Exploiting Wiktionary for Lightweight Part-of-Speech Tagging for Machine Learning Tasks

Mario Zechner, Stefan Klampfl, and Roman Kern
Know-Center GmbH
Graz, Austria
Email: {mzechner,sklampfl,rkern}@know-center.at

***Abstract*—Part-of-speech (PoS) tagging is a crucial part in many natural language machine learning tasks. Current state-of-the-art PoS taggers exhibit excellent qualitative performance, but also contribute heavily to the total runtime of text preprocessing and feature generation, which makes feature engineering a time-consuming task. We propose a lightweight dictionary and heuristics based PoS tagger that exploits Wiktionary as its information source. We demonstrate that its application to natural language machine learning tasks considerably decreases the feature generation runtime, while not degrading the overall performance on these tasks. We compare the lightweight tagger to a state-of-the-art maximum entropy based PoS tagger in clustering and classification tasks and evaluate its performance on the Brown Corpus. Finally, we explore future research scenarios where our tagger and Wiktionary lookup enables efficient processing of big data due to the significant decrease in runtime.**

***Keywords*—Machine learning; feature engineering; natural language processing; part-of-speech tagging; big data.**

I. INTRODUCTION

Large scale corpora of unstructured natural language text, as found on the web or in enterprise document management systems, are common application fields for various supervised and unsupervised machine learning algorithms. The goal in most of these scenarios is to extract structured information from the unstructured text. This can include global structure derived from clustering, local structure as detected by topic segmentation, or meta-data such as authors, named entities, facts, or genre.

In many of these scenarios, the unstructured text is first preprocessed and transformed into a suitable feature representation for the machine learning algorithm used. A common feature representation is the classical text vector space or bag-of-words model [1]. In this space, each term in the corpus is represented by an own dimension. A text is transformed into this vector space by counting the frequency of each term it contains and storing these frequencies in the corresponding components of the vector. In order to counteract certain artifacts, such as differences in the length of documents, these raw term-frequency vectors are often additionally weighted using various schemes such as term frequencyŰinverse document frequency (TF-IDF) [2] or BM25 [3].

In addition to weighting term vectors, terms can be pruned by not taking them into account when transforming texts to term vectors. Specific word categories do not transport the concepts or meanings of a text and are thus often omitted from the feature representation. One way to prune these words is via stop-word lists [4], which are usually non-exhaustive. A more sophisticated approach is part-of-speech tagging, which assigns a word category such as noun, verb or adjective, to each term in the document. Instead of specific words, entire word categories are omitted, which are unlikely to encode concepts, such as determiners, particles, verbs and so on. Nouns and proper nouns are the most likely word categories to transport meaning, and it is usually terms from these categories that get included into the feature representation of a text document.

Part-of-speech (PoS) tagging has been an active research topic over the last two decades. Various approaches have been devised, from rule-based systems [5], to different statistical approaches via Hidden Markov Models [6], Maximum Entropy Models [7], Conditional Random Field (CRF) models [8] or Support Vector Machines (SVM) [9]. Especially the statistical approaches exhibit excellent accuracy. However, these approaches also incur a considerable increase in runtime, and often the feature extraction based on PoS tags can take longer than the actual task at hand. Moreover, feature engineering becomes more cumbersome as modifications of and experimentation with the feature engineering strategy necessitates a rerun of the feature extraction stage.

For big data scenarios arising from web corpora, this increase in runtime is a big hindrance. Processing times can only be decreased at best in a linear manner by adding more machines to solve the problem. In research scenarios hardware budgets are bounded, so a different strategy that decreases the time complexity of tagging is preferable.

Large scale corpora as described above are not only problematic due to their size, but also due to the fact that they are often multi-lingual. While English PoS tagging models are available, PoS tagging models for other languages are harder to obtain. This can be attributed to the lack of training corpora through which statistical PoS tagging models can be trained. Therefore, a tagging system that does not rely on training corpora would be desirable.

We hypothesize that a dictionary and heuristics based tagging approach is sufficient in quality for the above described application scenarios if its recall is comparable to that of more sophisticated methods. Furthermore, this tagging approach should have a considerable edge over more sophisticated approaches in terms of runtime performance. In this paper, we exploit Wiktionary [10] as a freely available, multi-lingual information source which allows us to tackle the problem in an efficient and cheap manner.

