On Reverse Feature Engineering of Syntactic Tree Kernels

Daniele Pighin
FBK-irst, DISI, University of Trento
Via di Sommarive, 14
I-38123 Povo (TN) Italy
daniele.pighin@gmail.com

Alessandro Moschitti
DISI, University of Trento
Via di Sommarive, 14
I-38123 Povo (TN) Italy
moschitti@disi.unitn.it

Abstract

In this paper, we provide a theoretical framework for feature selection in tree kernel spaces based on gradient-vector components of kernel-based machines. We show that a huge number of features can be discarded without a significant decrease in accuracy. Our selection algorithm is as accurate as and much more efficient than those proposed in previous work. Comparative experiments on three interesting and very diverse classification tasks, i.e. Question Classification, Relation Extraction and Semantic Role Labeling, support our theoretical findings and demonstrate the algorithm performance.

1 Introduction

Kernel functions are very effective at modeling diverse linguistic phenomena by implicitly representing data in high dimensional spaces, e.g. (Cumby and Roth, 2003; Culotta and Sorensen, 2004; Kudo et al., 2005; Moschitti et al., 2008). However, the implicit nature of the kernel space causes two major drawbacks: (1) high computational costs for learning and classification, and (2) the impossibility to identify the most important features. A solution to both problems is the application of feature selection techniques.

In particular, the problem of feature selection in Tree Kernel (TK) spaces has already been addressed by previous work in NLP, e.g. (Kudo and Matsumoto, 2003; Suzuki and Isozaki, 2005). However, these approaches lack a theoretical characterization of the problem that could support and justify the design of more effective algorithms.

In (Pighin and Moschitti, 2009a) and (Pighin and Moschitti, 2009b) (P&M), we presented a heuristic framework for feature selection in kernel spaces that selects features based on the components of the weight vector, $\vec{w}$, optimized by Support Vector Machines (SVMs). This method appears to be very effective, as the model accuracy does not significantly decrease even when a large number of features are filtered out. Unfortunately, we could not provide theoretical or intuitive motivations to justify our proposed apporach.

In this paper, we present and empirically validate a theory which aims at filling the above-mentioned gaps. In particular we provide: (i) a proof of the equation for the exact computation of feature weights induced by TK functions (Collins and Duffy, 2002); (ii) a theoretical characterization of feature selection based on $\|\vec{w}\|$. We show that if feature selection does not sensibly reduces $\|\vec{w}\|$, the margin associated with $\vec{w}$ does not sensibly decrease as well. Consequently, the theoretical upperbound to the probability error does not sensibly increases; (iii) a proof that the convolutive nature of TK allows for filtering out an exponential number of features with a small $\|\vec{w}\|$ decrease. The combination of (ii) with (iii) suggests that an extremely aggressive feature selection can be applied. We describe a greedy algorithm that exploits these results. Compared to the one proposed in P&M, the new version of the algorithm has only one parameter (instead of 3), it is more efficient and can be *more easily* connected with the amount of gradient norm that is lost after feature selection.

In the remainder: Section 2 briefly reviews SVMs and TK functions; Section 3 describes the problem of selecting and projecting features from very high onto lower dimensional spaces, and provides the theoretical foundation to our approach; Section 4 presents a selection of related work; Section 5 describes our approach to tree fragment selection; Section 6 details the outcome of our experiments; finally, in Section 7 we draw our conclusions.

2 Fragment Weights in TK Spaces

The critical step for feature selection in tree kernel spaces is the computation of the weights of features (*tree fragments*) in the kernel machines' gradient. The basic parameters are the fragment frequencies which are combined with a decay factor used to downscale the weight of large subtrees (Collins and Duffy, 2002). In this section, after introducing basic kernel concepts, we describe a theorem that establishes the correct weight[1] of features in the STK space.

2.1 Kernel Based-Machines

Typically, a kernel machine is a linear classifier whose decision function can be expressed as:

$$c(\vec{x}) = \vec{w} \cdot \vec{x} + b = \sum_{i=1}^{\ell} \alpha_i y_i \vec{x_i} \cdot \vec{x} + b \qquad (1)$$

where $\vec{x} \in \Re^N$ is a classifying example and $\vec{w} \in \Re^N$ and $b \in \Re$ are the separating hyperplane's *gradient* and its *bias*, respectively. The gradient is a linear combination of ℓ training points $\vec{x_i} \in \Re^N$ multiplied by their labels $y_i \in \{-1, +1\}$ and their weights $\alpha_i \in \Re^+$. Different optimizers use different strategies to learn the gradient. For instance, an SVM learns to maximize the distance between positive and negative examples, i.e. the margin γ. Applying the so-called *kernel trick*, it is possible to replace the scalar product with a *kernel function* defined over pairs of *objects*, which can more efficiently compute it:

$$c(o) = \sum_{i=1}^{\ell} \alpha_i y_i k(o_i, o) + b,$$

where $k(o_i, o) = \phi(o_i) \cdot \phi(o)$, with the advantage that we do not need to provide an explicit mapping $\phi : \mathcal{O} \rightarrow \Re^N$ of our example objects $\mathcal{O}$ in a vector space. In the next section, we show a kernel directly working on syntactic trees.

