Detecting Uncertainty in Biomedical Literature: A Simple Disambiguation Approach Using Sparse Random Indexing

Erik Velldal
Department of Informatics, University of Oslo, Norway
erikve@ifi.uio.no

Abstract

This paper presents a novel approach to the problem of hedge detection, which involves the identification of so-called hedge cues for labeling sentences as certain or uncertain. This is the classification problem for Task 1 of the CoNLL-2010 Shared Task, which focuses on hedging in biomedical literature. We here propose to view hedge detection as a simple disambiguation problem, restricted to words that have previously been observed as hedge cues. Applying an SVM classifier, the approach achieves the best published results so far for sentence-level uncertainty prediction on the Shared Task test data. We also show that the technique of random indexing can be successfully applied for compressing the dimensionality of the original feature space by several orders of magnitude, while at the same time yielding better classifier performance.

1 Introduction

The problem of hedge detection refers to the task of identifying uncertainty or speculation in text. Being the topic of several recent shared tasks and dedicated workshops,[1] this is a problem that is receiving increased interest within the fields of NLP and biomedical text mining. In terms of practical motivation, hedge detection is particularly useful in relation to information extraction tasks, where the ability to distinguish between factual and uncertain information can be of vital importance.

The topic of the Shared Task at the 2010 Conference for Natural Language Learning (CoNLL), is hedge detection for the domain of biomedical research literature (Farkas et al., 2010). The task is defined for two levels of analysis: While Task 1 is described as *learning to detect sentences containing uncertainty*, the object of Task 2 is *learning to resolve the in-sentence scope of hedge cues*. The focus of the present paper is only on Task 1.

A hedge cue is here taken to mean the words or phrases that signal the attitude of uncertainty or speculation.[2] Examples 1-4 in Figure 1, taken from the BioScope corpus (Vincze et al., 2008), illustrate how cue words are annotated in the Shared Task training data. Moreover, the training data also annotates an entire sentence as *uncertain* if it contains a hedge cue, and it is the prediction of this sentence labeling that is required for Task 1.

The approach presented in this paper extends on that of Velldal et al. (2010), where a maximum entropy (MaxEnt) classifier is applied to automatically detect cue words, subsequently labeling sentences as uncertain if they are found to contain a cue. Furthermore, in the system of Velldal et al. (2010), the resolution of the in-sentence scopes of identified cues, as required for Task 2, is determined by a set of manually crafted rules operating on dependency representations. For readers interested in more details on this set of rules used for solving Task 2, the reader is referred to Øvrelid et al. (2010b). The focus of the present paper, however, is to present a new and simplified approach to the classification problem relevant for solving Task 1, and also partially Task 2, viz. the identification of hedge cues.

2 Overview

In Section 5 we first develop a Support Vector Machine (SVM) token classifier for identifying cue

[1] Hedge detection played a central part of the shared tasks of both BioNLP 2009 and CoNLL 2010, as well as the NeSp-NLP 2010 workshop (Negation and Speculation in NLP).

[2] As noted by Farkas et al. (2010), most hedge cues typically fall in the following categories; auxiliaries (*may*, *might*, *could*, etc.), verbs of hedging or verbs with speculative content (*suggest*, *suspect*, *indicate*, *suppose*, *seem*, *appear*, etc.), adjectives or adverbs (*probable*, *likely*, *possible*, *unsure*, etc.), or conjunctions (*either. . . or*, etc.).

(1) {ROI ⟨appear⟩ to serve as messengers mediating {directly ⟨or⟩ indirectly} the release of the inhibitory subunit I kappa B from NF-kappa B}.

(2) {The specific role of the chromodomain is ⟨unknown⟩} but chromodomain swapping experiments in Drosophila {⟨suggest⟩ that they {⟨might⟩ be protein interaction modules}} [18].

(3) These data {⟨indicate that⟩ IL-10 and IL-4 inhibit cytokine production by different mechanisms}.

(4) Whereas a background set of promoter regions is easy to identify, it is {⟨not clear⟩ how to define a reasonable genomic sample of enhancers}.

