purposes, and many variations of the existing ones as well as completely new learning methods are often proposed. KELP provides a large number of learning algorithms[6] ranging from batch, e.g. Support Vector Machines (Vapnik, 1995), to online learning models, e.g. `PassiveAggressive` algorithms (Crammer et al., 2006), and from linear to kernel-based methods, for tackling classification, regression or clustering tasks. Moreover, algorithms can be composed in meta-learning schemas, like multi-class classification (e.g. `One-VS-One` and `One-VS-All`, (Rifkin and Klautau, 2004)) and multi-label classification, or can be combined in ensembles. A simple interface taxonomy allows to easily extend the platform with new custom learning algorithms. A complete support for tackling NLP tasks is thus provided. For example, in scenarios where the syntactic information is necessary for achieving good accuracy, `C-SVM` or ν-`SVM` (Chang and Lin, 2011) operating on trees with kernels can be effectively applied. When dealing with large datasets, many efficient learning algorithm can be adopted, like linear methods, e.g. `Pegasos` (Shalev-Shwartz et al., 2007) or `LibLinear`, (Fan et al., 2008), or like budgeted kernel-based algorithms, e.g. `RandomizedPerceptron` (Cesa-Bianchi and Gentile, 2006).

Listing 1: A JSON example.

```
{"algorithm" : "oneVsAll",
  "baseAlgorithm" : {
    "algorithm" : "binaryCSvmClassification",
    "c" : 10,
    "kernel" : {
      "kernelType" : "linearComb",
      "weights" : [1,1],
      "toCombine" : [
        {
          "kernelType" : "norm",
          "baseKernel" : {
            "kernelType" : "ptk",
            "mu" : 0.4,
            "lambda" : 0.4,
            "representation" : "parseTree"
          }
        },
        {
          "kernelType" : "linear",
          "representation" : "Bag-of-Words"
        }
      ]
    }
  }
}
```

2.4 A JSON example

Kernel functions and algorithms are serializable in JSON or XML. This is useful for instantiating a new algorithm without writing a single line of Java

[6]All the algorithms are completely re-implemented in Java and they do not wrap any external library

code, i.e. the algorithm description can be provided in JSON to an interpreter that will instantiate it. Listing 1 reports a JSON example of a kernel-based Support Vector Machine operating in a one-vs-all schema, where a kernel linear combination between a normalized Partial Tree Kernel and a linear kernel is adopted. As the listing shows kernels and algorithms can be easily composed and combined in order to create new training models.

3 Case Studies in NLP

In this Section, the functionalities and use of the learning platform are shown. We apply KELP to very different NLP tasks, i.e. *Sentiment Analysis in Twitter*, *Text Categorization* and *Question Classification*, providing examples of kernel-based and linear learning algorithms. Further examples are available on the KELP website[7] where it is shown how to instantiate each algorithm or kernel via JSON and how to add new algorithms, representations and kernels.

3.1 Sentiment Analysis in Twitter

The task of Sentiment Analysis in Twitter has been proposed in 2013 during the SemEval competition (Nakov et al., 2013). We built a classifier for the subtask B, i.e. the classification of a tweet with respect to the *positive*, *negative* and *neutral* classes. The contribution of different kernel functions is evaluated using the Support Vector Machine learning algorithm. As shown in Table 1, we apply linear (Lin), polynomial (Poly) and Gaussian (Rbf) kernels on two different data representations: a Bag-Of-Words model of tweets (BoW) and a distributional representation (WS). The last is obtained by linearly combining the distributional vectors corresponding to the words of a message; these vectors are obtained by applying a Skip-gram model (Mikolov et al., 2013) with the *word2vec* tool[8] over 20 million of tweets. The linear combination of the proposed kernel functions is also applied, e.g. Poly$_{Bow}$+Rbf$_{WS}$. The mean F1-measure of the *positive* and *negative* classes (`pn`)[9] as well as of all the classes (`pnn`) is shown in Table 1.