2.2 Syntactic Tree Kernel (STK)

Tree Kernel (TK) functions are convolution kernels (Haussler, 1999) defined over pairs of trees. Different TKs are characterized by alternative fragment definitions, e.g. (Collins and Duffy, 2002; Kashima and Koyanagi, 2002; Moschitti, 2006). We will focus on the syntactic tree kernel described in (Collins and Duffy, 2002), which relies on a fragment definition that does not allow to break production rules (i.e. if any child of a node is included in a fragment, then also all the other children have to). As such, it is especially indicated for tasks involving constituency parsed texts.

Tree kernels compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole feature (fragment) space. Let $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$ be the set of tree fragments, i.e. the explicit representation for the components of the fragment space, and $\chi_i(n)$ be an indicator function[2], equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. A tree kernel function over T_1 and T_2 is defined as

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2), \qquad (2)$$

where N_{T_1} and N_{T_2} are the sets of nodes in T_1 and T_2, respectively and

$$\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \chi_i(n_1) \chi_i(n_2). \qquad (3)$$

The Δ function counts the number of common subtrees rooted in n_1 and n_2 and weighs them according to their size. It can be evaluated as follows (Collins and Duffy, 2002):
1. if the productions at n_1 and n_2 are different, then $\Delta(n_1, n_2) = 0$;
2. if the productions at n_1 and n_2 are the same, and n_1 and n_2 have only leaf children (i.e. they are pre-terminal symbols) then $\Delta(n_1, n_2) = \lambda$;
3. if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not pre-terminals then

$$\Delta(n_1, n_2) = \lambda \prod_{j=1}^{l(n_1)} (1 + \Delta(c_{n_1}^j, c_{n_2}^j)), \qquad (4)$$

where $l(n_1)$ is the number of children of n_1, c_n^j is the j-th child of node n and λ is a decay factor penalizing larger structures.

2.3 Tree Fragment Weights

Eq. 3 shows that Δ counts the shared fragments rooted in n_1 and n_2 in the form of scalar product, as evaluated by Eq. 2. However, when λ is used in Δ as in Eq. 4, it changes the weight of the product $\chi_i(n_1)\chi_i(n_2)$. As λ multiplies Δ in each recursion step, we may be induced to assume[3] that the

[1]In P&M we provided an approximation of the real weight.

[2]We will consider it as a weighting function.

[3]In (Collins and Duffy, 2002), there is a short note about the correct value weight of lambda for each product components (i.e. pairs of fragments). This is in line with the formulation we provide.

weight of a fragment is λ^d, where d is the depth of the fragment. On the contrary, we show the actual weight by providing the following:

Theorem 1. *Let T and f be a tree and one of its fragments, respectively, induced by STK. The weight of f accounted by STK is $\lambda^{\frac{s(f)}{2}}$, where $l_f(n)$ is the number of children of n in f and $s(f) = |\{n \in T : l_f(n) > 0\}|$ is the number of nodes that have active productions in the fragment, i.e. the* size *of the fragment.*

In other words, the exponent of λ is the number of fragment nodes that have at least one child (i.e. active productions), divided by 2.

Proof. The thesis can be proven by induction on the depth d of f. The base case is f of depth 1. Fragments of depth 1 are matched by step 2 of $\Delta(n_1, n_2)$ computation, which assigns a value $\lambda = \chi_i(n_1)\chi_i(n_2)$ independently of the number of children (where $f_i = f$). It follows that the weight of f is $\chi_i(n_1) = \chi_i(n_2) = \lambda^{1/2}$.

Suppose that the thesis is valid for depth d and let us consider a fragment f of depth $d+1$, rooted in r. Without loss of generality, we can assume that f is in the set of the fragments rooted in n_1 and n_2, as evaluated by Eq. 4. It follows that the production rules associated with n_1 and n_2 are identical to the production rule in r. Let us consider $M = \{i \in \{1, .., l(n_1)\} : l(c_r^i) > 0\}$, i.e. the set of child indices of r which have at least a child. Thus, for $j \in M$, c_r^i has a production shared by $c_{n_1}^j$ and $c_{n_2}^j$. Conversely, for $j \notin M$, there is no match and $\Delta(c_{n_1}^j, c_{n_2}^j) = 0$. Therefore, the product in Eq. 4 can be rewritten as $\lambda \prod_{j \in M} \Delta(c_{n_1}^j, c_{n_2}^j)$, where the term 1 in $(1 + \Delta(c_{n_1}^j, c_{n_2}^j))$ is not considered since it accounts for those cases where there are no common productions in the children, i.e. $c_{n1}^j \neq c_{n2}^j \forall j \in M$.