Figure 1: Examples of hedged sentences in the BioScope corpus. Hedge cues are here shown using angle brackets, with braces corresponding to their annotated scopes.

words. For a given sentence, the classifier considers each word in turn, labeling it as a *cue* or a *non-cue*. We will refer to this mode of cue classification as performing *word-by-word classification* (WbW). Later, in Section 6, we go on to show how better results can be obtained by instead approaching the task as a *disambiguation problem*, restricting our attention to only those tokens whose base forms have previously been observed as hedge cues. Reformulating the problem in this way simplifies the classification task tremendously, reducing the number of examples that need to be considered, and thereby also trimming down the relevant feature space to a much more manageable size. At the same time, the resulting classifier achieves the best published results so far on the Shared Task data (to the best of our knowledge).

Additionally, in Section 7 we show how the very large input feature space can be further compressed using *random indexing*. This is essentially a dimension reduction technique based on sparse random projections, which we here apply for feature extraction. We show that training the classifier on the reduced feature space yields better performance than when using the original input space. The evaluation measures and feature templates are detailed in Sections 5.2 and 5.3, respectively. Note that, while preliminary results for all models are presented for the development data throughout the paper, the performance of all models is ultimately compared on the official Shared Task held-out data in Section 8. We start, however, by providing a brief overview of related work in Section 3, and then describe the relevant data sets and preprocessing steps in Section 4.

3 Related Work

The top-ranked system for Task 1 in the official CoNLL 2010 Shared Task evaluation, described in (Tang et al., 2010), approaches cue identification as a *sequence labeling problem*. Similarly to Morante and Daelemans (2009), Tang et al. (2010) set out to label tokens according to a BIO-scheme, i.e. indicating whether they are at the Beginning, Inside, or Outside of a hedge cue. Tang et al. (2010) train both a Conditional Random Field (CRF) sequence classifier and an SVM-based Hidden Markov Model (HMM), finally combining the predictions of both models in a second CRF.

In terms of the overall approach, i.e. viewing the problem as a sequence labeling task, Tang et al. (2010) are actually representative of the majority of the ST participants for Task 1 (Farkas et al., 2010), including the top three performers on the official held-out data. As noted by Farkas et al. (2010), the remaining systems approached the task either as a WbW *token classification problem*, or directly as a *sentence classification problem*. Examples of the former are the systems of Velldal et al. (2010) and Vlachos and Craven (2010), sharing the 4th rank position (out of 24 submitted systems) for Task 1.

In both the sequence labeling and token classification approaches, a sentence is labeled as uncertain if it contains a word labeled as a cue. In contrast, the sentence classification approaches instead tries to label sentences directly, typically using Bag-of-Words (BoW) features. In terms of the official Task 1 evaluation, the sentence classifiers tended to achieve a somewhat lower relative rank.

4 Data Sets and Preprocessing

The training data for the CoNLL 2010 Shared Task is taken from the BioScope corpus (Vincze et al., 2008) and consists of 14,541 sentences (or other root-level utterances) from biomedical abstracts and articles. Some basic descriptive statistics for the data sets are provided in Table 1. We see that roughly 18% of the sentences are annotated as uncertain. The BioScope corpus also provides anno-

	Data Set	Sentences	Hedged Sentences	Cues	Multi-Word Cues	Tokens	Cue Tokens
Training	**Abstracts**	11,871	2,101	2,659	364	309,634	3,056
	Articles	2,670	519	668	84	68,579	782
	Total	14,541	2,620	3,327	448	378,213	3,838
	Held-Out	5,003	790	1,033	87	138,276	1,148

Table 1: The Shared Task data sets. The top three rows lists the properties of the training data, separately detailing its two components—biomedical abstracts and full articles. The bottom row summarizes the official held-out test data (articles only). Token counts are based on the tokenizer described in Section 4.1.

tation for hedge cues as well as their scope. Out of a total of 378,213 tokens, 3,838 are annotated as being part of a hedge cue. As can be seen, the total number of cues is somewhat lower (3,327), due to the fact that some tokens are part of the same cue, so-called multi-word cues (448 in total), such as *indicate that* in Example 3.

For evaluation purposes, the task organizers provided newly annotated biomedical articles, comprising 5,003 additional utterances, of which 790 are annotated as hedged (see overview in Table 1). The data contains a total of 1,033 cues, of which 87 are multi-word cues spanning multiple tokens, comprising 1,148 cue tokens altogether.