 ISBN: 978-1-61208-295-0

Contribution

Our contribution consists of the following:

1) A lightweight, dictionary and heuristics based PoS tagger based on Wiktionary, that is fast, sufficiently accurate and cheaply adaptable to other languages.
2) An evaluation of PoS tagging and runtime performance on the Brown Corpus relative to a state-of-the-art tagger, allowing us to estimate its performance when used for the feature engineering stage of machine learning tasks.
3) An evaluation of the PoS tagger as part of the feature engineering stage of a text clustering task, showing that using the lightweight tagger decreases the overall runtime of the scenario considerably, while retaining the same quality as achieved with the state-of-the-art tagger.
4) An evaluation of the PoS tagger as part of a text classification task, again showing that the lightweight tagger decreases the overall runtime while retaining the same quality as achieved with the state-of-the-art tagger.
5) A discussion of potential applications and implications on big data tasks.

II. WIKTIONARY-BASED PART-OF-SPEECH TAGGING

Our lightweight PoS tagger uses Wiktionary as an information source. Wiktionary is a freely available, multilingual dictionary, thesaurus, and phrase book. It is edited by volunteers all over the world and currently contains dictionaries for 170 languages [11]. Wiktionary has an exhaustive list of criteria for inclusion of a word [12] and aims to capture common vocabulary. Proper nouns arising from person and company names, places and other named entities are included with specific caveats.

An article describing a term in Wiktionary generally contains information about a term's part-of-speech, word sense, pronunciation and so on. Usage examples are also often provided, as well as synonyms, antonyms, hypernyms and hyponyms. In addition, different spelling variations as well as reflections of a term are present.

This information varies across articles for one language, and between the collections of the 170 languages found in Wiktionary. The number of terms found in a language collection is also varying heavily depending on the language, from a few millions to a few dozens. Table I lists the top 10 languages and their individual article count at the time of writing. This shows that Wiktionary contains many large corpora for various languages. In this work we focus on English; however, performing the following experiments on corpora of other languages is easily possible, since building a lightweight tagger for a different languages involves parsing the corresponding articles in Wiktionary. This task is far less labour-intense than manually tagging a sufficient amount of training data.

As a first step we have built a parser for Wiktionary articles that extracts each word's forms, possible part-of-speech tags, synonyms, hyponyms, hypernyms and translations. For most words in Wiktionary, all its possible inflexions, e.g., "see", "saw", "seen" etc., are usually present. It should be noted that Wiktionary also contains multi-token phrases. For this work, we omit all such phrases.

Table I. THE TOP TEN LANGUAGES BY ARTICLE COUNT IN WIKTIONARY. THE LARGE AMOUNT OF ARTICLES AVAILABLE FOR DIFFERENT LANGUAGES MAKES OUR TAGGING APPROACH READILY APPLICABLE TO NON-ENGLISH LANGUAGES, WHICH IS NOT THE CASE FOR TRADITIONAL POS TAGGING METHODS.

Language	# articles
English	3,188,521
French	2,289,494
Malagasy	2,232,273
Chinese	828,580
Lithuanian	610,707
Russian	458,634
Greek	406,259
Korean	349,626
Swedish	329,137
Turkish	311,471

Table II. UNIFIED TAG SET USED BY OUR LIGHTWEIGHT TAGGER.

Tag	Examples
ADJECTIVE	green, valuable
ADVERB	strongly, quickly
CONJUNCTION	and, but, so
DETERMINER	the, an
NOUN	house, car
PROPER_NOUN	Stefanie, Linux
NUMBER	3.14, hundred
PARTICLE	who, whom
PRONOUN	his, theirs
PREPOSITION	in, from, on
PUNCTUATION	.;:!
VERB	see, fell, had
OTHER	anything else
UNKNOWN	special tag

The parsed information is then fed into an index that is easily queriable. From this index, we derive the necessary information to build the lightweight tagger. We only use the forms and part-of-speech tags of a word. The tagger itself consists of a simple hash map with the lower cased word forms/inflexions as keys and the corresponding lists of potential PoS tags as values. The tags are normalized to a set of 14 word categories, given in Table II. We map other tagging systems such as the Penn Treebank tag set or the Stuttgart tag set to these 14 word categories as well. For brevity, we omit these mappings here, they can be found online [13] (user: anonymous, no password).