We can now substitute $\Delta(c_{n_1}^j, c_{n_2}^j)$ with the weight of the subtree t_j of f rooted in c_r^j (and extended until its leaves), which is $\lambda^{s(t_j)}$ by inductive hypothesis (since t_j has depth lower than d). Thus, the weight of f is $s(f) = \lambda \prod_{j \in M} \lambda^{s(t_j)} = \lambda^{1+\sum_{j \in M} s(t_j)}$, where $\sum_{j \in M} s(t_j)$ is the number of nodes in f's subtrees rooted in r's children and having at least one child; by adding 1, i.e. the root of f, we obtain $s(f)$. Finally, $\lambda^{s(f)} = \chi_i(n_1)\chi_i(n_2)$, which satisfies our thesis: $\chi_i(n_1) = \chi_i(n_2) = \lambda^{\frac{s(f)}{2}}$. □

2.4 Weights in Feature Vectors

In the light of this result, we can use the definition of a TK function to project a tree t onto a linear space by recognizing that t can be represented as a vector $\vec{x}_i = [x_i^{(1)}, \ldots, x_i^{(N)}]$ whose attributes are the counts of the occurrences for each fragment, weighed with respect to the decay factor λ.

For a normalized STK kernel, the value of the j-th attribute of the example $\vec{x}_i$ is therefore:

$$x_i^{(j)} = \frac{t_{i,j}\lambda^{\frac{s(f_j)}{2}}}{\|\vec{x}_i\|} = \frac{t_{i,j}\lambda^{\frac{s(f_j)}{2}}}{\sqrt{\sum_{k=1}^{N} t_{i,k}^2 \lambda^{s(f_k)}}} \tag{5}$$

where: $t_{i,j}$ is the number of occurrences of the fragment f_j, associated with the j-th dimension of the feature space, in the tree t_i. It follows that the components of $\vec{w}$ (see Eq. 1) can be rewritten as:

$$w^{(j)} = \sum_{i=1}^{\ell} \alpha_i y_i x_i^{(j)} = \sum_{i=1}^{\ell} \frac{\alpha_i y_i t_{i,j}\lambda^{\frac{s(f_j)}{2}}}{\sqrt{\sum_{k=1}^{N} t_{i,k}^2 \lambda^{s(f_k)}}} \tag{6}$$

3 Projecting Exponentially Large Spaces

In order to provide a theoretical background to our feature selection technique and to develop effective algorithms, we want to relate our approach to statistical learning and, in particular, support vector classification theory. Since we select features with respect to their weight $w^{(j)}$, we can use the following theorem that establishes a general bound for margin-based classifiers.

Theorem 2. *(Bartlett and Shawe-Taylor, 1998) Let $\mathcal{C} = \{\vec{x} \rightarrow \vec{w} \cdot \vec{x} : \|\vec{w}\| \leq 1, \|\vec{x}\| \leq R\}$ be the class of real-valued functions defined in a ball of radius R in $\Re^N$. Then there is a constant k such that $\forall c \in \mathcal{C}$ having a margin γ, i.e. $|\vec{w} \cdot \vec{x}| \geq \gamma, \forall \vec{x} \in \mathcal{X}$ (training set), the error of c is bounded by $b/\ell + \sqrt{\frac{k}{\ell}\left(\frac{R^2}{\gamma^2}\log^2\ell + \log\frac{1}{\delta}\right)}$ with a probability $1-\delta$, where $\ell = |\mathcal{X}|$ and b is the number of examples with margin less than γ.*

In other words, if $\mathcal{X}$ is separated with a margin γ by a linear classifier, then the error has a bound depending on γ. Another conclusion is that a feature selection algorithm that wants to preserve the accuracy of the original space should not affect the margin.

Since we would like to exploit the availability of the initial gradient $\vec{w}$ derived by the application of SVMs, it makes sense to try to quantify the percentage of γ reduction after feature selection, which we indicate by ρ. We found out that γ is linked to the reduction of $||\vec{w}||$, as illustrated by the next lemma.

