Journal of Machine Learning Research 5 (2004) 605-621 Submitted 3/03; Revised 10/03; Published 6/04

New Techniques for Disambiguation in Natural Language and Their Application to Biological Text

Filip Ginter FILIP.GINTER@IT.UTU.FI
Jorma Boberg JORMA.BOBERG@IT.UTU.FI
Jouni Järvinen JOUNI.JARVINEN@IT.UTU.FI
Tapio Salakoski TAPIO.SALAKOSKI@IT.UTU.FI
Department of Information Technology, University of Turku, and
The Turku Centre for Computer Science (TUCS)
Lemminkäisenkatu 14A
20520 Turku, Finland

Editor: William W. Cohen

Abstract

We study the problems of disambiguation in natural language, focusing on the problem of gene vs. protein name disambiguation in biological text and also considering the problem of context-sensitive spelling error correction. We introduce a new family of classifiers based on ordering and weighting the feature vectors obtained from word counts and word co-occurrence in the text, and inspect several concrete classifiers from this family. We obtain the most accurate prediction when weighting by positions of the words in the context. On the gene/protein name disambiguation problem, this classifier outperforms both the Naive Bayes and SNoW baseline classifiers. We also study the effect of the smoothing techniques with the Naive Bayes classifier, the collocation features, and the context length on the classification accuracy and show that correct setting of the context length is important and also problem-dependent.

Keywords: biological text, gene vs. protein name disambiguation, textual data mining, word sense disambiguation, context-sensitive spelling error correction

1. Introduction

Disambiguation in natural language is a general problem of resolving the ambiguity present in natural language. The problems are, for example, word sense disambiguation, context-sensitive spelling error correction, the more special problem of gene/protein name disambiguation, and many other related problems.

Word sense disambiguation (WSD) is a long studied problem in the natural language processing community and it is important especially in the areas of information extraction and text understanding research. Given an ambiguous word in a text, the task of word sense disambiguation is to decide which of the several possible senses the word takes in this given instance. An often used example is the word “bank”. Bank can be a river bank, it can be a financial institution, or it can be the house in which the financial institution resides.

One common and closely related problem to WSD is context-sensitive spelling error correction such that the misspelled variant of the original word belongs to the language. Consider, for example, misspelling the word *dessert* as *desert*. Because *desert* belongs to the English lexicon, the traditional

lexicon-based spell-checkers will fail to discover the spelling error. A set of similar and correct English words, which are commonly misspelled, is called *confusion set*, for example, {"dessert", "desert"}. The task of context-sensitive spelling correction is to choose, for an instance of a word in text, its correct spelling from its confusion set. Thus, for example, whenever the word *dessert* is encountered in the text, the context-sensitive spell-checker decides, whether the correct spelling is *dessert* or *desert* in this case. It is easy to cast this problem as WSD: each word of the confusion set is considered as a "sense".

In this paper, we concentrate on another related disambiguation problem arising from the area of biological text, where very often a protein carries the same name as the gene which codes the protein. The application of WSD here is to decide whether the given gene/protein name in the given context refers to the sense "gene" or to the sense "protein". Hatzivassiloglou et al. (2001) give as an example the following two sentences: "By UV cross-linking and immunoprecipitation, we show that SBP2 specifically binds selenoprotein mRNAs both in vitro and in vivo." "The SBP2 clone used in this study generates a 3173 nt transcript (2541 nt of coding sequence plus a 632 nt 3′ UTR truncated at the polyadenylation site)." In the first sentence the occurrence of SBP2 is a protein, while the occurrence of SBP2 in the second sentence is a gene. Often the gene/protein name ambiguity is also an issue for human readers, as evidenced by the occasional inclusion of disambiguating information (for example "the SBP2 gene") by the authors of an article, and by the establishment of typographic conventions involving capitalization or italicizing by some journals. However, the authors do not follow these conventions faithfully, and therefore they cannot serve as a disambiguation technique for genes and proteins. Further, the italicizing is not preserved in the plain text format in which the PubMed abstracts are accessible.

Lexical disambiguation problems other than WSD can also be used as alternatives for the purpose of evaluating WSD systems. For example, Yarowsky (1994) uses the problem of restoring accents in Spanish and French texts as a substitute for the WSD problem. Similarly, we cast context-sensitive spelling error correction and gene/protein name disambiguation as alternatives to the WSD problem and use WSD terminology. We will refer to the work of Hatzivassiloglou et al. (2001), which offers a basic insight into the various existing methods used to solve the WSD problem for gene/protein name disambiguation. A more exhaustive review of existing methods is presented, for example, by Manning and Schütze (1999). Hatzivassiloglou et al. (2001) compare three existing state-of-the-art machine learning methods used for the gene/protein disambiguation problem: the C4.5 implementation of decision tree learning (Quinlan, 1993), the RIPPER implementation of inductive rule learning (Cohen, 1996), and the Naive Bayes classifier (e.g., Gale et al., 1992; Manning and Schütze, 1999). They experimentally show that the Naive Bayes classifier and the C4.5 classifier perform essentially with the same accuracy, whereas the RIPPER classifier gains accuracy of about 2% lower than the other two classifiers. However, the Naive Bayes classifier is considerably faster than the C4.5 classifier, both in training and in prediction. Therefore, they decided to use the Naive Bayes classifier in their subsequent experiments.

Here we propose a family of classifiers for the word sense disambiguation task. The classifiers are based on ordering and weighting of the feature vectors obtained from word counts and word co-occurrence in the text. We experimentally evaluate several classifiers from the family and compare their performance with the Naive Bayes classifier. As a second baseline method, we employ the SNoW learning architecture. Although SNoW is not a commonly used approach to the WSD problem, it is an efficient method shown to perform well on WSD by Golding and Roth (1999).