4.1 Tokenization

The GENIA tagger (Tsuruoka et al., 2005) takes an important role in our preprocessing set-up, as it is specifically tuned for biomedical text. Nevertheless, its rules for tokenization appear to not always be optimally adapted for the BioScope corpus. (For examples, GENIA unconditionally introduces token boundaries for some punctuation marks that can also occur token-internally.) Our preprocessing pipeline therefore deploys a home-grown, cascaded finite-state tokenizer (adapted from the open-source English Resource Grammar; Flickinger (2000)), which aims to implement the tokenization decisions made in the Penn Treebank (Marcus et al., 1993)—much like GENIA, in principle—but properly treating certain corner cases found in the BioScope data.

4.2 PoS Tagging and Lemmatization

For part-of-speech (PoS) tagging and lemmatization, we combine GENIA and TnT (Brants, 2000), which operates on pre-tokenized inputs but in its default model is trained on financial news from the Penn Treebank. Our general goal here is to take advantage of the higher PoS accuracy provided by GENIA in the biomedical domain, while using our improved tokenization.

For the vast majority of tokens, we use GENIA PoS tags and base forms (i.e. lemmas). However, GENIA does not make a PoS distinction between proper and common nouns, as in the Penn Treebank, and hence we give precedence to TnT outputs for tokens tagged as nominal by both taggers.

5 Hedge Cue Classification

This section develops a binary cue classifier similar to that of Velldal et al. (2010), but using the framework of large-margin SVM classification (Vapnik, 1995) instead of MaxEnt. For a given sentence, the word-by-word classifier (referred to as C_{WbW}) considers each token in turn, labeling it as a *cue* or *non-cue*. Any sentence found to contain a cue is subsequently labeled as *uncertain*.

5.1 Defining the Training Instances

As annotated in the training data, it is possible for a hedge cue to span multiple tokens, e.g. as in *whether or not*. The majority of the multi-word cues in the training data are very infrequent, however, most occurring only once, and the classifier itself is not sensitive to the notion of multi-word cues. A given word token is considered a cue as long as it falls within the span of a cue annotation.

As presented to the learner, a given token w_i is represented as a feature vector $f(w_i) = \vec{f_i} \in \Re^d$. Each dimension f_{ij} represents a feature function which can encode arbitrary properties of w_i. Section 5.3 describes the particular features we are using. Each training example can be thought of as a pair of a feature vector and a label, $\langle \vec{f_i}, y_i \rangle$. If w_i is a cue we have y_i=+1, while for non-cues the label is -1. For estimating the actual SVM classifier for predicting the labels on unseen examples we use the SVM^{light} toolkit (Joachims, 1999).

5.2 Evaluation Measures

We will be reporting precision, recall and F_1 for two different levels of evaluation; the sentence-level and the token-level. While the token-level scores indicate how well the classifiers succeed in identifying individual cue words, the sentence-level scores are what actually correspond to Task 1, i.e. correctly identifying sentences as being *certain* or *uncertain*.

5.3 Feature Templates

In the Shared Task system description paper of Velldal et al. (2010), results are reported for MaxEnt cue classifiers using a wide variety of feature types of both surface-oriented and syntactic nature. For the latter, Velldal et al. (2010) defines a range of syntactic and dependency-based features extracted from parses produced by the MaltParser (Nivre et al., 2006; Øvrelid et al., 2010a) and the XLE (Crouch et al., 2008), recording information about dependency relations, subcategorization frames, etc. However, it turned out that the simpler lexical and surface-oriented features were sufficient for the identification of hedge cues.

Drawing on the observation above, the classifiers trained in this paper are only based on simple sequence-oriented n-gram features collected for PoS-tags, lemmas and surface forms. For all these types of features we record neighbors for up to 3 positions left/right of the focus word. For increased generality, all these n-gram features also include non-lexicalized variants, i.e. excluding the focus word itself.

5.4 Preliminary Results

Instantiating all feature templates described above for the BioScope training data, using the maximal span for all n-grams (n=4, i.e. including up to 3 neighbors), we end up with a total of more than 6,500,000 unique feature types. However, after testing different feature configurations, it turns out that the best performing model only uses a small subset of this feature pool. The configuration we will be using throughout this paper includes; n-grams over base forms ± 3 positions of the focus word; n-grams over surface forms up to $+2$ positions only; and PoS of the focus word. This results in a set of roughly 2,630,000 feature types. In addition to reporting classifier performance for this feature configuration, we also provide results for a baseline model using only *unigram* features over surface forms. The behavior of this classifier is similar to what we would expect from simply compiling a list of cue words from the training data, based on the majority usage of each word as cue or non-cue.