Lemma 1. *Let $\mathcal{X}$ be a set of points in a vector space and $\vec{w}$ be the gradient vector which separates them with a margin γ. If the selection decreases $||\vec{w}||$ of a ρ rate, then the resulting hyperplane separates $\mathcal{X}$ by a margin larger than $\gamma_{in} = \gamma - \rho R||\vec{w}||$.*

Proof. Let $\vec{w} = \vec{w}_{in} + \vec{w}_{out}$, where $\vec{w}_{in}$ and $\vec{w}_{out} \in \Re^N$ are constituted by the components of $\vec{w}$ that are selected in and out, respectively, and have zero values in the remaining positions. By hypothesis $|\vec{w} \cdot \vec{x}| \geq \gamma$; without loss of generality, we can consider just the case $\vec{w} \cdot \vec{x} \geq \gamma$, and write $\vec{w} \cdot \vec{x} = \vec{w}_{in} \cdot \vec{x} + \vec{w}_{out} \cdot \vec{x} \geq \gamma \Rightarrow \vec{w}_{in} \cdot \vec{x} \geq \gamma - \vec{w}_{out} \cdot \vec{x} \geq \gamma - |\vec{w}_{out} \cdot \vec{x}| \geq \gamma - ||\vec{w}_{out}|| \times ||\vec{x}||$, where the last inequality holds owing to Cauchy-Schwarz inequality. The margin associated with $\vec{w}_{in}$, i.e. γ_{in}, is therefore $\gamma - ||\vec{w}_{out}|| \times ||\vec{x}|| \geq \gamma - ||\vec{w}_{out}||R = \gamma - \rho R||\vec{w}||$. $\square$

Remark 1. *The lemma suggests that, even in case of very aggressive feature selection, if a small percentage ρ of $||\vec{w}||$ is lost, the margin reduction is small. Consequently, through Theorem 2, we can conclude that the accuracy of the model is by and large preserved.*

Remark 2. *We prefer to show the lemma in the more general form, but if we use normalized $\vec{x}$ and classifiers with $||\vec{w}|| \leq 1$, then $\gamma_{in} = \gamma - ||\vec{w}||\rho > \gamma - \rho$.*

The last result that we present justifies our selection approach as it demonstrates that most of the gradient norm is concentrated in relatively few features, with respect to the huge space induced by tree kernels. The selection of these few features allows us to preserve most of the norm and the margin.

Lemma 2. *Let $\vec{w}$ be a linear separator of a set of points $\mathcal{X}$, where each $\vec{x}_i \in \mathcal{X}$ is an explicit vector representations of a tree t_i in the space induced by STK and let ν be the largest $s(t_i)$, i.e. the maximum tree size. Then, if we discard fragments of size greater than η, $||\vec{w}_{out}|| \leq \frac{\nu}{\gamma^2}\sqrt{\frac{(\lambda\nu)^\eta - (\lambda\nu)^\nu}{1-\lambda\nu}}$.*

Proof. By applying simple norm properties, $||\vec{w}_{out}|| = \left\|\sum_{i=1}^{\ell} \alpha_i y_i \vec{x}_{out_i}\right\| \leq \sum_{i=1}^{\ell} ||\alpha_i y_i \vec{x}_{out_i}|| = \sum_{i=1}^{\ell} \alpha_i ||\vec{x}_{out_i}||$. To evaluate the latter, we first re-organize the summation in Eq. 5 (with no normalization) such that $\|\vec{x}_i\|^2 = \sum_{k=1}^{\nu} \sum_{j:s(f_j)=k} t_{i,j}^2 \lambda^{s(f_j)}$. Since a fragment f_j can be at maximum rooted in ν nodes, then $t_{i,j} \leq \nu$. Therefore, by replacing the number of trees of size k with the upperbound ν^k, we have $\|\vec{x}_i\| < \sqrt{\sum_{k=1}^{\nu} \nu^2 \lambda^k \nu^k} = \sqrt{\sum_{k=1}^{\nu} \nu^2 (\nu\lambda)^k} = \sqrt{\nu^2 \frac{1-\mu^\nu}{1-\mu}}$, where we applied geometric series summation. Now if we assume that our algorithm selects out (i.e. discards) fragments with size $s(f) > \eta$, $\|\vec{x}_{out_i}\| < \sqrt{\nu^2 \frac{\mu^\eta - \mu^\nu}{1-\mu}}$. It follows that $||\vec{w}_{out}|| < \sum_{i=1}^{\ell} \alpha_i \sqrt{\nu^2 \frac{\mu^\eta - \mu^\nu}{1-\mu}}$. In case of hard-margin SVMs, we have $\sum_{i=1}^{\ell} \alpha_i = 1/\gamma^2$. Thus, $||\vec{w}_{out}|| < \frac{\nu}{\gamma^2}\sqrt{\frac{\mu^\eta - \mu^\nu}{1-\mu}} = \frac{\nu}{\gamma^2}\sqrt{\frac{(\lambda\nu)^\eta - (\lambda\nu)^\nu}{1-\lambda\nu}}$. $\square$