The performance of the proposed methods, Naive Bayes classifier with six smoothing techniques, and the SNoW classifier are studied carefully on the gene/protein name disambiguation task. We consider the effect of the context length and the use of collocations on the accuracy of these methods. The proposed method that incorporates the information about positions of words in the context of the word to be disambiguated into the decision function proves to be the best, demonstrating that the positional information improves the accuracy of the classification. In addition to the gene/protein name disambiguation problem, we test the performance of all the classifiers on the context-sensitive spelling error correction problem.

The paper is organized in the following way. In Section 2 we present the new classifiers for the WSD problem and in Section 3 we consider collocation features. Section 4 is devoted to a brief introduction to the baseline methods. In Section 5 we describe the experimental setting, the data used and the cross-validation method, and present the results for gene/protein name disambiguation and context-sensitive spelling error correction. We discuss the results and propose possible directions for the future research in Section 6.

2. The Proposed Method

Let w be the term whose sense we are disambiguating and let $s_1,\ldots,s_K$ be the K possible senses the term w can take. Let further $\overline{w}=(w_1,\ldots,w_n)$ be the context of the term w, where n is a positive even integer. The context words $w_1,\ldots,w_{\frac{n}{2}}$ are the words immediately preceding the term w in the text and the context words $w_{\frac{n}{2}+1},\ldots,w_n$ are the words immediately following the term w in the text. The context words appear in the vector $\overline{w}$ in the same order as in the text. The term w itself is not included to its context unless another instance of w belongs to the context $\overline{w}$. Let T be a training text, where the occurrences of the term w are labeled with their correct sense. We say that a *sense* s_k *has occurred* in the training text T, when a term w labeled with the sense s_k has occurred in T. Similarly, we say that a word w_i *belongs to a context of a sense* s_k, if it belongs to a context of a term w labeled with the sense s_k. Further, let $\mathrm{count}(w_i,s_k)$ be the number of occurrences of the word w_i in contexts of the sense s_k in the training text T (every occurrence of w_i is counted, also in case w_i appears more than once in the same context) and let $\mathrm{count}(s_k)$ be the number of occurrences of the sense s_k in the training corpus T.

For each context word w_i and sense s_k, we want to be able to measure to what extent the occurrence of the word w_i in the context of w suggests that the term w takes the sense s_k in this context. From the training text T we can count how many times the word w_i appeared in the context of the sense s_k. We call this number *evidence* and define it by

$$\mathrm{ev}(w_i,s_k)=\mathrm{count}(w_i,s_k)\ .$$

In order to estimate to what extent the word w_i is positively (or negatively) tied to the sense s_k, we define *expectation* which measures how many times the word w_i would be expected to appear in a context of the sense s_k, if w_i and s_k were independent, that is, if w_i was neither positively nor negatively tied to s_k. The expectation is defined by

$$\mathrm{ex}(w_i,s_k)=\frac{\mathrm{count}(w_i)}{|T|}\cdot\mathrm{total}(s_k)\ , \tag{1}$$

where $\mathrm{count}(w_i)$ is the number of times the word w_i occurred in the training text T, $|T|$ is the number of words in T, and $\mathrm{total}(s_k)$ is the number of the words in all the contexts of the sense s_k in T so that

in overlapping contexts the common words are counted only once. Multiplying total(s_k) with the relative frequency of w_i, we get how many of the occurrences of w_i would be expected to belong to a context of the sense s_k, if both w_i and s_k were evenly distributed in T and conditionally independent.

The more the evidence and the expectation differ, the more the word w_i is either positively or negatively tied to the sense s_k. We define a function

$$f(w_i,s_k) = \begin{cases} \frac{\text{ev}(w_i,s_k)-\text{ex}(w_i,s_k)}{\text{count}(w_i)\cdot\text{count}(s_k)} & \text{if count}(w_i) \neq 0, \\ 0 & \text{if count}(w_i) = 0, \end{cases}$$

which computes the *feature value* of the word w_i with respect to the sense s_k. The nominator measures the difference between the evidence (i.e., the really observed co-occurrence) and the expected co-occurrence. The denominator normalizes the value. The normalization step is important because the same value of $\text{ev}(w_i,s_k) - \text{ex}(w_i,s_k)$ has different significance for differently represented words and senses. For example, a difference of 10 for a word which has occurred 1000 times is less significant than a difference of 10 for a word which has occurred 100 times. In the former case the difference makes only 1% of the occurrences of the word, whereas in the latter case the difference makes 10% of the occurrences of the word. For similar reasons we also normalize by count(s_k) to account for different frequencies of senses. The normalization step assures that the feature values are of the same magnitude and comparable with each other.

A positive value of $f(w_i,s_k)$ corresponds the situation when the word w_i appears in the context of the sense s_k more often than would be by random and thus the word w_i is positively tied to the sense s_k. Similarly, a negative value of $f(w_i,s_k)$ means that w_i is negatively tied to s_k. If the value of $f(w_i,s_k)$ equals to zero, then w_i is in no way tied to s_k and brings no information as to what sense the given context belongs to.

The problem of zero-counts needs to be addressed since the value of count(w_i) becomes zero for words which do not appear in the training text T. It can be done by setting $f(w_i,s_k) = 0$ whenever count$(w_i) = 0$. Note that count(s_k) never becomes zero, as it is assumed that at least one training case for each sense exists. The reason why $f(w_i,s_k)$ is defined to be zero for unknown words is intuitive. The value zero means that there is no dependence between w_i and s_k, and thus the word w_i does not provide any information for the disambiguation.