The annotation process consists of the following steps:

1) Tokenize the input text, e.g. via OpenNLP [14] or JTokenizer [15]
2) Transform each token to lower case, taking its locale into account
3) For each token, look it up in the hash map
   a) If found, return a random PoS tag from the list of the found entry
   b) If not found
      i) If the token does not start with a letter, return UNKNOWN
      ii) If the token starts with an upper-case letter, return PROPER_NOUN

iii) Else, return NOUN

This simple dictionary and heuristics based approach is equal to the baseline systems used in the evaluation of many statistical PoS tagger models. The handling of unknown words is motivated by the assumption that Wiktionary covers non-noun word groups exhaustively, while nouns and proper nouns are underrepresented. We therefore assume that any word not in the dictionary is a proper-noun, in case it starts with an upper case letter, and a noun otherwise. Tokens that do not start with a letter are tagged with UNKNOWN, indicating that the tagger has no information about what category this token belongs to.

III. EVALUATION

Our evaluation strives to provide empirical evidence for the following hypothesis:

- The lightweight tagger is comparable in recall to state-of-the-art taggers.
- The lightweight tagger is sufficiently precise.
- The lightweight tagger is considerably faster than the state-of-the-art tagger.
- The errors introduced do not negatively influence supervised and unsupervised machine learning computations.

The following sections describe experiments carried out to gather evidence for the above assumptions.

A. PoS Tagging

We evaluated the lightweight tagger's precision and recall on the English Brown Corpus [16], [17] and compared it to results obtained from the Maximum Entropy based PoS tagger in the freely available OpenNLP package, which was trained on the Penn Treebank corpus [18]. The Brown Corpus consists of roughly one million tagged words, from various genres such as news articles, editorials, humorous texts and so on. For the evaluation we mapped both the Brown Corpus tag set and the Penn Treebank tags emitted by the OpenNLP PoS tagger to our simplified tag set described in Section II. This allowed us to directly quantify the relative performance of our tagger relative to the OpenNLP PoS tagger.

We let both taggers tag the entire Brown Corpus and then calculated precision and recall based on the ground truth found in the corpus. We used the tokenization as given in the Brown Corpus instead of using a dedicated tokenizer. The results are shown in Figure 1, which shows the precision and recall of each tagger for specific word categories. Our focus is on nouns and proper nouns, as these are used to generate features for text classification and clustering. The lightweight tagger has similar recall for nouns compared to the OpenNLP tagger. The precision of the lightweight tagger for nouns is what one would expect for a baseline dictionary tagger. While not exactly stellar, it is still performing surprisingly well. For proper nouns, the OpenNLP tagger is clearly superior in both precision and recall. We attribute this to the fact that we only tag tokens that start with an upper case letter as proper nouns.