Remark 3. *The lemma shows that for an enough large η and $\lambda < 1/\nu$, $||\vec{w}_{out}||$ can be very small, even though it includes an exponential number of features, i.e. all the subtrees whose size ranges from η to ν. Therefore, according to Lemma 1 and Theorem 2, we can discard an exponential number of features with a limited loss in accuracy.*

Remark 4. *Regarding the proposed norm bound, we observe that ν^k is a rough overestimation of the the real number of fragments having size k rooted in the nodes of the target tree t. This suggests that we don't really need $\lambda < 1/\nu$. Moreover, in case of soft-margin SVMs, we can bound α_i with the value of the trade-off parameter C.*

4 Previous Work

Initial work on feature selection for text, e.g. (Yang and Pedersen, 1997), has shown that it may improve the accuracy or, at least, improve efficiency while preserving accuracy. Our context for feature selection is different for several important reasons: (i) we focus on structured features with a syntactic nature, which show different behaviour from lexical ones, e.g. they tend to be more sparse; (ii) in the TK space, the a-priori weights are very skewed, and large fragments receive exponentially lower scores than small ones; (iii) there is high redundancy and inter-dependency between such features; (iv) we want to be able to observe the most relevant features automatically generated by TKs; and (v) the huge number of features makes it impossible to evaluate the weight of each feature individually.

Guyon and Elisseeff (2003) carries out a very informative survey of feature selection techniques. Non-filter approaches for SVMs and kernel machines are often concerned with polynomial and

Gaussian kernels, e.g. (Weston et al., 2001; Neumann et al., 2005). In (Kudo and Matsumoto, 2003), an extension of the PrefixSpan algorithm (Pei et al., 2001) is used to efficiently mine the features in a low degree polynomial kernel space. The authors discuss an approximation of their method that allows them to handle high degree polynomial kernels. Suzuki and Isozaki (2005) present an embedded approach to feature selection for convolution kernels based on χ^2-driven relevance assessment. With respect to their work, the main differences in the approach that we propose are that we want to exploit the SVM optimizer to select the most relevant features, and to be able to observe the relevant fragments.

Regarding work that may directly benefit from reverse kernel engineering is worthwhile mentioning: (Cancedda et al., 2003; Shen et al., 2003; Daumé III and Marcu, 2004; Giuglea and Moschitti, 2004; Toutanova et al., 2004; Kazama and Torisawa, 2005; Titov and Henderson, 2006; Kate and Mooney, 2006; Zhang et al., 2006; Bloehdorn et al., 2006; Bloehdorn and Moschitti, 2007; Moschitti and Zanzotto, 2007; Surdeanu et al., 2008; Moschitti, 2008; Moschitti and Quarteroni, 2008; Martins et al., 2009; Nguyen et al., 2009a)

5 Mining Fragments Efficiently

The high-level description of our feature selection technique is as follows: we start by learning an STK model and we greedily explore the support vectors in search for the most relevant fragments. We store them in an index, and then we decode (or linearize) all the trees in the dataset, i.e. we represent them as vectors in a linear space where only a very small subset of the fragments in the original space are accounted for. These vectors are then employed for learning and classification in the linear space.

To explore the fragment space defined by a set of support vectors, we adopt the greedy strategy described in Algorithm 5.1. Its arguments are a model M, and the *threshold factor* L. The greedy algorithm explores the fragment space in a small to large fashion. The first step is the generation of the all *base* fragments F encoded in each tree, i.e. the smallest possible fragments according to the definition of the kernel function. For STK, such fragments are all those consisting of a node and all its direct children (i.e. production rules of the grammar). We assess the cumulative relevance of each

Algorithm 5.1: GREEDY_MODEL_MINER(M, L)

```
$B \leftarrow$ BASE_FRAGS($model$)
$B \leftarrow$ REL(BEST($B$))
$\sigma \leftarrow B/L$
$\mathcal{D}_{prev} \leftarrow$ FILTER($B, \sigma$)
UPDATE($\mathcal{D}_{prev}$)
while $\mathcal{D}_{prev} \neq \emptyset$
  do
    $\mathcal{D}_{next} \leftarrow \emptyset$
    $\tau \leftarrow 1 /* widthfactor */$
    $\mathcal{W}_{prev} \leftarrow \mathcal{D}_{prev}$
    while $\mathcal{W}_{prev} \neq \emptyset$
      do
        $\mathcal{W}_{next} \leftarrow \emptyset$
        for each $f \in \mathcal{W}_{prev}$
          do
            $E_f \leftarrow$ EXPAND($f, \tau$)
            $F \leftarrow$ FILTER($E_f, \sigma$)
            if $F \neq \emptyset$
              then
                $\mathcal{W}_{next} \leftarrow \mathcal{W}_{next} \cup \{f\}$
                $\mathcal{D}_{next} \leftarrow \mathcal{D}_{next} \cup F$
                UPDATE($F$)
        $\tau \leftarrow \tau + 1$
        $\mathcal{W}_{prev} \leftarrow \mathcal{W}_{next}$
    $\mathcal{D}_{prev} \leftarrow \mathcal{D}_{next}$
return ($result$)
```

base fragment according to Eq. 6 and then use the relevance B of the heaviest fragment, i.e. the fragment with the highest relevance in absolute value, as a criterion to set our fragment mining threshold σ to B/L. We then apply the FILTER($\cdot$) operator which discards all the fragments whose cumulative score is less than σ. Then, the UPDATE($\cdot$) operator stores the ramaining fragments in the index.