As shown in Table 2, after averaging results from 10-fold cross-validation on the training data, the baseline cue classifier (shown as C_{WbW}^{Uni}) achieves a sentence-level F_1 of 88.69 and a token-level F_1 of 79.59. In comparison, the classifier using all the available n-gram features (C_{WbW}) achieves F-scores of 91.19 and 87.80 on the sentence-level and token-level, respectively. We see that the improvement in performance compared to the baseline is most pronounced on the token-level, but the differences in scores for both levels are found to be statistically significant at $p < 0.005$ using a two-tailed sign-test.

6 Reformulating the Classification Problem

An error analysis of our initial WbW classifier revealed that it is not able to generalize to new hedge cues beyond those that have already been observed during training. Even after adding the non-lexicalized variants of all feature types (i.e. making features more general by not including the focus word itself), the classifier still fails to identify any unseen hedge cues whose base form did not occur as a cue in the training material. On the other hand, only very few of the test cues are actually unseen ($\approx$1.5%), meaning that the set of cue words might reasonably be treated as a near-closed class (at least for the biomedical data considered in this study). As a consequence of these observations, we here reformulate the problem as follows. Instead of approaching the task as a classification problem defined for all words, we only consider words that have a base form observed as a hedge cue in the training material. In effect, any word whose base form has never been observed as a cue in the training data is automatically considered to be a non-cue when testing. Part of the rationale here is that, while it seems reasonable to assume that any word occurring as a cue can also occur as a non-cue, the converse is less likely.

While the training data contains a total of approximately 17,600 unique base forms (given the preprocessing outlined in Section 4), only 143 of these ever occur as hedge cues. By restricting the classifier to only this subset of words, we man-

Model	Sentence Level			Token Level		
	Prec	Rec	F1	Prec	Rec	F1
C^{Uni}_{WbW}	91.01	86.53	88.69	90.60	71.03	79.59
C_{WbW}	94.31	88.30	91.19	94.67	81.89	87.80
C_{Disamb}	93.64	89.68	91.60	94.01	83.55	88.45
C^{RI}_{Disamb}	93.78	88.45	91.03	94.05	81.97	87.58

Table 2: 10-fold cross-validation on the biomedical abstracts and articles in the training data.

age to simplify the classification problem tremendously, but without any loss in performance.

Note that, although we view the task as a disambiguation problem, it is not feasible to train separate classifiers for each individual base form. The frequency distribution of the cue words in the training material is very skewed with most cues being very rare—many occurring as a cue only once ($\approx$ 40%). (Note, that most of these words also have many additional occurrences in the training data as non-cues, however.) For the majority of the cue words then, it seems we can not hope to gather enough reliable information to train individual classifiers. Instead, we want to be able to draw on information from the more frequently occurring cues also when classifying or disambiguating the less frequent ones. Consequently, we still train a single global classifier as for the original WbW set-up. However, as the disambiguation classifier still only needs to consider a small subset of the number of words considered by the full WbW classifier, the number of instantiated feature types is, of course, greatly reduced.

For the full WbW classification, the number of training examples is 378,213. Using the feature configuration described in Section 5.4, this generates a total of roughly 2,630,000 feature types. For the disambiguation model, using the same feature configuration, the number of instantiated feature types is reduced to just below 670,000, as generated for 94,155 training examples.

Running the new disambiguation classifier by 10-fold cross validation on the training data, we find that it has substantially better recall than the original WbW classifier. The results are shown in the row C_{Disamb} in Table 2. Across all levels of evaluation the C_{Disamb} model achieves a boost in F_1 compared to C_{WbW}. However, when applying a two-tailed sign-test, considering differences in classifier decisions on both the sentence-level and token-level, only the latter differences are found to be significant (at $p < 0.005$).

7 Sparse Random Indexing

As mentioned in Section 5.1, each training example is represented by a d-dimensional feature vector $\vec{f_i} \in \Re^d$. Given n examples and d features, the feature vectors can be thought of as rows in a matrix $F \in \Re^{n \times d}$. One potential problem with using a vector-based numerical encoding of local context features such as those described in Section 5.3, is that the dimensionality of the feature space grows very rapidly with the number of training examples. Using local features, e.g. context windows recording properties such as direction and distance, the number of unique features grows much faster than when using, say, BoW features. In order to make the vector encoding scalable, we would like to somehow be able to put a bound on the number of dimensions.