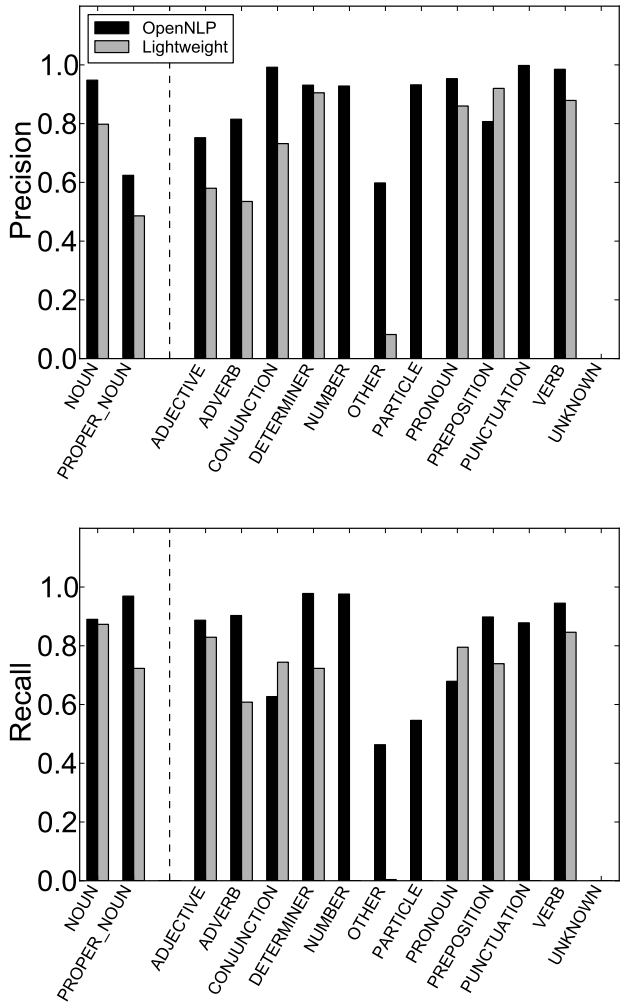

Figure 1. Comparison of precision (left) and recall (right) obtained by both the state-of-the-art tagger based on OpenNLP and our lightweight tagger on the tag set considered in this paper. For the relevant categories NOUN and PROPER_NOUN the lightweight tagger shows an acceptable precision and a recall comparable to the OpenNLP tagger.

We also measured the time spent on tagging. Our test machine was an Intel Core i7 CPU at 2.8Ghz, with 8GB RAM. Both the OpenNLP tagger and our lightweight tagger were written in Java. We ran the experiments in a 64-bit JVM from Oracle, version 1.7, update 9, assigning a maximum of 1GB of heap memory to the Java process. We ran each experiment 10 times and averaged the runtimes. The classification results are deterministic and where thus not averaged. Table III shows the average time taken by each annotator to annotate the entire Brown Corpus. As expected, the lightweight tagger was much faster than the much more sophisticated Maximum Entropy based tagger, which was outperformed by a factor of 100.

Another attribute of a dictionary and heuristics based tagger like the one presented here, is that it is more robust to grammatically incorrect text, found in social media or HTML pages cleaned and converted to plain text. The OpenNLP tagger, like other PoS taggers, needs sentence boundaries to function correctly. Our tagger has no such requirement and

Table III. TAGGING RUNTIME OF THE LIGHTWEIGHT TAGGER AND THE OPENNLP TAGGER FOR THE ENTIRE BROWN CORPUS AVERAGED ACROSS 10 RUNS. THE LIGHTWEIGHT TAGGER OUTPERFORMS THE MORE SOPHISTICATED MAXIMUM ENTROPY TAGGER BY A FACTOR OF 100.

Lightweight tagger	OpenNLP maximum entropy tagger
0.274s	34.219s

lends itself well to noisy text.

To establish whether the decrease in tagging performance has an impact on common natural text machine learning tasks, we evaluated both taggers in clustering and classification tasks as described in the next sections.

B. Feature Generation for Clustering & Classification

Our clustering and classification experiments share the same feature generation stage. Regardless of the corpus, we performed the following steps for each text document to transform it into a term vector. First, we tokenized each text document using the Maximum Entropy based tokenizer from the OpenNLP package. Each token was then stemmed using the Porter stemmer [19], [20] and normalized taking the English locale into consideration. We also applied a simple stop-word list found in Gate [21], filtering out high frequency words. Note that this step could actually be omitted since we are only taking nouns and proper nouns into account in the later stage.

For the OpenNLP tagger we also had to detect sentence boundaries with the corresponding Maximum Entropy sentence splitter in OpenNLP, which added additional processing time to the feature generation stage. Finally, we either tagged the tokens using the OpenNLP tagger or our lightweight tagger.

From the resulting list of tokens, we only took the stemmed and normalized form of nouns and proper nouns and converted them to term vectors as described in Section I. We then applied common TF-IDF weighting and normalized the vectors to unit length. In the TF-IDF scheme, the weight of term t in document d is given by

$$w_d(t) = \left(1 + \sqrt{n(t,d)}\right) \cdot \ln\left(1 + \frac{|D|}{n(t)}\right), \qquad (1)$$

where $n(t,d)$ is the number of times term t occurs in document d, $n(t)$ is the number of times the term t occurs in all documents, and $|D|$ is the total number of documents. These term vectors were then used as the input to our clustering and classification experiments.