The exploration of the kernel space is carried out via the process of fragment expansion, by which each fragment retained at the previous step is incrementally grown to span more levels of the tree and to include more nodes at each level. These two directions of growth are controlled by the outer and the inner *while* loops, respectively. Fragment expansion is realized by the EXPAND(f, n) operator, that grows the fragment f by including the children of n *expandable* nodes in the fragment. Expandable nodes are nodes which are leaves in f but that have children in the tree that originated f.

After each expansion, the FILTER($\cdot$) operator is invoked on the set of generated fragments. If the filtered set is empty, i.e. no fragments more relevant than σ have been found during the previous iteration, then the loop is terminated.

Unlike previous attempts, this algorithm relies on just one parameter, i.e. L. As it revolves around the weight of the most relevant fragment, it operates according to the norm-preservation principle described in the previous sections. In fact, if we call N the number of fragments mined for a given value of L, the norm after feature selection can be bounded by $\frac{B}{L}\sqrt{N} \leq \|w_{in}\| \leq B\sqrt{N}$.

The choice of B, i.e. the highest relevance of a base fragment, as an upper bound for fragment relevance is motivated as follows. In Eq. 6, we can identify a term $T_i = \alpha_i y_i / \|t_i\|$ that is the same for all the fragments in the tree t_i. For $0 < \lambda \leq 1$, if f_j is an expansion of f_k, then from our definition of fragment expansion it follows that $\lambda^{\frac{s(f_j)}{2}} < \lambda^{\frac{s(f_k)}{2}}$. It can also be observed that $t_{i,j} \leq t_{i,k}$. Indeed, if $t_{i,k}$ is a subset of $t_{i,j}$, then it will occur at least as many times as its expansion $t_{i,k}$, possibly occurring as a seed fragment for different expansions in other parts of the tree as well. Therefore, if $\mathcal{E}_f$ is the set of expansions of f, for every two fragments $f_{i,j}, f_{i,k}$ coming from the same tree t_i, we can conclude that $x_i^{(j)} < x_i^{(k)}\ \forall f_{i,j} \in \mathcal{E}_{f_{i,k}}$. In other words, for each tree in the model, base fragments are the most relevant, and we can assume that the relevance of the heaviest fragment is an upper bound for the relevance of any fragment [4].

6 Experiments

We ran a set of thorough experiments to support our claims with empirical evidence. We show our results on three very different benchmarks: Question Classification (QC) using TREC 10 data (Voorhees, 2001), Relation Extraction (RE) based on the newswire and broadcast news domain of the ACE 2004 English corpus (Doddington et al., 2004) and Semantic Role Labeling (SRL) on the CoNLL 2005 shared task data (Carreras and Màrquez, 2005). In the next sections we elaborate on the setup and outcome of each set of experiments. As a supervised learning framework we used SVM-Light-TK[5], which extends the SVM-Light optimizer (Joachims, 2000) with support for tree kernel functions.

Unless differently stated, all the classifiers are parametrized for optimal Precision and Recall on a development set, obtained by selecting one example in ten from the training set with the same positive-to-negative example ratio. The results that we show are obtained on the test sets by using all the available data for training. For multi-class scenarios, the classifiers are arranged in a one vs. all configuration, where each sentence is a positive example for one of the classes, and negative for the others. While binary classifiers are evaluated in terms of F_1 measure, for multi-class classifiers we show the final accuracy.

The next paragraphs describe the datasets used for the experiments.

Question Classification (QC) Given a question, the task consists in selecting the most appropriate expected answer type from a given set of possibilities. We adopted the question taxonomy known as *coarse grained*, which has been described in (Zhang and Lee, 2003) and (Li and Roth, 2006), consisting of six non overlapping classes: Abbreviations (ABBR), Descriptions (DESC, e.g. definitions or explanations), Entity (ENTY, e.g. animal, body or color), Human (HUM, e.g. group or individual), Location (LOC, e.g. cities or countries) and Numeric (NUM, e.g. amounts or dates).

