HOO 2012 Shared Task: UKP Lab System Description

Torsten Zesch[†‡] **and Jens Haase**[†]

[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt

[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research and Educational Information

www.ukp.tu-darmstadt.de

Abstract

In this paper, we describe the UKP Lab system participating in the HOO 2012 Shared Task on preposition and determiner error correction. Our focus was to implement a highly flexible and modular system which can be easily augmented by other researchers. The system might be used to provide a level playground for subsequent shared tasks and enable further progress in this important research field on top of the state of the art identified by the shared task.

1 Introduction

UKP Lab already participated in the previous HOO Shared Task in 2011. Our knowledge-based system (Zesch, 2011) was targeted towards detecting real-word spelling errors, but performed also well on a number of other error classes.[1] However, it was not competitive for article and preposition errors where supervised systems based on confusion sets constitute the state of the art. Thus, we tailor the HOO 2011 system towards correcting article and preposition errors, but also implement a supervised approach based on confusion sets (Golding and Schabes, 1996; Jones and Martin, 1997; Carlson et al., 2001).

We decided to implement a basic system that should be as flexible as possible and might serve as a basis for experiments in future rounds of the shared task. We also plan to model the most successful systems in our framework as soon as the system descriptions are made available.[2] This might provide a level playground for subsequent shared tasks and enable real progress in this important field on top of the state of the art identified by the HOO shared task.

2 Supervised Error Detection

We implement a generic framework for article and preposition error detection based on the open-source DKPro framework.[3] DKPro is a collection of software components for natural language processing based on the Apache UIMA framework (Ferrucci and Lally, 2004). It comes with a collection of ready-made modules which can be combined to form more complex applications.

Our goal is to develop a system which is as flexible as possible with respect to (i) linguistic preprocessing, (ii) the extraction of features, and (iii) the applied classification method. We will make the source code publicly available as part of the DKPro infrastructure and hope that this will lower the obstacles for participating in future rounds of the HOO Shared Task.

We also provide a reference implementation of the HOO 2012 experiments based on the DKPro Lab framework (Eckart de Castilho and Gurevych, 2011) which enables (i) parameter sweeping, (ii) modeling of interdependent tasks (like e.g. training and test cycles), (iii) generating performance reports, and (iv) storing all experimental results in a convenient manner.

[1] http://clt.mq.edu.au/research/projects/hoo/hoo2011/reports/hoo2011-UDposter.pdf

[2] We invite other participating teams to help with this effort.

[3] http://code.google.com/p/dkpro-core-asl/

The 7th Workshop on the Innovative Use of NLP for Building Educational Applications, pages 302–306,
Montréal, Canada, June 3-8, 2012.

2.1 Linguistic Preprocessing

For our basic implementation, we only use a few preprocessing steps. We tokenize and sentence split the data with the default DKPro segmenter, and then use TreeTagger (Schmid, 2004) to POS-tag and chunk the sentences. However, the framework allows the effortless addition of other preprocessing components, e.g. parsing or named-entity recognition.

2.2 Feature Extraction

We implement a generic feature extraction process based on the ClearTK project (Ogren et al., 2008). ClearTK provides a set of highly flexible feature extractors that access the annotations (e.g. POS tags, chunks, etc.) created by the linguistic preprocessing.

One important decision during training is to decide which instances should be used for feature extraction. In the simplest setting, each token is used to generate an instance, but this would result in a very high number of negative instances for every positive instance. For the error classes RT/UT and RD/UD, a more balanced distribution of instances can be easily enforced by only creating a positive instance if the token equals an element in the corresponding confusion set. We create a negative instance by removing or changing the article/preposition.

For articles, we use the confusion set:

{a, an, the, this}[4]

For prepositions, we use the confusion set:

{as, at, but, by, for, from, in, of, on, out, over, since, than, to, up, with}

The confusion set is a parameter to the feature extraction method and can be changed easily. This also makes it possible to apply the framework to other error classes, e.g. for correcting frequently confused words like (accept, except) or (than, then).