C. Text Clustering

We evaluated our lightweight tagger in a text clustering scenario by comparing the achieved performance values to those obtained when using the OpenNLP tagger in the feature generation stage. We chose the 20 newsgroups corpus [22], [23] for our experiments as it is commonly used for text clustering evaluations. The 20 newsgroups corpus consists of roughly 20,000 documents, mostly equally collected from 20 different news groups, spanning topics such as atheism, politics, and sports. Each group is represented by around 1,000 documents. The corpus is split into a training and testing set for classification tasks, which we merged for this clustering scenario. We ran the experiment on both a manually selected subset of 5 groups ("rec.motorcycles", "alt.atheism", "talk.politics.guns", "comp.windows.x", "sci.crypt") as well as on the complete subset of all 20 newsgroups.

Table IV. PURITY AND CONDITIONAL CLUSTER ENTROPY AS WELL AS THE RUNTIMES OF FEATURE GENERATION (FE) AND CLUSTERING (C) ON THE SELECTED SUBSET OF 5 NEWSGROUPS. GOOD CLUSTERINGS HAVE HIGH PURITY AND LOW ENTROPY.

Configuration	Purity	Entropy	FE Runtime	C Runtime
All Tokens	0.3461	0.8928	13.678s	1.059s
NN/PN OpenNLP	0.8864	0.2897	29.090s	0.368s
NN/PN Lightweight	0.8915	0.2765	1.978s	0.378s

For clustering, we implemented the efficient online spherical k-means algorithm [24], using a constant learning rate of 0.01. We set the number of desired clusters to 5 and 20, respectively, and terminated the clustering algorithm after 3 iterations over the document set. For centroid initialization we used the k-means++ seeding strategy [25], which stochastically selects the initial centroids based on their distance to each other. In order to guarantee that all runs have the same initial conditions we provided a constant seed to the random number generator used. This enables a more effective comparison of different PoS taggers.

For each experiment we measured the purity and conditional cluster entropy of the resulting clusterings, as well as the runtimes of feature generation and clustering, averaged over 10 runs on our test machine described in Section III-A. Purity and conditional cluster entropy are both criteria for evaluating the clustering quality against a ground truth labeling [26]. Purity ranges between 0 and 1 and measures the accuracy of the assignment that is obtained if every instance would be labeled with the majority label within its corresponding cluster. A perfect purity of 1 is thus reached if all instances within each cluster have the same label. On the other hand, conditional cluster entropy measures the average amount of uncertainty about the cluster assignment of one instance that remains if its label is known. That is, a perfect clustering has a conditional cluster entropy of 0, since knowing the label completely determines the assigned cluster.

We performed the experiment on a total of 6 configurations, 3 runs for the selected subset of 5 newsgroups and another 3 runs on the full dataset of 20 newsgroups. The 3 configurations defined how the feature generation stage was carried out:

1) all tokens, stemmed, normalized,
2) nouns and proper nouns, stemmed, normalized, using the OpenNLP tagger,
3) nouns and proper nouns, stemmed, normalized, using the lightweight tagger.

Table IV summarizes the results on the 5 newsgroup subset, Table V describes the results on the full 20 newsgroup corpus.

The runtime of the feature generation stage was considerably lower for the lightweight tagger compared to the OpenNLP tagger. This was expected, though the relative speed-up is here 10 to 20-fold. This is a result of timing

Table V. PURITY AND CONDITIONAL CLUSTER ENTROPY AS WELL AS THE RUNTIMES OF FEATURE GENERATION (FE) AND CLUSTERING (C) ON THE FULL CORPUS OF ALL 20 NEWSGROUPS. GOOD CLUSTERINGS HAVE HIGH PURITY AND LOW ENTROPY.