3.2 Text Categorization

In order to show the scalability of the platform, a second evaluation considers linear algorithms.

[7] `http://sag.art.uniroma2.it/demo-software/kelp/`
[8] `https://code.google.com/p/word2vec/`
[9] `pn` was the official metric of the SemEval competition.

Kernel	**MeanF1(pn)**	**MeanF1(pnn)**
Lin$_{BoW}$	59.72	63.53
Poly$_{BoW}$	54.58	59.27
Lin$_{WS}$	60.79	63.94
Rbf$_{WS}$	61.68	65.05
Lin$_{BoW}$+Lin$_{WS}$	66.12	68.56
Poly$_{BoW}$+Rbf$_{WS}$	64.92	68.10

Table 1: Results of Sentiment Analysis

We selected the Text Categorization task on the RCV1 dataset (Lewis et al., 2004) with the setting that can be found on the LibLinear website[10]. In this version of the dataset, *CCAT* and *ECAT* are collapsed into a positive class, while *GCAT* and *MCAT* are the negative class, resulting in a dataset composed by $20,242$ examples. As shown in Table 2, we applied the LibLinear, Pegasos and Linear Passive-Aggressive implementations, computing the accuracy and the standard deviation with respect to a 5-fold cross validation strategy.

Task	**Accuracy**	**Std**
LibLinear	96.74%	0.0029
Pegasos	95.31%	0.0033
Passive Aggressive	96.60%	0.0024

Table 2: Text Categorization Accuracy

3.3 Question Classification

The third case study explores the application of Tree Kernels to Question Classification (QC), an inference task required in many Question Answering processes. In this problem, questions written in natural language are assigned to different classes. A QC system should select the correct class given an instance question. In this setting, Tree Kernels allow to directly model the examples in terms of their parse trees. The reference corpus is the UIUC dataset (Li and Roth, 2002), including 5,452 questions for training and 500 questions for test[11], organized in six coarse-grained classes, such as `HUMAN` or `LOCATION`. Again, Kernel-based SVM has been evaluated adopting the same setup of (Croce et al., 2011). A pure lexical model based on a linear kernel over a Bag-of-Words (BoW) is considered a baseline. The contribution of the syntactic information is demonstrated by the results achieved by the Partial Tree Kernel (PTK), the Smoothed Partial Tree Kernels (SPTK) and the Compositionally Smoothed Partial Tree Kernel (CSPTK), as shown in Table 3.

[10] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`
[11] `http://cogcomp.cs.illinois.edu/Data/QA/QC/`

Kernel	Accuracy
BoW	87.2%
Poly$_{BoW}$	88.8%
PTK	91.6%
SPTK	94.6%
CSPTK	95.0%

Table 3: Question Classification Accuracy.

4 Related Work

Many software tools for computational linguistic research already exist. Tools like Stanford CoreNLP or OpenNLP provide a complete pipeline for performing linguistic tasks such as stemming, lemmatization, Part-of-Speech tagging or parsing. They are complementary to KELP: they can be used in the feature extraction phase, while KELP will care about the machine learning part. Regarding other machine learning platforms there are plenty of available possibilities, but for different reasons no one can provide something close to what the proposed library offers.

Weka (Hall et al., 2009) is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from Java. It contains various tools for different data mining activities: data pre-processing, classification, regression, clustering and visualization.

Mallet (McCallum, 2002) is more oriented to NLP applications. It is entirely in Java and includes feature extraction tools for converting text into vectors and statistical analysis tools for document classification, clustering, topic modeling, information extraction, and other machine learning applications to text. Regarding the kernel-based learning both Weka and Mallet leverage on LibSVM, and obviously inherit its limits.