Table 1 lists the set of basic features implemented in the reference system. As our goal was to implement a highly flexible system, we put more effort in the overall architecture than in the feature engineering. N-gram features are computed based on the Google Web1T n-gram corpus (Brants and Franz, 2006) which is accessed using jWeb1T.[5]

The listed features can be improved in many ways, e.g. the chunk feature could also encode the type of the chunk. As the framework allows to easily add new feature extractors, we are going to integrate the most successful features from the shared task. Due to the modular architecture of ClearTK, the implemented feature extractors could even be re-used for other classification tasks unrelated to spelling correction.

2.3 Classification

ClearTK provides a wide range of adapters to well known machine learning frameworks and classification tools. As of April 2012, the following adapters are supported:

- LIBSVM[6]
- MALLET[7] (McCallum, 2002)
- OpenNLP Maxent[8]
- SVM$^{\text{light}}$[9] (Joachims, 1999)
- SVM$^{\text{light}}$-TK[10] (Moschitti, 2006)
- Weka[11] (Hall et al., 2009)

As we can easily switch the classifier, we tried a wide range of classifiers, but SVM worked generally best. For the official runs, we used SVM as implemented in the Weka toolkit with the parameter "BuildLogisticModels" which allows to base a detection decision on the confidence of the classifier in order to improve precision.

3 Knowledge-based Error Detection

Besides the supervised system described above, we also apply our knowledge-based system from the HOO 2011 Pilot Round (Zesch, 2011). We re-implemented two state-of-the-art approaches: the

[4] In the official runs, *an* was not part of the confusion set, but was specially handled in a post-processing step. In the current version of the framework, we removed this heuristic and now treat *an* as a normal part of the confusion set.

[5] code.google.com/p/jweb1t/

[6] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[7] http://mallet.cs.umass.edu/

[8] http://opennlp.apache.org/

[9] http://svmlight.joachims.org/

[10] http://disi.unitn.it/moschitti/Tree-Kernel.htm

[11] http://www.cs.waikato.ac.nz/ml/weka/

Name	Description	Range of Values / Examples
pos_{-2-1}	The neighboring POS tags. For the example we assume "IN NNP *DT* NN VBD".	IN-NNP
pos_{-2}		IN
pos_{-1}		NNP
pos_{+1}		NN
pos_{+2}		VBD
pos_{+1+2}		NN-VBD
$chunk_{-1}$	Whether the neighboring tokens are part of a chunk. For the example, we assume "in *the* [United States]".	O
$chunk_{+1}$		B
$chunk_{+2}$		I
$chunk_{+1+2}$		B-I
$vowel_{+1}$	Whether the next token starts with a vowel or not.	0/1
$cons_{+1}$	Whether the next token starts with a consonant or not.	0/1
$sign_{+1}$	Any sign that is not an alphabetic character.	0/1
n-gram$(t_{-1}\{x \rightarrow y\})$	Let f(n-gram) be the frequency of the n-gram in a certain corpus. All n-gram features are then computed as $\frac{f(xt_{+1}t_{+2})}{f(t_{+1}t_{+2})} - \frac{f(yt_{+1}t_{+2})}{f(t_{+1}t_{+2})}$	n-gram("{the $\rightarrow$ a} big house"); f("the big house") $= 100$; f("a big house") $= 50$; f("big house") $= 1000$; $\frac{100}{1000} - \frac{50}{1000} = 0.05$
n-gram$(\{x \rightarrow y\}t_{+1})$		
n-gram$(t_{-1}\{x \rightarrow y\}t_{+1})$		
n-gram$(t_{-2}t_{-1}\{x \rightarrow y\})$		
n-gram$(\{x \rightarrow y\}t_{+1}t_{+2})$		

Table 1: List of features used for classification.

knowledge-based approach (Hirst and Budanitsky, 2005) and the statistical approach (Mays et al., 1991; Wilcox-OHearn et al., 2008). Both approaches measure the contextual fitness of a word and the surrounding context. For that purpose, the *knowledge-based approach* computes the semantic relatedness of a target word with all other words in a certain context window. This approach is not suitable for correcting article or preposition errors, as these word classes are not linked to the context via lexical-semantic relations. Thus, we only use the *statistical approach* that computes the probability of a sentence based on a n-gram language model. We use the Google Web1T n-gram data (Brants and Franz, 2006).