Configuration	Purity	Entropy	FE Runtime	C Runtime
All Tokens	0.2659	0.7537	79.692s	12.833s
NN/PN OpenNLP	0.4432	0.5532	143.706s	4.145s
NN/PN Lightweight	0.4486	0.5541	7.300s	4.202s

Table VI. TOTAL NUMBER OF FEATURES GENERATED BY EACH CONFIGURATION, FOR BOTH THE SELECTED SUBSET 5 NEWSGROUPS AND THE FULL CORPUS

Configuration	# Features 5 NG	# Features 20 NG
All Tokens	99220	343408
NN/PN OpenNLP	28432	83388
NN/PN Lightweight	20842	63853

Table VII. COMPARISON OF THE NUMBER OF FEATURES OBTAINED WITH DIFFERENT FEATURE ENGINEERING METHODS ON BOTH THE TOTAL REUTERS CORPUS AND THE TRAINING SET ONLY. BOTH OUR LIGHTWEIGHT TAGGER AND THE STATE-OF-THE-ART OPENNLP TAGGER ONLY EXTRACT NOUNS AND PROPER NOUNS.

	training set	total
# documents	23,149	806,791
# features (all tokens)	49,427	303,732
# features (OpenNLP)	44,840	281,170
# features (lightweight)	42,049	255,147

Table VIII. COMPARISON OF THE RUNTIMES OF VECTORIZATION AND CLASSIFICATION OF THE REUTERS CORPUS IN DIFFERENT FEATURE ENGINEERING SCENARIOS.

	vectorization	classification
all tokens	382s	1810s
OpenNLP	1171s	1172s
lightweight	395s	1100s

the entire feature engineering stage instead of just the PoS tagging stage as described in Section III-A. Clustering times were comparable in case of the OpenNLP and lightweight tagger configurations, and higher in case of using all tokens. This can be explained by the fact that the clustering time is dominated by adjusting cluster centroids, which is proportional to the number of features in a cluster centroid. Using all tokens significantly increased the number of features of cluster centroids, as shown in Table VI.

As far as the quality of the resulting clusterings is concerned, using all tokens was highly detrimental to the clustering quality as shown in Tables IV and V. The additional tokens generated much noise, and the resulting cluster centroids were capturing features from all newsgroups. The other configurations for which only nouns and proper nouns were used performed considerably better. Both performed at approximately the same level on both the subset and the full corpus, supporting our hypothesis that using a sufficiently accurate PoS tagger like our lightweight tagger does not decrease clustering performance.

D. Text Classification

We evaluated our lightweight tagger also in a text classification scenario, by comparing the achieved performance values to those obtained with standard feature engineering methods. For this comparison we chose a classification task on the Reuters RCV1 corpus [27], a well-known dataset for document classification. It consists of 806,791 newswire stories that were collected over the period of one year, manually categorized, and made available by Reuters Ltd for research purposes. The dataset was labeled with respect to three different sets of categories: *Topics*, *Industries*, and *Regions*; in this paper we focus on the 103 *Topic* categories. Note that in contrast to the results described in [27] we use the raw RCV1 corpus without any additional corrections, thus the obtained performance values should not be directly compared.

The documents were transformed into term vectors as described in Section III-B. According to [27], the documents were then split into a training set of 23,149 documents, and a test set consisting of 781,278 instances. This is a chronological split that selects all documents published within the first 12 days in the corpus as training documents, while retaining most of the complete year as test data. This asymmetric split resulted in the interesting fact that only a relatively small subset of all terms in the corpus occurred in the training set (Table VII).

Classification was then performed by training a linear SVM on the training set for each of the 103 available topics, i.e., for each topic we solved a binary classification problem. We used the SVM implementation from LIBLINEAR [28] with default parameters $C = 4$ and $eps = 0.1$ (solver `L2R_L2LOSS_SVC_DUAL`). The individual performance values obtained for each topic on the test set were then combined using micro- and macro-averages and compared for the same feature engineering scenarios described in Section III-C: i) using all available tokens, ii) using nouns and proper nouns tagged by the state-of-the-art maximum entropy based tagger provided by the OpenNLP library, and iii) using nouns and proper nouns tagged by our lightweight tagger based on Wiktionary.

Table VII shows the number of features extracted for each of these scenarios. Interestingly, about 90% to 95% of all tokens are nouns, indicating the rather factual nature of newswire articles. This further supports the hypothesis that in many real world scenarios, the restriction to nouns in feature engineering preserves most of the information contained in documents.