As mentioned above, even after simplifying the classification problem, our input feature space is still rather huge, totaling roughly 670,000 feature types. Given that the number of training examples is only around $n \approx$ 95,000 we have that $d \gg n$, and whenever we want to add more feature templates or add more training data, this imbalance will only become more pronounced. It is also likely that many of the n-gram features in our model will not be relevant for the classification of new data points. The combination of many irrelevant features, and few training examples compared to the number of features, makes the learner prone to overfitting.

In previous attempts to reduce the feature space, we have applied several *feature selection* schemes, such as *filtering* on the correlation coefficient between a feature and a class label, or using simple frequency cutoffs. Although such methods are effective in reducing the number of features, they typically do so at the expense of classifier performance. Due to both data sparseness and the likelihood of many features being only locally relevant, it is difficult to reliably asses the relevance of the input features, and we risk filtering out many relevant features as well. Using simple filtering methods, we did not manage to considerably reduce the number of features without also significantly reducing the performance of the classifier. Although better results can be expected by using so-called *wrapper methods* (Guyon and Elisseeff, 2003) instead, this is not computationally feasible for large feature sets.

As an alternative to such feature selection methods, we here report on experiments with a technique known as *random indexing* (RI). This allows us to drastically compress the feature space without explicitly throwing out any features.

The technique of random indexing was initially introduced by Kanerva et al. (2000) for modeling the semantic similarity of words by their distribution in text.[3] Actually RI forms part of a larger family of dimension reduction techniques based on *random projections*. Such methods typically work by multiplying the feature matrix $F \in \Re^{n \times d}$ by a random matrix $R \in \Re^{d \times k}$, where $k \ll d$, thereby reducing the number of dimensions from d to k:

$$G = FR \in \Re^{n \times k}, \qquad \text{with } k \ll d \qquad (5)$$

Given that k is sufficiently high, the Johnson-Lindenstrauss lemma (Johnson and Lindenstrauss, 1984) tells us that the pairwise distances (and thereby separability) in F can be preserved with high probability within the lower-dimensional space G (Li et al., 2006). While the only condition on the entries of R is that they are i.i.d. with zero mean, they are typically also specified to have unit variance (Li et al., 2006).

One advantage of the particular random indexing approach is that the full $n \times d$ feature matrix F does not need to be explicitly computed. The method constructs the representation of the data in G by *incrementally accumulating* so-called *index vectors* assigned to each of the d features (Sahlgren and Karlgren, 2005). The process can be described by the following two simple steps:

- When a new feature is instantiated, it is assigned a randomly generated vector of a fixed dimensionality k, consisting of a small number of -1s and $+1$s (the remaining elements being 0). This is then the so-called *index vector* of the feature. (The index of the ith feature corresponds to the ith column of R.)

- The vector representing a given training example (the jth row of G represents the jth example) is then constructed by simply summing the random index vectors of its features.

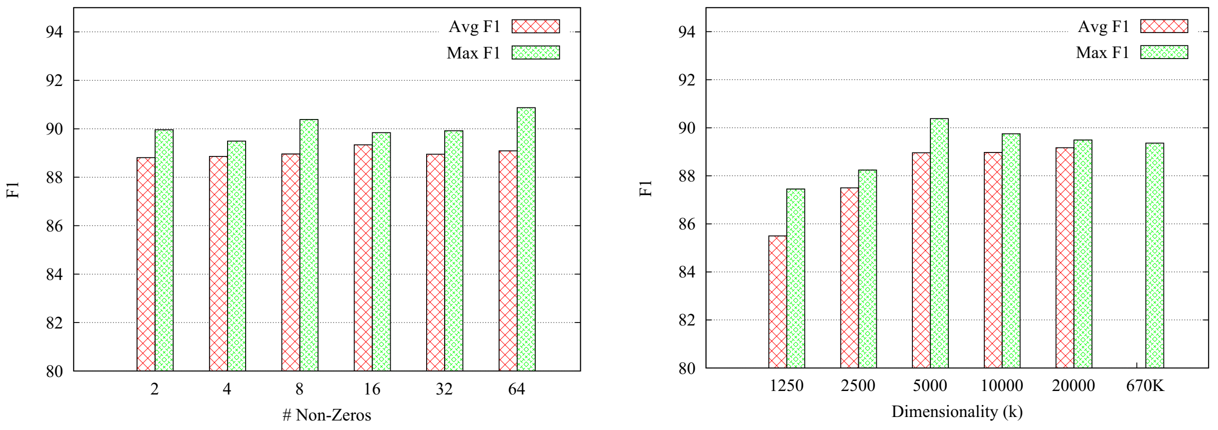
Note that, although we want to have $k \ll d$, we still operate in relatively high-dimensional space (with k being on the order of thousands). As noted by Sahlgren (2005), high-dimensional vectors having random directions are very likely to be close to orthogonal, and the approximation to F will generally be better the higher we set k.