Although being generally applicable to article and preposition errors, the statistical approach needs some adaptations in order to achieve acceptable performance. In the original definition, the approach computes the probability of all alternative sentences where the target word is replaced with a word from the vocabulary that has low edit distance to the target word. This results in a very high false detection rate. Thus, we (i) limit detections to positions where an article or preposition is already present, and (ii) select the substitution candidate not from all tokens with low edit distance to the original token, but only from the appropriate confusion set.

As the statistical approach is purely based on n-gram frequencies, while this is only one feature of the supervised approach, we expect the supervised approach to outperform our adapted knowledge-based system by a wide margin.

4 Experimental Setup

We model all experiment pipelines in the previously described framework. As training data, we use the publicly available Brown corpus (Francis W. Nelson and Kuçera, 1964), but limit training to 3,700 randomly selected sentences in order to speed up the training process.

4.1 Unofficial Runs

Due to technical problems, we were not able to submit all runs in time. We therefore report also unofficial runs which we evaluated on the test data that was available for participants for a limited amount of time.[12] Although we did not tailor the unofficial runs in any way towards the test data, they have certainly a different status than the official runs. We do not consider this as a major problem, as our basic

[12]HOO 2012 test data was subject to a strict license and needed to be deleted after the evaluation period.

	Description	Run	Detection			Recognition			Correction		
			P	R	F	P	R	F	P	R	F
Baselines	Always *the*	-	7.09	6.13	6.58	7.09	6.13	6.58	2.00	1.69	1.81
	Always *of*	-	11.51	28.54	16.40	11.51	28.54	16.40	1.53	3.81	2.19
Unofficial	2011 Articles; $\alpha = .005$	-	9.62	8.47	9.00	9.62	8.47	9.00	0.96	0.85	0.90
	2011 Prepositions; $\alpha = .005$	-	18.11	23.04	20.28	18.11	23.04	20.28	9.00	11.42	10.05
	2012 Naive Bayes	-	9.35	39.32	15.11	9.35	39.32	15.11	1.26	5.29	2.03
	2012 SVM	-	10.46	33.40	15.93	10.46	33.40	15.93	2.71	8.67	4.13
Official	$\theta_{RD} = 0.95; \theta_{RT} = 0.8$	UD0	8.64	7.73	8.16	4.94	4.42	4.66	1.48	1.32	1.40
	$\theta_{RD} = 0.8; \theta_{RT} = 0.7$	UD1	8.36	15.45	10.85	4.18	7.73	5.43	1.19	2.21	1.55
	$\theta_{RD} = 0.5; \theta_{RT} = 0.3$	UD2	8.94	31.13	13.88	5.51	19.21	8.57	1.20	4.19	1.87

Table 2: HOO 2012 test data: Results (in %) for article and preposition errors combined.

feature set is not competitive with the best performing systems anyway.

We implemented two baseline systems, one for articles and one for prepositions. The baselines replace every occurrence of an article/preposition with the most frequent article/preposition from the confusion set (*the* for articles, *of* for prepositions).

We also apply the adapted HOO 2011 statistical approach in two versions as described above: one adapted towards articles, and one adapted towards prepositions.

Finally, we use the new framework for supervised error correction based on the basic feature set described above with two classifiers: Naive Bayes and SVM as implemented in the Weka toolkit version 3.7.5. We treat the correction task as a multi-class problem and only target the error classes RD, RT, UD, UT. The remaining error classes MD and MT (missing articles and prepositions) are more challenging, as it is less obvious how to create good training data from a non-error annotated corpus.