In order to disambiguate between different possible senses of the term w, we consider its context $\bar{w} = (w_1,\ldots,w_n)$. Let us define a vector $f^k = (f_1^k,\ldots,f_n^k)$, where

$$f_i^k = f(w_i,s_k) \text{ for all } 1 \leq i \leq n.$$

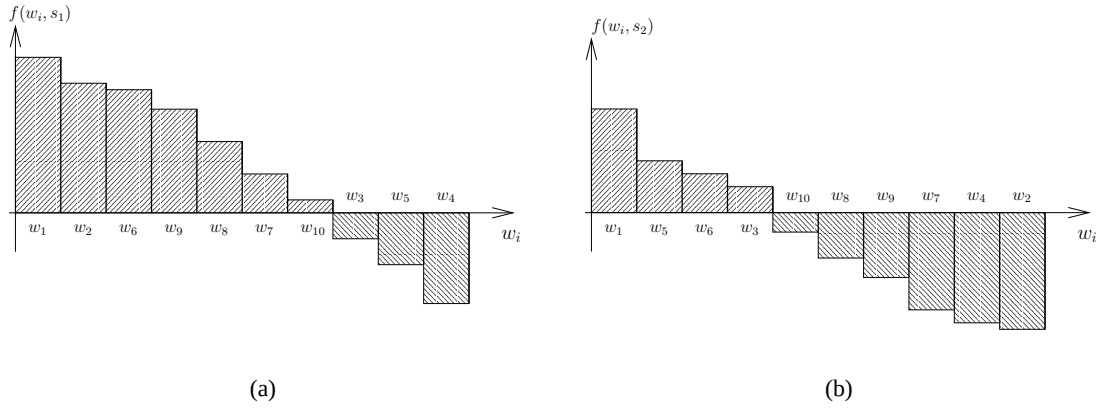
The vector f^k expresses how much the context words $w_1,\ldots,w_n$ are positively or negatively tied to the sense s_k. For each context $\bar{w}$, we construct the vectors $f^1,\ldots,f^K$, corresponding the senses $s_1,\ldots,s_K$. Let us call these vectors *feature vectors* of the context $\bar{w}$.

2.1 Unweighted Method: Sum of the Feature Values

In contrast with the Naive Bayes classifier (Section 4.1), which calculates a product of conditional probabilities, we define an additive decision function

$$s^\star = \arg\max_k \sum_{i=1}^{n} f_i^k \,. \tag{2}$$

In Figure 1 is depicted an example case for two senses ($K = 2$). For illustrative reasons, the components of the vectors f^1 and f^2 have been ordered in a descending order by value. Notice that even for $K = 2$, some words have positive feature values for more than one sense. For example, the word w_1 is positively tied to both senses s_1 and s_2. The figure also provides a graphical explanation for the decision function in Equation 2. It is easy to see that the decision function compares the total areas delimited by the feature values for each sense (where total area is the area delimited by positive feature values minus the area delimited by negative feature values). In the case of Figure 1 the sense s_1 is chosen.



(a) (b)

Figure 1: Example of feature vectors for two senses ($K = 2$). The components of the feature vectors have been ordered by value for visualization purposes.

2.2 Generalized Method: Weighting the Feature Values

In this section, we generalize the method described in Section 2.1. We introduce a weighting scheme and consider different orderings of the feature vectors. The decision function defined in Equation 2 performs unweighted summing of the components of the feature vectors, and thus the feature values of all context words w_i influence the final decision with equal strength. A straightforward generalization is to introduce a weighting scheme to allow different influences of the individual context words. The weighting scheme is carried out by ordering the feature vectors, and thereby the ordering is a tool to assign appropriate weights for the words in the context.

Let $M(\pi, v, n)$ be a classifier, where $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$ is a weight vector, n is an even context length, and $\pi : \mathbb{R}^n \to \mathbb{R}^n$ is a function that orders the components of $x \in \mathbb{R}^n$, that is, if $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, then $\pi(x) = (x_{j_1}, \ldots, x_{j_n})$ for some permutation $(j_1, \ldots, j_n)$ of $(1, \ldots, n)$. The classifiers $M(\pi, v, n)$ form a family, where each member of this family is distinguished by the ordering function π, the associated weight vector v, and the context length n.

A classifier $M(\pi, v, n)$ decides the sense of the word with K associated feature vectors $f^1, \ldots, f^K$ using the decision function

$$s^\star = \arg\max_k \sum_{i=1}^{n} v_i \cdot \pi(f^k)_i \,. \tag{3}$$

Hence, the classifier performs a weighted sum of the ordered feature vector using the associated weights and chooses such sense s_k for which the weighted sum is maximal.

In the following sections, we inspect more closely two subfamilies of classifiers distinguished by their associated ordering function π.

2.2.1 WEIGHTING BY FEATURE VALUES

Let us define a subfamily of $M(\pi, v, n)$ in which the ordering function π orders the vectors f^k in a descending order of feature values (recall Figure 1). If we further restrict the weights by the following conditions: $v_i \in [0,1]$, $1 \leq i \leq n$, and $\sum_{i=1}^{n} v_i = 1$, then the weighted sum $\sum_{i=1}^{n} v_i \cdot \pi(f^k)_i$ is an ordered weighted averaging (OWA) operator, well known in the theory of decision making (Yager, 1988). Furthermore, the Equation 3 can be interpreted as multicriteria decision making (Yager, 1997), where the aggregation function is an OWA operator and the individual criteria are the context words. The word sense disambiguation problem is thus cast as a multicriteria decision making process.

The weight vector v for this kind of ordering must be set with a knowledge of the given problem (Yager, 1997). However, in our extensive experiments we were not able to find such a feature-value-based weight vector v that the classifier introduced in Equation 3 would outperform the unweighted classifier introduced in Section 2.1. Instead, we found that better results are obtained when weighting the feature values by the positions of their corresponding words in the context.