The TREC 10 QA data set accounts for 6,000 questions. For each question, we generate the full parse of the sentence and use it to train our models. Automatic parses are obtained with the Stanford parser[6] (Klein and Manning, 2003), and we actually have only 5,953 sentences in our data set due to parsing issues. During preliminary experiments, we observed an uneven distribution of examples in the traditional training/test split (the same used in P&M). Therefore, we used a random selection to generate an unbiased split, with 5,468 sentences for training and 485 for testing. The resulting data set is available for download at `http://danielepighin.net/cms/research/QC_dataset.tgz`.

Relation Extraction (RE) The corpus consists of 348 documents, and contains seven relation classes defined over pairs of mentions: Physical, Person/Social, Employment/Membership/Subsidiary, Agent-Artifact, PER/ORG Affiliation, GPE Affiliation, and Discourse. There are 4,400 positive and 38,696 negative examples when the potential relations are generated using all the entity/mention pairs in the same sentence.

Documents are parsed using the Stanford Parser, where the nodes of the entities are enriched with information about the entity type. Overall, we used the setting and data defined in (Nguyen et al., 2009b).

[4]In principle, the weight of some fragment encoded in the model M may be greater than B. However, as an empirical justification, we report that in all our experiments we have never been able to observe such case. Thus, with a certain probability, we can assume that the highest weight will be obtained from the heaviest of the base fragments.

[5]http://disi.unitn.it/~moschitt/Tree-Kernel.htm

[6]http://nlp.stanford.edu/software/lex-parser.shtml

Semantic Role Labeling (SRL) SRL can be decomposed into two tasks: *boundary detection*, where the word sequences that are arguments of a predicate word w are identified, and *role classification*, where each argument is assigned the proper role. For these experiments we concentrated on this latter task and used exactly the same setup as P&M. We considered all the argument nodes of any of the six PropBank (Palmer et al., 2005) core roles[7] (i.e. A0, ..., A5) from all the available training sections, i.e. 2 through 21, for a total of 179,091 training instances. Similarly, we collected 9,277 test instances from the annotations of Section 23.

6.1 Model Comparison

To show the validity of Lemma 1 in practical scenarios, we compare the accuracy of our linearized models against vanilla STK classifiers. We designed two types of classifiers:

LIN, a *linearized STK model*, which uses the weights estimated by the learner in the STK space and linearized examples; in other words LIN uses $\vec{w}_{IN}$. It allows us to measure exactly the loss in accuracy with respect to the reduction of $||\vec{w}||$.

OPT, a linearized STK model that is *re-optimized in the linear space*, i.e. for which we retrained an SVM using the linearized training examples as input data. Since the LIN solution is part of the candidate solutions from which OPT is selected, we always expect higher accuracy from it.

Additionally, we compare selection based on gradient $\vec{w}$ (as detailed in Section 2.4) against to χ^2 selection, which evaluates the relevance of features, in a similar way to (Suzuki and Isozaki, 2005). The relevance of a fragment is calculated as

$$\chi^2 = \frac{N(yN - Mx)^2}{x(N-x)M(N-M)},$$

where N is the number of support vectors, M is the number of positive vectors (i.e. $\alpha_i > 0$), and x and y are the fractions of N and M where the fragment is instantiated, respectively. We specify the selection models by means of **Grad** for the former and **Chi** for the latter. For example, a model called *OPT/Grad* is a re-trained model using the features selected according the highest gradient weights, while *LIN/Chi* would be a linearized tree kernel model using χ^2 for feature selection.

[7]We do not consider adjuncts because we preferred the number of classes to be similar across the three benchmarks.

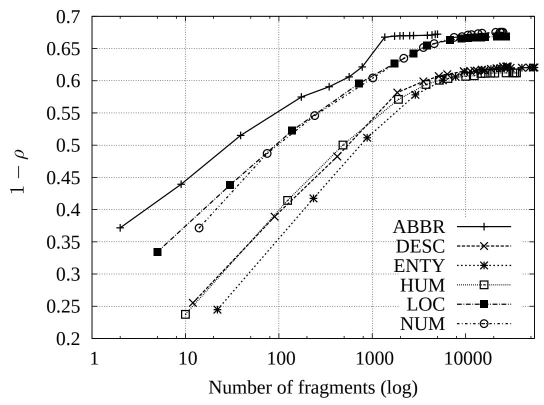


Figure 1: Percentage of gradient Norm, i.e. $1 - \rho$, according to the number of selected fragments, for different QC classifiers.