The runtimes of the Reuters classification algorithm in all three feature engineering scenarios are shown in Table VIII. All times were averaged over 10 runs on our test machine specified in Section III-A. It can be seen that in the vectorization stage the lightweight tagger significantly outperforms the heavy-weight OpenNLP tagger and is almost as fast as using no tagger at all (all tokens). The runtime of the SVM classification is only indirectly depending on the type of feature engineering used; since the other preprocessing steps (tokenization, stemming, stop word removal) and the number of data samples are the same for all three scenarios, its runtime mainly depends on the number of features, which is highest for the case where all tokens are used.

 ISBN: 978-1-61208-295-0

Table IX. COMPARISON OF THE CLASSIFICATION PERFORMANCE OBTAINED WITH DIFFERENT FEATURE ENGINEERING METHODS. SHOWN ARE MACRO- AND MICRO-AVERAGES OF PRECISION, RECALL, AND F1, RESPECTIVELY, FOR THE THREE DIFFERENT SCENARIOS.

	Macro-Averages			Micro-Averages		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
all tokens	0.446	0.692	0.519	0.721	0.852	0.781
OpenNLP	0.428	0.663	0.497	0.704	0.840	0.766
lightweight	0.418	0.657	0.491	0.691	0.835	0.756

Finally, Table IX shows the classification performance for the different feature engineering scenarios. Macro- and micro-averages are calculated over the individual performance values obtained by single binary classifiers on different topics. Micro-F1 values of about 0.75 to 0.8 are comparable to other studies on the RCV1 corpus [27], [29]. It can be seen that the performance is higher when all tokens are used compared to the cases where only nouns and proper nouns enter the feature space, which is expected since the higher the dimensionality of the input space, the more likely the linear classifier is able to find a good separating hyperplane. However, a comparison of only the noun-based preprocessing methods reveals that both the lightweight tagger and the OpenNLP tagger roughly achieve the same classification performance.

Thus, we can conclude that also for supervised classification tasks the significant reduction of runtime when using the lightweight tagger does not come at the cost of a decreased performance, even though the general restriction to nouns and proper nouns indirectly influences classification performance through the dimensionality of the feature space.

IV. APPLICATIONS & FUTURE WORK

Our lightweight tagger enables the processing of big data in tolerable amounts of time as compared to using more sophisticated PoS tagging models. As shown in Section III-A, our tagger has comparable recall for nouns and proper nouns compared to state-of-the-art PoS taggers. As outlined above, it lends itself well as a subsitute for more sophisticated PoS tagging models in various text machine learning tasks. However, the tagger as well as the parser and index for Wiktionary from which the tagger is built can be exploited for other tasks as well.

A recent hot topic in text mining literature is the extraction of facts from large (web) corpora [30]. Part-of-speech tagging plays a role in fact extraction, as it is often the basis for extracting patterns that could represent factual information. We envision a big data scenario, e.g., on the Common Crawl corpus [31], where our lightweight tagger can be used to extract a set of candidate patterns quickly. These candidate patterns can then be further refined by using a more accurate PoS tagger. Instead of having to tag the entire corpus accurately, we preselect a much more managable set of candidates to which we apply the costly, more precise PoS tagger. We plan on investigating this approach in future research.

The parser and index we devised for Wiktionary has other interesting application scenearios. Lemmatization [32] could greatly benefit from using Wiktionary as an information source. Wiktionary entries provide us with information on the lemma of words, together with all or most of their inflexions. This allows us to build a comprehensive dictionary that can be used to lemmatize known words. Developing a hybrid model of Wiktionary based lookup and statistical methods for unknown words is planned for future research.

V. CONCLUSION

Part-of-speech tagging is a crucial and time-consuming preprocessing step in many machine learning scenarios on natural language text. Our work tries to reduce the runtime of this step by approximating the performance of state-of-the-art PoS taggers, exploiting Wiktionary as the information source. We demonstrated that our approach can be used as a replacement for more precise but also more expensive PoS tagging models. The feature generation stage is considerably faster, while the quality of the machine learning results do not deteriorate. Furthermore, our approach is extendable to other languages without having to manually PoS tag large corpora of text. Instead, language collections from Wiktionary can be easily parsed and integrated. Our approach is especially well suited for big data scenarios, where short processing times directly translate into more experiments that can be carried out within a fixed time span. We described possible application scenarios and our planned future work, which includes fact extraction from large web corpora and lemmatization.