2.2.2 WEIGHTING BY POSITION

Next we consider classifiers whose ordering function is the identity function $\pi(x) = x$, which preserves the contextual order in the vector f^k, that is, the order in which the words appear in the text. The meaning of the weight vector v here is to assess the relative importance of a context word w_i with respect to its distance from the term w (the term being disambiguated). Assuming that words closer to w are more significant for the decision, we can define the weight vector v, for example, as

$$v_i = \frac{1}{\mathrm{dist}(w_i)^{\alpha}} + \beta \text{ for all } 1 \leq i \leq n, \tag{4}$$

where $\alpha, \beta \in \mathbb{R}$, $\alpha > 0$, $\beta \geq 0$, and $\mathrm{dist}(w_i)$ is the distance between the positions of the words w and w_i, that is,

$$\mathrm{dist}(w_i) = \left\lceil \left| \frac{n+1}{2} - i \right| \right\rceil .$$

The values of the weights adopt a hyperbolic shape with highest values at the center of the vector v (see Figure 2). The parameter α determines how steeply the weight values grow towards the center of the vector, and the parameter β is an offset of the values. The role of the parameter β is to reduce the ratio between the weights of the words which are close to the term w and the weights of the words which are far from the term w.

3. Collocation Features

In order to introduce a local syntax information to the classifier, the *collocation* features are commonly used. The collocation features test for the presence of a pattern of up to l contiguous words around the target term w. We set $l = 2$ in our experiments, and thus in the context $\bar{w}$ of the term w we

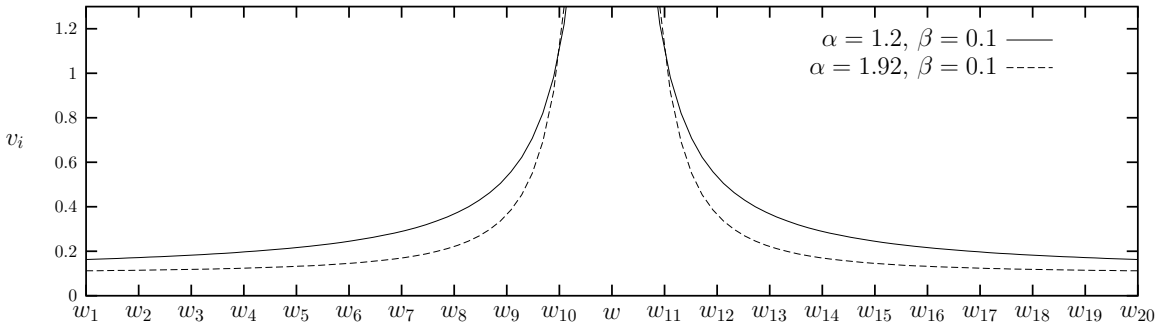


Figure 2: An example of the weight values defined in Equation 4 for different parameters α and β with $n = 20$.

have the five collocation features c_1 :="w_i _", c_2 :="_ w_i", c_3 :="w_i w_{i+1} _", c_4 :="_ w_i w_{i+1}", and c_5 :="w_i _ w_{i+1}", where the symbol _ matches the term w, regardless of its sense.

The collocations can be incorporated into the proposed classifiers by considering them as pseudowords. The evidence ev is defined as in Section 2. However, the expectation ex must be redefined. Since some collocation c_i can appear only in one place in the context of some sense s_k, and hence in count(s_k) places in T, we define

$$\mathrm{ex}(c_i, s_k) = \frac{\mathrm{count}(c_i)}{|T|} \cdot \mathrm{count}(s_k).$$

For the purpose of calculating the value of count(c_i), the symbol _ in the collocation matches any word in the text.

We will also introduce for each collocation c_i a pseudodistance dist(c_i), which will be used in Equation 4 to determine the weight for c_i. The pseudodistances we used in our experimental studies are described below, in Section 5.1.2.

4. Baseline Methods

In this section, we briefly introduce the Naive Bayes classifier and the SNoW classification architecture, the two baseline methods used in the experimental evaluation in Section 5.

4.1 The Naive Bayes Classifier

We introduce the Naive Bayes classifier as it is applied to the problem of word sense disambiguation. The specific formalization we describe is due to Gale et al. (1992) and Manning and Schütze (1999). The decision rule of the Naive Bayes classifier is

$$s^\star = \arg\max_k P(s_k|\overline{w}) = \arg\max_k (P(s_k) \cdot \prod_{i=1}^{n} P(w_i|s_k)), \tag{5}$$

where $P(s_k|\overline{w})$ is the conditional probability of the term w taking the sense s_k, given the context $\overline{w}$. The decision rule thus picks the sense s_k that is most probable for the context $\overline{w}$. The decision

rule assumes that the context words $w_1, \ldots, w_n$ are conditionally independent (the Naive Bayes assumption). The Maximum-Likelihood estimates

$$P(w_i|s_k) = \frac{\text{count}(w_i, s_k)}{\text{count}(s_k)} \qquad \text{and} \qquad P(s_k) = \frac{\text{count}(s_k)}{\sum_{j=1}^{K} \text{count}(s_j)} \tag{6}$$

are computed from the labeled training corpus T.

When disambiguating an occurrence of the term w, we face the problem of small values of $\text{count}(w_i, s_k)$, which make the Maximum-Likelihood estimate of $P(w_i|s_k)$ unreliable. In the extreme case when $\text{count}(w_i, s_k) = 0$, we get $P(w_i|s_k) = 0$ by Equation 6 and then $P(s_k|\bar{w}) = 0$ by Equation 5. To address this problem, various smoothing techniques have been derived, which redistribute some of the probability mass to the rare and unseen events. For the definitions and explanations of the smoothing techniques used in this paper, please refer to Chen and Goodman (1998) for Add-1, Kneser-Ney and Katz smoothing, Ng (1997) for Ng's smoothing, Kohavi et al. (1997) for No-matches-0.01 smoothing, and Golding and Roth (1999) for the Interpolative smoothing. Chen and Goodman (1998) provide a very valuable insight into various smoothing techniques and their performance on the language modeling problem.

4.2 The SNoW Classification Architecture

As a baseline alternative to the Naive Bayes classifier, we employ the SNoW[1] (Sparse Network of Winnows) classification architecture. Next we provide a brief introduction into the Winnow classifier. For a more detailed introduction refer to Golding and Roth (1999).