Finally, it is worth noting that RI has traditionally been applied on the *type level*, with the purpose of accumulating context vectors that represent the distributional profiles of words in a semantic space model (Sahlgren, 2005). Here, on the other hand, we apply it on the *instance level* and as a general means of compressing the feature space of a learning problem.

7.1 Tuning the Random Indexing

Regarding the *ratio of non-zero elements*, the literature on random projections contains a wide range of suggestions as to how the entries of the random matrix R should be initialized. In the context of random indexing, Sahlgren and Karlgren (2005) set approximately 1% of the entries in each index to $+1$ or -1. It is worth bearing in mind, however, that the computational complexity of dot-product operations (as used extensively by the SVM learner) depend not only on the number of dimensions itself, but on the number of non-zero elements. We therefore want to take care to avoid ending up with a reduced space that is much more dense.Nevertheless, the appeal of using a random projection technique is in our case more related to its potential as a feature extraction step, and less to its potential for speeding up computations and reducing memory load, as the original feature vectors are already very sparse. After experimenting with different parametrizations, it seems that the classifier performance on our data sets are fairly stable with respect to varying the ratio of non-zeros. Moreover, we find that the non-zero entries can be very sparsely distributed, e.g. ≈ 0.05–0.2%, without much loss in classifier performance. Figure 2a shows the effect of varying the ratio of non-zero elements while keeping the dimensionality fixed (at k=5,000), always assigning an equal number of $+1$s and -1s (giving zero mean and unit variance). For each parametrization we perform a batch of 5 experiments using different random initializations of the index vectors. The scores shown in Figure 2a are the average and maximum within each batch. As can be seen, with index vectors of 5,000 elements, using 8 non-zero entries (corresponding to a ratio of 0.16%) here

[3] Readers are referred to Sahlgren (2005) for a good introduction to random indexing.



(a)

(b)

Figure 2: While varying various parameters of the random indexing, the plot shows averaged and maximum sentence-level F_1 from 5 different runs for each setting (using different random initializations of the index vectors), testing on $1/10$th of the training data. In (a) we vary the number of non-zero elements in the index vectors, while keeping the dimensionality fixed at k=5,000. In (b) we apply the disambiguation classifier using random indexing while varying the dimensionality k of the index vectors. The number of non-zeros varies from 2 (for k=1,250) to 32 (for k=20,000). For reference, the last column shows the result for using the original non-projected feature space.

seems to strike a reasonable balance between index density and performance.

As expected, we do, however, see a clear deterioration of classifier accuracy if the *dimensionality* of the index vectors is set very low. Figure 2b shows the effect of varying the dimensionality k of the index vectors, while fixing the ratio of non-zero entries per vector to 0.16%. Again we perform batches of 5 experiments for each value of k, reporting the average and maximum within each batch. For our cue classification data, the positive effect of increasing k seems to flatten out at around k=5,000. When considering the standard deviation of scores within each batch, however, the variability of the results seems to steadily decrease as k increases. For example, while we find σ=1.34 for the set of runs using k=1,250, we find σ=0.29 for k=20,000.

When looking at the *maximum scores* shown in Figure 2b, one of the runs using k=5,000 turns out to have the peak performance, achieving a (sentence-level) F_1 of 90.38. Not only does it score higher than any of the other RI-runs with $k{>}5{,}000$, it also outperforms the original C_{Disamb} model, which achieves an F_1 of 89.36 for the same single "fold" (the models in Figure 2b are tested using $1/10$th of the training material).