REFERENCES

[1] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975.

[2] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc., 1986.

[3] S. E. Robertson and S. Walker, "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '94. New York, NY, USA: Springer-Verlag New York, Inc., 1994, pp. 232–241.

[4] C. Fox, "A stop list for general text," *SIGIR Forum*, vol. 24, no. 1-2, pp. 19–21, Sep. 1989.

[5] E. Brill, "A simple rule-based part of speech tagger," 1992.

[6] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun, "A practical part-of-speech tagger," in *In Proceedings of the Third Conference on Applied Natural Language Processing*, 1992, pp. 133–140.

[7] A. Ratnaparkhi, "A maximum entropy model for Part-Of-speech tagging," in *Proceedings of the Empirical Methods in Natural Language Processing*, E. Brill and K. Church, Eds., 1996, pp. 133–142.

[8] P. V. S. Avinesh and G. Karthik, "Part-Of-speech tagging and chunking using conditional random fields and Transformation-Based learning," in *Proceedings of the IJCAI and the Workshop On Shallow Parsing for South Asian Languages (SPSAL)*, 2007, pp. 21–24.

[9] J. Gimenez and L. Marquez, "Svmtool: A general pos tagger generator based on support vector machines," in *In Proceedings of the 4th International Conference on Language Resources and Evaluation*, 2004, pp. 43–46.

[10] http://www.wiktionary.org/, [Online; accessed July 10, 2013].

[11] http://meta.wikimedia.org/wiki/Wiktionary, [Online; accessed July 10, 2013].

[12] http://en.wiktionary.org/wiki/Wiktionary:Criteria_for_inclusion, [Online; accessed July 10, 2013].

[13] https://www.knowminer.at/svn/opensource/components/ie/trunk/api/src/main/java/at/knowcenter/ie/postags/, [Online; accessed July 10, 2013].

[14] http://opennlp.apache.org/, [Online; accessed July 10, 2013].

 ISBN: 978-1-61208-295-0

[15] https://github.com/andyroberts/jTokenizer, [Online; accessed July 10, 2013].

[16] W. N. Francis and H. Kucera, "Brown corpus manual," Department of Linguistics, Brown University, Providence, Rhode Island, US, Tech. Rep., 1979.

[17] http://nltk.org/nltk_data/, [Online; accessed July 10, 2013].

[18] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: the penn treebank," *Comput. Linguist.*, vol. 19, no. 2, pp. 313–330, Jun. 1993.

[19] M. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems*, vol. 14, no. 3, pp. 130–137, 1980.

[20] M. F. Porter, "Snowball: A language for stemming algorithms," Published online, October 2001, accessed 11.03.2008, 15.00h.

[21] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters, *Text Processing with GATE (Version 6)*, 2011.

[22] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 331–339.

[23] http://qwone.com/~jason/20Newsgroups/, [Online; accessed July 10, 2013].

[24] S. Zhong, "Efficient online spherical k-means clustering," in *Proc. 2005 IEEE International Joint Conference on Neural Networks*, vol. 5, 2005, pp. 3180–3185.

[25] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[26] C. D. Manning, P. Raghavan, and H. Schuetze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[27] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A New Benchmark Collection for Text Categorization Research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, 2004.

[28] R.-e. Fan, K.-w. Chang, C.-j. Hsieh, X.-r. Wang, and C.-j. Lin, "LIBLINEAR: A Library for Large Linear Classification," *Journal of Machine Learning Research*, vol. 9, no. 2008, pp. 1871–1874, 2012.

[29] R.-e. Fan and C.-j. Lin, "A Study on Threshold Selection for Multi-label Classification," Tech. Rep., 2007.

[30] M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain, "Names and similarities on the web: fact extraction in the fast lane," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ser. ACL-44. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006, pp. 809–816.

[31] http://commoncrawl.org/, [Online; accessed July 10, 2013].

[32] T. Korenius, J. Laurikkala, K. Järvelin, and M. Juhola, "Stemming and lemmatization in the clustering of finnish text documents," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, ser. CIKM '04. New York, NY, USA: ACM, 2004, pp. 625–633.

 ISBN: 978-1-61208-295-0