Let $\mathcal{F}$ be a space of features and let $\mathcal{F}_A \subseteq \mathcal{F}$ be a set of active features of an example. In our setting, $\mathcal{F}$ represents the set of all words in T and $\mathcal{F}_A$ represents the set of words present in the context $\bar{w}$. Further let $v_f \in \mathbb{R}_+$ be the weight of a feature $f \in \mathcal{F}$. The Winnow classifier then returns a positive classification 1 if

$$\sum_{f \in \mathcal{F}_A} v_f > \theta \,,$$

where $\theta \in \mathbb{R}$ is a suitable threshold, and a negative classification 0 otherwise.

The online mistake-driven training algorithm of the Winnow classifier is governed by three parameters: the promotion parameter $\alpha \in \mathbb{R}$, $\alpha > 1$, the demotion parameter $\beta \in \mathbb{R}$, $0 < \beta < 1$, and the default weight $v_{\text{def}} \in \mathbb{R}_+$. The weights v_f for all features $f \in \mathcal{F}$ are initialized to $v_f = 0$. When the classifier is presented an example, its prediction is computed. In case the prediction was correct, no changes are made to the classifier. In case the prediction was incorrect, the weights of all features $f \in \mathcal{F}_A$ are updated by

$$v_f \leftarrow \begin{cases} \alpha \cdot v_f & \text{if the example belongs to the positive class,} \\ \beta \cdot v_f & \text{if the example belongs to the negative class.} \end{cases}$$

In the former case (the misclassified example belongs to the positive class), all weights v_f such that $v_f = 0$ are set to $v_f = v_{\text{def}}$ before updating. Note that the weights of features which only occurred in negative examples always remain zero.

In the SNoW architecture several classifiers representing the same positive class are grouped into one *cloud*. The same examples are presented to each classifier in the cloud both in training and

1. http://l2r.cs.uiuc.edu/~cogcomp/

in prediction; the classifiers differ only by their training algorithm parameters. The decision of the classifiers within one cloud is combined as a weighted majority, where the weights depend on the performance of the individual classifiers during the training. The final prediction of SNoW is the class whose cloud had the highest weighted majority prediction. For details, see Golding and Roth (1999) or Littlestone and Warmuth (1994).

5. Evaluation of the Methods

We evaluate the unweighted classifier and the positionally weighted classifier. As baseline methods, we evaluate the Naive Bayes classifier with various smoothing techniques and the SNoW classification architecture. We evaluate the performance of the classifiers on the gene/protein name disambiguation problem and on the context-sensitive spelling error correction problem.

5.1 The Gene/Protein Name Disambiguation Task

Let us first consider the experimental setup and results for the gene/protein name disambiguation task.

5.1.1 DATA AND ITS PREPROCESSING

A common issue when using statistical methods is to obtain a large enough training set, which allows the calculation of the word frequencies on a sufficiently representative corpus of text. For the gene/protein disambiguation task, it would demand an enormous amount of expert work to obtain a large enough manually annotated corpus of text. Hatzivassiloglou et al. (2001) propose a simple approach to obtain the necessary annotated corpus in a fully automatic way. The automatic annotation method is based on the fact that sometimes the author disambiguates the term by explicitly following it with the word "gene" or "protein". These instances are clearly disambiguated by the author and can be used as training cases. The training text T is thus formed by a text where the instances readily disambiguated by the authors are tagged to be a gene or a protein, and the term "gene" or "protein" immediately following the disambiguated occurrences is removed. The document boundaries are preserved—the contexts $\bar{w}$ may not span between two different documents. The reported value of n, the context length, is thus a maximum value. Since the authors of the documents may explicitly disambiguate primarily the most difficult instances, the data may suffer of a certain bias. However, there is no practical way to obtain a large-enough set of certainly unbiased examples.

As the corpus we use 560093 documents which are article abstracts from years 1998–2002 downloaded from the PubMed database.[2] In order to identify the protein/gene names in the text, we use a list of names derived from 92849 records in the Swissprot database.[3] The actual number of searched terms is larger because many names have several synonyms. A small number (less than 1%) of gene/protein names are English words, which would lead to many false positives in gene/protein name identification. Thus, we remove from the list of gene/protein names all the words that occur in the English lexicon of the Brill's part-of-speech tagger.[4] In the corpus we have identified 65068 instances of gene/protein names, out of which 30768 were readily disambiguated

2. http://www.ncbi.nlm.nih.gov/PubMed/
3. http://www.expasy.org/sprot/
4. http://www.cs.jhu.edu/~brill/

as proteins and 34300 were readily disambiguated as genes. Any gene/protein name found in the text is replaced with an artificial word.

We further perform the following modifications of the text, as proposed by Hatzivassiloglou et al. (2001):

Case mapping All words are capitalized.

Removing stopwords Extremely common function words (for example: "is", "are", "the", "a") are removed from the text.

Stemming All words are stemmed using the Porter stemming algorithm (Porter, 1980).[5] Stemming maps related words to their stem so that, for example, "activate", "activating", "activated" all become one word "activ".

5.1.2 EXPERIMENTAL RESULTS

In order to estimate the parameters of the weighted classifier, we use random 200523 of the available documents and 5-fold cross-validation. For computational reasons, the parameter estimation is done in two phases. First we search for the parameters α, β, and n without collocation features. In this experiment, we perform an exhaustive grid search for the parameters trying all possible combinations of $\alpha \in [0, 3.5]$ with step 0.1, $\beta \in [0, 1]$ with step 0.1, and $n \in [2, 352]$ with step 10. We then select the best-performing combination of parameters $\alpha = 1.2$, $\beta = 0.1$, and $n = 252$. With these values of α, β, and n, we search for the distances of the collocations. Because collocations c_1 and c_2 are "symmetric", we fix $\mathrm{dist}(c_1) = \mathrm{dist}(c_2)$. Similarly, we set $\mathrm{dist}(c_3) = \mathrm{dist}(c_4)$. Thus, it suffices to search for $\mathrm{dist}(c_1)$, $\mathrm{dist}(c_3)$, and $\mathrm{dist}(c_5)$ only. We perform exhaustive grid search where $\mathrm{dist}(c_i) \in [0.05, 1]$, $i \in \{1, 3, 5\}$, with step 0.05. The best values found were $\mathrm{dist}(c_1) = 0.1$, $\mathrm{dist}(c_3) = 0.15$, and $\mathrm{dist}(c_5) = 0.15$.