In our experience, although the random projection provided by the RI vectors only represents an approximation to the original input space, it still appears to preserve a lot more information than feature selection based on filtering methods.

7.2 Preliminary Results

The bottom row of Table 2 (C^{RI}_{Disamb}), shows the results of applying an SVM-classifier by full 10-fold cross-validation over the training set using the same random index assignments that yielded the maximum F_1 in Figure 2b for k=5,000 (with eight randomly set non-zeros in each index). We see that the performance of C^{RI}_{Disamb} is actually slightly lower than for C_{Disamb}. The differences are not detected as being significant though (applying the sign-test in the same manner as described above). Moreover, it should also be pointed out that we have not yet tried tuning the random indexing by multiple runs of full 10-fold cross-validation on the training data, which would be expected to improve these results. Given the fact that the effective feature space for the classifier is reduced from 670,000 to just 5,000 dimensions, we find it notable that the C^{RI}_{Disamb} model achieves comparable results, with only preliminary tuning.

Another important observation is that the *complexity* of the resulting SVM in terms of the number of support vectors (SVs), is considerably reduced for the RI-model: While the number of SVs for C_{Disamb} averages just below 8% of the training examples, this is reduced to just above 4% for C^{RI}_{Disamb} (using the SVM^{light} default settings). In

	Sentence Level			Token Level		
Model	Prec	Rec	F1	Prec	Rec	F1
C^{Uni}_{WbW}	77.54	81.27	79.36	75.89	66.90	71.11
C_{WbW}	89.02	84.18	86.53	87.58	74.30	80.40
C_{Disamb}	87.37	85.82	86.59	85.92	76.57	80.98
C^{RI}_{Disamb}	88.83	84.56	**86.64**	86.65	74.65	80.21
Tang	85.03	87.72	86.36	–	–	–

Table 3: Results on the Shared Task test data.

addition to halving the number of SVs, as well as reducing the feature space by two orders of magnitude, the upper bound on the VC-dimension (as estimated by SVM^{light}) is also reduced by 12%. It is also worth noting that the run-time differences for estimating the SVM on the original input space and the reduced (but slightly denser) feature space, are negligible ($\approx$ 5 CPU-seconds more for the RI-model when re-training on the full training set).

8 Held-Out Testing

Table 3 presents the final results for the various classifiers developed in this paper, testing them on the biomedical articles of the CoNLL 2010 Shared Task held-out test set (see Table 1). In addition to the evaluation results for our own classifiers, Table 3 also include the official test results for the system described by Tang et al. (2010). The sequence classifier developed by Tang et al. (2010), combining a CRF classifier and a large-margin HMM model, obtained the best results for the official ST evaluation for Task 1 (i.e. sentence-level uncertainty detection).

As seen from Table 3, all of our SVM classifiers C_{WbW}, C_{Disamb}, and C^{RI}_{Disamb}, achieve a higher sentence-level F_1 than the system of Tang et al. (2010) (though it is unknown whether the differences are statistically significant). We also note that our reformulation of the cue classification task as a disambiguation problem leads to better performance also on the held-out data, with C_{Disamb} performing slightly better than C_{WbW} across both evaluation levels. Interestingly, the best performer of them all proves to be the random indexing model (C^{RI}_{Disamb}), even though this model was not the top-performer on the training data. One possible explanation for the strong held-out performance of C^{RI}_{Disamb} is that the reduced complexity of this classifier (see Section 7.2) has made it less prone to overfitting, leading to better generalization performance on new data. Applying the sign-test as described above to the classifier decisions of C^{RI}_{Disamb}, we find statistically significant differences with respect to C_{WbW} but not with respect to C_{Disamb}. Nonetheless, the encouraging results of the C^{RI}_{Disamb} model on the held-out data means that further tuning of the RI configuration on the training data will be a priority for future experiments.

It is also worth noting that many of the systems participating in the ST challenge used fairly complex and resource-heavy feature types, being sensitive to document structure, grammatical relations, etc. (Farkas et al., 2010). The fact that comparable or better results can be obtained using a relatively simple approach as demonstrated in this paper—with low cost in terms of both computation and external resources—might lower the bar for employing a hedge detection component in an actual IE system.