4.2 Official Runs

The three runs that were officially submitted are also based on the SVM implementation in Weka, but we applied the parameter "BuildLogisticModels" which allows to base a detection decision on the confidence of the classifier in order to improve precision. We tuned parameters on the training data and report three runs for the threshold combinations $(\theta_{RD}, \theta_{RT})$ = (0.95, 0.8), (0.8, 0.7), and (0.5, 0.3).

5 Results

Table 2 summarizes the results of all runs. As expected, the basic feature set used in our experiments is not competitive with the top-performing systems in the shared task.[13] However, some observations can be made from the relative differences between the scores. The thresholds applied in the official runs are not working as expected, as precision is not influenced, while recall drops a lot. The HOO 2011 system based on the statistical approach performs quite well for prepositions, but not for articles. Its performance is comparable to the supervised runs, but this is only due to the limited feature set used in our experiment.

As mentioned above, our focus was to implement a highly flexible and modular system for supervised error correction which can be easily augmented by other researchers. We plan to model the most successful systems in our framework as soon as the system descriptions are made available, and we invite other participating teams to help with this effort. The system might provide a level playground for subsequent shared tasks and enable further progress in this important field of research.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806.

[13] The best performing systems achieve about 40% F-score for detection, 35% for recognition, and 28% for correction. See (Dale et al., 2012) for an overview of the results.

References

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram Version 1.

Andrew J Carlson, Jeffrey Rosen, and Dan Roth. 2001. Scaling Up Context-Sensitive Text Correction. In *Proceedings of IAAI*.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A Report on the Preposition and Determiner Error Correction Shared Task. In *Proceedings of the Seventh Workshop on Innovative Use of NLP for Building Educational Applications*, Montreal, Canada.

Richard Eckart de Castilho and Iryna Gurevych. 2011. A lightweight framework for reproducible parameter sweeping in information retrieval. In *Proceedings of the 2011 workshop on Data infrastructurEs for supporting information retrieval evaluation (DESIRE '11)*, New York, NY, USA. ACM.

David Ferrucci and Adam Lally. 2004. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, 10(3-4):327–348.

Francis W. Nelson and Henry Kuçera. 1964. Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers.

Andrew R. Golding and Yves Schabes. 1996. Combining Trigram-based and feature-based methods for context-sensitive spelling correction. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 71–78, Morristown, NJ, USA. 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 Explorations*, 11(1).

Graeme Hirst and Alexander Budanitsky. 2005. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(1):87–111, March.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*.

Michael P Jones and James H Martin. 1997. Contextual spelling correction using latent semantic analysis. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 166–173, Morristown, NJ, USA. Association for Computational Linguistics.

Eric Mays, Fred. J Damerau, and Robert L Mercer. 1991. Context based spelling correction. *Information Processing & Management*, 27(5):517–522.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. In *Proceedings of the Eleventh International Conference on European Association for Computational Linguistics*, Trento, Italy.

Philip V. Ogren, Philipp G. Wetzler, and Steven Bethard. 2008. ClearTK: A UIMA Toolkit for Statistical Natural Language Processing. In *Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP workshop at Language Resources and Evaluation Conference (LREC)*.

Helmut Schmid. 2004. Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vectors. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland.

Amber Wilcox-OHearn, Graeme Hirst, and Alexander Budanitsky. 2008. Real-word spelling correction with trigrams: A reconsideration of the Mays, Damerau, and Mercer model. In *Proceedings of the 9th international conference on Computational linguistics and intelligent text processing (CICLing)*.

Torsten Zesch. 2011. Helping Our Own 2011: UKP Lab System Description. In *Proceedings of the Helping Our Own Working Group Session at the 13th European Workshop on Natural Language Generation*, pages 260–262.