Using the remaining 359570 documents, which have not been used for estimating the parameters, we then perform a series of 10-fold experiments for various settings of the context length n in order to study its effect on the accuracy.

The setting for the SNoW classifier is that of Golding and Roth (1999). For each sense, we define a cloud of five Winnows, differing the demotion parameter from $\beta = 0.5$ to $\beta = 0.9$. The promotion parameter $\alpha = 1.5$ and the default weight is set to 0.1.

Each classifier is evaluated for $n = 2, 4, \ldots, 160$ and the best accuracy value of each classifier and its corresponding context length is reported in Table 1. Note that the weighted classifier uses $n = 252$ obtained during the parameter estimation. The weighted classifier outperforms both the Naive Bayes classifier and the SNoW classifier, both with and without collocation features. Further, we see that the information about position of words in the context notably increases the accuracy of the weighted classifier when compared to the unweighted classifier that uses no positional information. The collocation features further substantially increase the accuracy of all the tested classifiers. It is interesting to note that the optimal value of n is generally bigger with collocation features.

The accuracy values for the various context lengths are presented in Figure 3. For the Naive Bayes classifier, we observe a steadily descending curve meaning that the Naive Bayes classifier performs best for short context lengths around $n = 10$ without collocations and $n = 24$ with collocations. On the contrary, both the weighted classifier and SNoW perform best for very long contexts.

5. http://www.tartarus.org/~martin/PorterStemmer/

Classifier		Without collocations		With collocations	
		n	Accuracy	*n*	Accuracy
New method	Weighted	252	82.37	252	86.12
	Unweighted	252	81.21	16	82.47
SNoW		98	77.87	102	84.54
Naive Bayes	Ng	10	78.54	24	84.44
	Add-1	14	78.79	24	84.40
	Kneser-Ney	10	78.42	24	83.97
	Interpolative	10	78.22	22	83.74
	No-matches-0.01	8	77.38	38	83.08
	Katz	28	77.87	30	82.67

Table 1: The maximum accuracy achieved by the compared methods together with the respective value of *n* for the gene/protein name disambiguation problem.

In order to test for statistical significance, we perform a 10-10-fold experiment (with collocations) for the weighted classifier, the Naive Bayes classifier with the Ng smoothing, and the SNoW classifier, in addition to the experiments described so far. In this experiment we repeat a 10-fold cross-validated experiment ten times, every time with a different split of the data. We then average the results of the ten 10-fold experiments and test for statistical significance using the paired Student's *t*-Test on the ten 10-fold results. The results are presented in Table 2. The test shows that all differences between the classifiers are significant ($p < 0.01$).

Classifier	*n*	Accuracy	Standard deviation
Weighted	252	86.12	0.05
Ng	24	84.29	0.06
SNoW	102	83.47	0.69

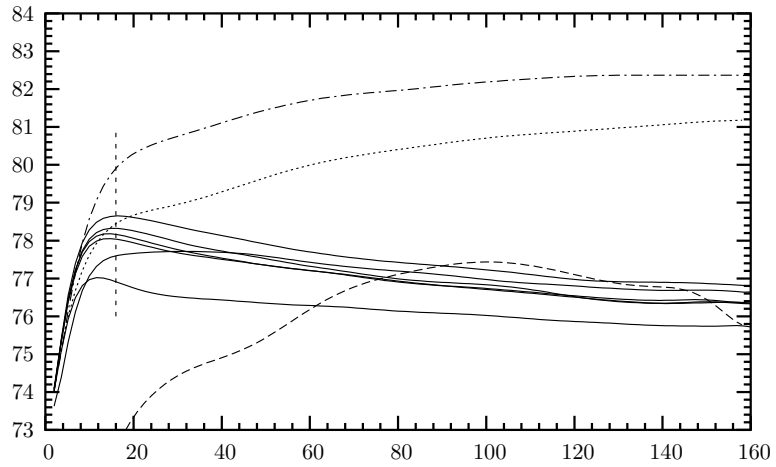
Table 2: Average results of the ten 10-fold experiments and standard deviation of the 10-fold experiment results for the gene/protein name disambiguation problem.

5.2 Context Sensitive Spelling Error Correction

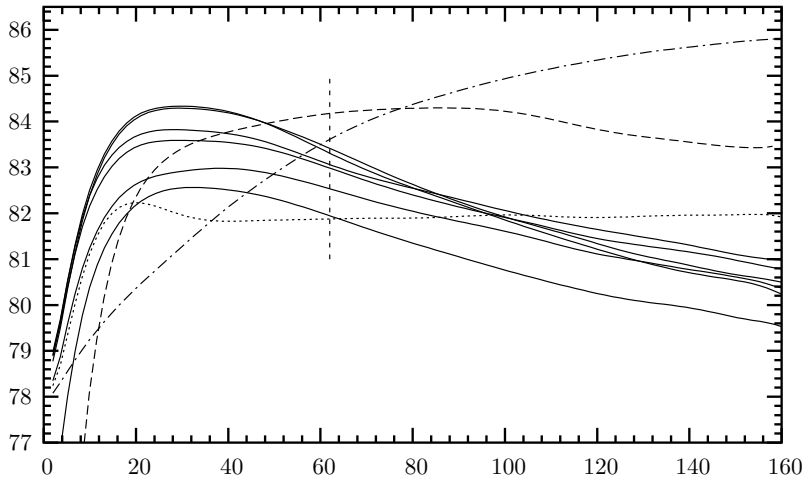
We perform further experiments on the context-sensitive spelling error correction problem. We evaluate the weighted classifier, the Naive Bayes classifier with various smoothing techniques, and the SNoW classification architecture. We only consider the case with collocations.