Finally, we also observe that our simple unigram baseline classifier proves to be surprisingly competitive. In fact, comparing its Task 1 F_1 to those of the official ST evaluation, it actually outranks 7 of the 24 submitted systems.

9 Conclusions

This paper has presented the incremental development of uncertainty classifiers for detecting hedging in biomedical text—the topic of the CoNLL 2010 Shared Task. Using simple n-gram features over words, lemmas and PoS-tags, we first develop a (linear) SVM cue classifier that outperforms the top ranked system for Task 1 in the official Shared Task evaluation (i.e. sentence-level uncertainty detection). We then show how the original classification task can be greatly simplified by viewing it as a disambiguation task restricted to only those words that have previously been observed as hedge cues. Operating in a smaller (though still fairly large) feature space, this second classifier achieves even better results. Finally, we apply the method of random indexing, further reducing the dimensionality of the feature space by two orders of magnitude. This final classifier—combining an SVM-based disambiguation model with random indexing—is our best performer, achieving a sentence-level F_1 of 86.64 on the CoNLL 2010 Shared Task held-out data.

Acknowledgments

The experiments reported in this paper represent an extension of previous joint work with Stephan Oepen and Lilja Øvrelid (Velldal et al., 2010; Øvrelid et al., 2010b). The author also wishes to thank the anonymous reviewers, as well as colleagues at the Uni. of Oslo, for their valuable comments.

References

Thorsten Brants. 2000. TnT. A statistical Part-of-Speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 224 – 231, Seattle, WA.

Dick Crouch, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman. 2008. XLE documentation. Palo Alto Research Center.

Richard Farkas, Veronika Vincze, Gyorgy Mora, Janos Csirik, and Gyorgy Szarvas. 2010. The CoNLL 2010 Shared Task: Learning to detect hedges and their scope in natural language text. In *Proceedings of the 14th Conference on Natural Language Learning*, Uppsala, Sweden.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1):15 – 28.

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research; Special issue on variable and feature selection*, 3:1157 – 1182, March.

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

William Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189 – 206.

P. Kanerva, J. Kristoferson, and A Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, page 1036, Pennsylvania. Mahwah, New Jersey: Erlbaum.

Ping Li, Trevor Hastie, and Kenneth Church. 2006. Very sparse random projections. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06)*, Philadelphia.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English. The Penn Treebank. *Computational Linguistics*, 19:313 – 330.

Roser Morante and Walter Daelemans. 2009. Learning the scope of hedge cues in biomedical texts. In *Proceedings of the BioNLP 2009 Workshop*, pages 28 – 36, Boulder, Colorado.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 2216–2219.

Magnus Sahlgren and Jussi Karlgren. 2005. Automatic bilingual lexicon acquisition using random indexing of parallel corpora. *Journal of Natural Language Engineering, Special Issue on Parallel Texts*, 11(3), September.

Magnus Sahlgren. 2005. An introduction to random indexing. In *Proceedings of the Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering (TKE)*, Copenhagen, Denmark.

Buzhou Tang, Xiaolong Wang, Xuan Wang, Bo Yuan, and Shixi Fan. 2010. A cascade method for detecting hedges and their scope in natural language text. In *Proceedings of the 14th Conference on Natural Language Learning*, Uppsala, Sweden.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust Part-of-Speech tagger for biomedical text. In *Advances in Informatics*, pages 382 – 392. Springer, Berlin.

Vladimir Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

Erik Velldal, Lilja Øvrelid, and Stephan Oepen. 2010. Resolving speculation: MaxEnt cue classification and dependency-based scope rules. In *Proceedings of the 14th Conference on Natural Language Learning*, Uppsala, Sweden.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The BioScope corpus: Annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of the BioNLP 2008 Workshop*, Columbus, USA.

Andreas Vlachos and Mark Craven. 2010. Detecting speculative language using syntactic dependencies and logistic regression. In *Proceedings of the 14th Conference on Natural Language Learning*, Uppsala, Sweden.

Lilja Øvrelid, Jonas Kuhn, and Kathrin Spreyer. 2010a. Cross-framework parser stacking for data-driven dependency parsing. *TAL 2010 special issue on Machine Learning for NLP*, 50(3).

Lilja Øvrelid, Erik Velldal, and Stephan Oepen. 2010b. Syntactic scope resolution in uncertainty analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, Beijing, China.