LibSVM (Chang and Lin, 2011) is a machine learning platform focusing on Support Vector Machines. It is written in C++ language and it includes different SVM formulations: `C-svm`, `Nu-svm` and `OneClass-svm`, as well as a one-vs-one multi classification schema. It implements also regression support vector solvers. It has been ported in different languages, including Java. The batch learning part of KELP is strongly inspired by LibSVM formulations and implementations. LibSVM is mainly intended for plain users and does not provide any support for extendibility. It can operate only on sparse feature vectors via standard kernel functions. No structured representations are considered.

Another very popular Support Vector Machines (SVM) package is SvmLight (Joachims, 1999). It is entirely written in C language and its main feature is speed. It solves classification and regression problems, as well as ranking problems. Its efficiency is paid in terms of extensibility: C language does not allow a fast prototyping of new machine learning kernels or algorithms. Many times in research contexts fast prototyping is more important than performances: the proposed platform has been developed with extensibility in mind.

The most similar platform to ours is JKernelMachines (Picard et al., 2013). It is a Java based package focused on Kernel machines. Just like the proposed library, JKernelMachines is primary designed to deal with custom kernels that cannot be easily found in standard libraries. Standard SVM optimization algorithms are implemented, but also more sophisticated learning-based kernel combination methods such as Multiple Kernel Learning (MKL). However, many features covered by KELP are not offered by JKernelMachines, just like tree kernels, regression and clustering. Moreover, different architectural choices have been applied in KELP in order to support an easier composition and combination of representations, kernels as well as learning algorithms.

5 Conclusions

This paper presented KELP, a Java framework to support the application of Kernel-based learning methods with a particular attention to Language Learning tasks. The library implements a large variety of kernel functions used in NLP (such as Tree Kernels or Sequence Kernels) as well as many learning algorithms useful in classification, regression, novelty detection or clustering problems. KELP can be imported via Maven but its usage is not restricted to a Java-compliant environment as it allows to build complex kernel machine based systems, leveraging on JSON/XML interfaces to instantiate classifiers. The entire framework has been designed to support researchers in the development of new kernel functions or algorithms, providing a principled decoupling of the data structures in order to maximize the re-use of existing functionalities. The benefits of the proposed environment have been shown in three NLP tasks, where results in line with the state-of-the-art have been reached with the simple application of various kernel functions available in KELP.

References

Paolo Annesi, Danilo Croce, and Roberto Basili. 2014. Semantic compositionality in tree kernels. In *Proc. of CIKM 2014*, pages 1029–1038, New York, NY, USA. ACM.

Razvan C. Bunescu and Raymond J. Mooney. 2005. Subsequence kernels for relation extraction. In *NIPS*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 152–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nicolò Cesa-Bianchi and Claudio Gentile. 2006. Tracking the best hyperplane with a simple budget perceptron. In *In Proc. of the 19th Annual Conference on Computational Learning Theory*, pages 483–498. Springer-Verlag.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *NIPS*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 7:551–585, December.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *EMNLP*, Edinburgh.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015. Structural representations for learning relations between pairs of texts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, Beijing, China, July. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: An update. *sigkdd explor.*, 11(1).

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, December.

X. Li and D. Roth. 2002. Learning question classifiers. In *Proceedings of ACL '02*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*, Berlin, Germany, September.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of SemEval 2013*, pages 312–320, Atlanta, USA. ACL.

David Picard, Nicolas Thome, and Matthieu Cord. 2013. Jkernelmachines: A simple framework for kernel machines. *Journal of Machine Learning Research*, 14:1417–1421.

Ryan Rifkin and Aldebaro Klautau. 2004. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, December.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *Proceedings of the 2013 Conference on EMNLP*, pages 458–467, Seattle, USA. ACL.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. 2007. Pegasos: Primal estimated sub–gradient solver for SVM. In *Proc. of ICML*.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Libin Shen and Aravind K. Joshi. 2003. An svm based voting algorithm with application to parse reranking. In *In Proc. of CoNLL 2003*, pages 9–16.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Fabio massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *Nat. Lang. Eng.*, 15(4):551–582, October.