5.2.1 DATA AND ITS PREPROCESSING

We use first 473877 articles from the Reuters corpus (Rose et al., 2002). The confusion sets are those used in the experiments of Golding and Roth (1999). Once the words of the 21 confusion sets



(a) Without collocations. The full lines cross the dashed vertical marker line, from top to bottom in the following order: Add-1, Ng, Kneser-Ney, Interpolative, Katz, No-matches-0.01.



(b) With collocations. The full lines cross the dashed vertical marker line, from top to bottom in the following order: Add-1, Ng, Kneser-Ney, Interpolative, No-matches-0.01, Katz.

Figure 3: The relationship between context length and accuracy, measured on the complete gene/protein test data using 10-fold cross-validation. The curves are smoothed (Bezier curves). The full line represents Naive Bayes with various smoothing methods, the dotted line represents the unweighted classifier, the dash-dot line represents the weighted classifier, and the dashed line represents the SNoW classifier.

	α	β	dist(c_1)	dist(c_3)	dist(c_5)
Average	1.92	0.10	0.33	0.52	0.25
St. dev.	0.78	0.14	0.25	0.22	0.28
Min.	0.30	0.00	0.10	0.10	0.10
Max.	3.00	0.45	1.00	0.90	1.00

Table 3: Summary of the best performing parameter values found for the 21 confusion sets.

have been identified (case-insensitive except for the word "I") and labeled as training examples, we perform case mapping and stemming. The article boundaries are preserved.

5.2.2 EXPERIMENTAL RESULTS

After preliminary experiments we observed that the value $n = 252$ is too large for the weighted classifier on the context-sensitive spelling error correction problem. We thus set $n = 20$ for all the tested classifiers, which is also the value of *n* used by Golding and Roth (1999) when applying the SNoW and Naive Bayes classifiers to context-sensitive spelling error correction problem. In order to estimate the parameters, we use random 158105 of the available articles and 5-fold cross-validation. We then estimate the parameters α, β, dist(c_1), dist(c_3), and dist(c_5) separately for each of the 21 confusion sets. We use a similar two-phase protocol as for the gene/protein name disambiguation problem. First we perform a grid search of $\alpha \in [0,3]$ with step 0.1 and $\beta \in [0,0.5]$ with step 0.025. Then, using the best performing combination of α and β for each confusion set, we perform the grid search dist(c_i) $\in [0,1]$, $i \in \{1,3,5\}$, with step 0.1. A summary of the parameter values found is presented in Table 3. The weights induced by the average values of α and β are presented in Figure 2.

Using the remaining 315772 articles we then perform 10-fold cross-validated experiment for each of the 21 confusion sets. The results for the weighted classifier, SNoW, and Naive Bayes with the best smoothing are presented in Table 4. In Table 5 we present the average results for the various smoothing techniques of the Naive Bayes classifier. The *majority baseline* is the accuracy obtained by always selecting the most common member of each confusion set. We measure statistical significance of the average difference of the classifiers by the paired Student's *t*-Test on the 21 10-fold results of the confusion sets. The difference between the weighted classifier and the SNoW and Naive Bayes classifiers is statistically significant ($p \approx 0.01$ and $p \approx 0.03$). The difference between SNoW and Naive Bayes is not significant ($p \approx 0.77$).

6. Conclusions and Discussion

We propose for the problems of disambiguation in natural language a new family of classifiers characterized by additive decision function and weighting of word co-occurrence features. The proposed classifiers perform weighted combination of features, where the weights are assigned based on an ordering of the features. The concrete member of the proposed family on which we focus in this work assigns the weights of the features based on their distance from the word to be disambiguated.

Of the problems of disambiguation in natural language, we focus on gene/protein name disambiguation and also consider context-sensitive spelling error correction. For the gene/protein name

Confusion set	Num. of examples	Majority	SNoW	Naive Bayes	Weighted
accept, except	16828	71.76	98.26	94.75	98.38
affect, effect	14063	71.37	96.14	96.74	96.95
among, between	80731	74.46	94.29	94.77	94.54
amount, number	43408	63.19	93.02	90.56	90.83
begin, being	43243	79.86	98.39	98.44	97.45
cite, sight, site	6633	79.48	94.13	95.64	94.20
country, county	48838	80.16	98.36	97.97	97.77
fewer, less	18670	89.99	93.63	94.01	90.90
I, me	93579	93.60	99.41	99.40	99.23
its, it's	288962	91.01	98.91	99.03	97.00
lead, led	36655	56.41	95.90	95.64	95.05
maybe, may be	12269	82.03	95.36	95.54	91.45
passed, past	24589	79.27	97.68	97.90	97.17
peace, piece	22661	95.14	99.02	99.36	99.02
principal, principle	4720	53.05	92.12	93.23	93.15
quiet, quite	12946	53.77	96.60	96.87	96.73
raise, rise	56435	76.23	98.21	97.97	94.12
than, then	115760	80.26	98.37	97.84	97.97
their, there, they're	219097	54.46	98.85	98.50	95.96
weather, whether	29064	68.75	99.20	99.20	99.05
your, you're	5855	74.39	93.43	94.42	92.97
Average		74.70	96.63	96.56	95.71
Standard deviation		12.84	2.36	2.35	2.70

Table 4: Results for the 21 confusion sets.

Smoothing	Accuracy
No-matches-0.01	96.56
Ng	96.43
Kneser-Ney	96.10
Add-1	95.80
Katz	95.02
Interpolative	94.58

Table 5: Average accuracy on the 21 confusion sets for various smoothing techniques.

disambiguation problem, we perform a study of the effect of the context length *n* and show that each of the tested classifiers generally performs best for different context lengths on the gene/protein name disambiguation problem. While the Naive Bayes classifier performs best for short context lengths, SNoW and the proposed weighted classifier perform best for very long contexts. This is, however, problem-dependent, because shorter context length gave better results for all classifiers on the context-sensitive spelling error correction problem. We also evaluate the effect of collocation features which provide the classifiers with local syntax information. As expected, the collocation

features increase the accuracy of all the evaluated classifiers on both problems. The increase is bigger for the baseline methods than for the weighted classifier, which is consistent with understanding the positional weights as an alternative approach of introducing the local syntax to the classifier.

On our main task, that is, the gene/protein name disambiguation task, the proposed weighted classifier is shown to outperform the baselines, thus meeting our objective of improving the classification accuracy on this problem. On context-sensitive spelling error correction the baselines outperform the new method. A feature of context-sensitive spelling error correction to consider is the context length, where in context-sensitive spelling error correction, a short context length performs better. This might indicate that the proposed classifier performs better than the two baselines on problems which allow combining of features from a very long context.

Considering the per-confusion-set parameters presented in Table 3, we observe that the parameter α is relatively high in most of the cases, verifying the intuition that close features are more important for the disambiguation. This is also true for the gene/protein problem. Further, we find that the parameters dist(c_i) set the weight of collocation features higher than that of context words, verifying the intuition that collocation features are more important for the disambiguation. Further, the values of the parameters dist(c_i) suggest that, on average, the features c_3 and c_4 (that is, collocations of the types "$w_i\ w_{i+1}\ \_$" and "$\_\ w_i\ w_{i+1}$") are, somewhat surprisingly, least important among the collocation features. The variance of the parameters is relatively high, suggesting that their optimal values are data-dependent.

The new method is comparable to both the Naive Bayes and SNoW classifiers in its computational complexity, as it performs a simple word count statistics similar to that of the two baselines. However, the new method is more demanding in terms of space, since it stores a dictionary of words appearing in the whole text (due to the term count(w_i) in Equation 1) rather than words appearing only in the contexts $\overline{w}$.

An advantage of the new method is the simple way it deals with the zero-count problem. The proposed method permits zero feature values and does not require any special smoothing technique. For the Naive Bayes classifier, it is not always obvious which of the smoothing techniques should be used. This is demonstrated also in this study, where the relative accuracy of the various smoothing techniques differs between the two problems/corpora.

The new method exploits the information about the position of the words in the context, which has not been successfully accomplished with the Naive Bayes classifier for the same task by Hatzivassiloglou et al. (2001), who made the position a part of the feature and consequently the classifier apparently suffered from sparse data. In this paper, we show that the positional information can be incorporated in the form of weights and it substantially improves the accuracy of the classifier, as shown in the experiments.

Hatzivassiloglou et al. (2001) report classification accuracy for Naive Bayes on the gene/protein disambiguation task to be 84.48%. We have achieved a comparable accuracy of 84.44% for the Naive Bayes classifier with the collocation features. We are not sure whether Hatzivassiloglou et al. used the collocation features, what smoothing for Naive Bayes they used, and what was the context length in their experiments. Further, the two studies differ by the corpus used. Thus, it is impossible to compare the results directly.

We introduce the weighting scheme via ordering of the feature vectors. We perform our experiments on the context order, that is, the natural order of the words in the sentence. As a future work, we find interesting to study other possible orderings, that is, other possible models of relative importance of the individual features. For example, a word of biological relevance in the context may

be very important regardless its position in the text, and an ordering based on biological relevance of the context words could be considered.

Acknowledgments

We would like to acknowledge the people of the MediCel company, particularly Meelis Kolmer, Ph.D., who have kindly answered our numerous questions in the field of biology and advanced our understanding of the specific problems which biological texts bring to the natural language processing. This work uses the Reuters corpus volume 1 distributed by Reuters. This work has been supported by Tekes, the Finnish National Technology Agency.

References

Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Centre for Research in Computing Technology, Harvard University, Cambridge, Massachusetts, 1998.

William W. Cohen. Learning trees and rules with set-valued features. In William J. Clancey and Dan Weld, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 709–716. AAAI Press, Menlo Park, California, 1996.

William A. Gale, Kenneth W. Church, and David Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26:415–439, 1992.

Andrew R. Golding and Dan Roth. A Winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34:107–130, 1999.

Vasileios Hatzivassiloglou, Pablo A. Duboué, and Andrey Rzhetsky. Disambiguating proteins, genes, and RNA in text: A machine learning approach. *Bioinformatics*, 17:97–106, 2001.

Ron Kohavi, Barry Becker, and Dan Sommerfield. Improving Simple Bayes. In Maarten van Someren and Gerhard Widmer, editors, *Proceedings of the 9th European Conference on Machine Learning*, pages 78–87. Springer Verlag, Heidelberg, 1997.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999.

Hwee Tou Ng. Exemplar-based word sense disambiguation: Some recent improvements. In Claire Cardie and Ralph Weischedel, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 208–213. Association for Computational Linguistics, Somerset, New Jersey, 1997.

Martin F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

Ross J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, California, 1993.

Tony G. Rose, Mark Stevenson, and Miles Whitehead. The Reuters Corpus Volume 1: From yesterday's news to tomorrow's language resources. In Manuel Gonzales Rodriguez and Carmen Paz Suarez Araujo, editors, *Proceedings of the Third International Conference on Language Resources and Evaluation*. ELRA, Paris, 2002.

Ronald R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18:183–190, 1988.

Ronald R. Yager. On the inclusion of importances in OWA aggregations. In Ronald R. Yager and Janusz Kacprzyk, editors, *The Ordered Weighted Averaging Operators, Theory and Applications*, pages 41–59. Kluwer Academic Publishers, Norwell, Massachusetts, 1997.

David Yarowsky. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 88–95. Association for Computational Linguistics, Somerset, New Jersey, 1994.