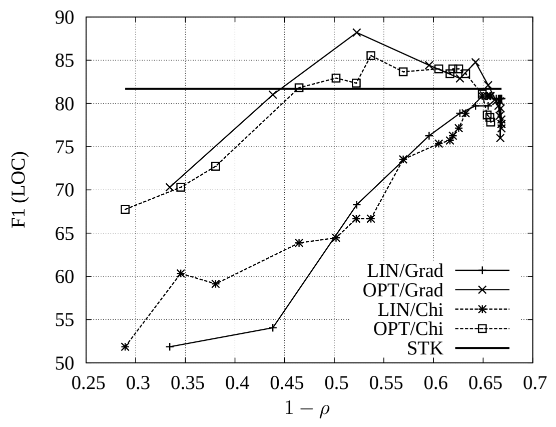
	STK		Linearized			
				LIN		OPT
	F$_1$	$\|\|\vec{w}\|\|$	Frags	F$_1$	$\|\|\vec{w}_{in}\|\|$	F$_1$
A	80.00	11.77	566	66.67	7.13	90.91
D	86.26	41.33	5161	81.87	25.10	83.72
E	76.86	51.71	5,702	73.03	31.06	75.56
H	84.92	43.61	5,232	80.47	26.20	77.08
L	81.69	38.73	1,732	78.87	24.27	82.89
N	92.31	37.65	1,015	85.07	24.53	87.07

Table 1: Per-class comparison between STK and the LIN/Grad and OPT/Grad models on the QC task. Each class is identified by its initial (e.g. A=ABBR). For each class, we considered a value of the threshold factor parameter L so as to retain at least 60% of the gradient norm after feature selection.

6.2 Results

The plots in Figure 1 show, for each class, the percentage of the gradient norm (i.e. $1 - \rho$, see Section 3) retained when including a different number of fragments. This graph empirically validates Lemma 2 since it clearly demonstrates that after 1,000-10,000 features the percentage of the norm reaches a plateau (around 60-65%). This means that after such threshold, which interestingly generalizes across all classifiers, a huge number of features is needed for a small increase of the norm. We recall that the maximum reachable norm is around 70% since we apriori filter out fragments of frequency lower than three.

Table 1 shows the F$_1$ of the binary question classifiers learned with STK, LIN/Grad and OPT/Grad models. It also shows the norm of the gradient before, $||\vec{w}||$, and after, $||\vec{w}_{in}||$, feature selec-



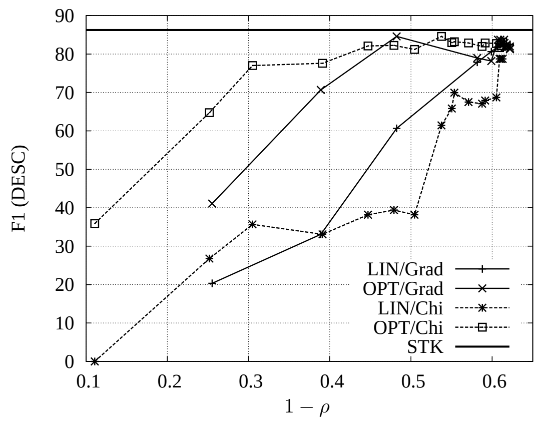


Figure 2: F1-measure of LOC and DESC wrt different $1-\rho$ values.

tion along with the number of selected fragments, *Frags*. Instead of selecting an optimal number of fragments on a validation set, we investigated the 60% value suggested by the previous plot. Thus, for each category we selected the feature set reaching approximately 60% of $||\vec{w}||$. The table shows that the accuracy of the OPT/Grad model is in line with STK. In some cases, e.g. ABBR, the projected model is more accurate, i.e. 90.91 vs. 80.00, whereas in others, e.g. HUM, STK performs better, i.e. 84.92 vs. 77.08. It is interesting to see how the empirical results clearly complement the theoretical findings of the previous sections. For example, the LOC classifier uses only 1,732 of the $\sim 10^{12}$ features encoded by the corresponding STK model, but since only 40% of the norm of $\vec{w}$ is lost, classification accuracy is affected only marginally.

As mentioned above, the selected number of features is not optimal for every class. Figure 2 plots the accuracy of the LIN/Grad and OPT/Grad for different numbers of fragments on two classes [8]. These show that the former, with

[8]The other classes, which show similar behaviour, are omitted due to lack of space.

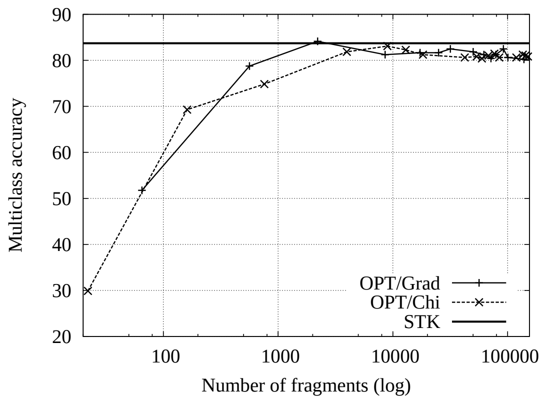


Figure 3: Multiclass accuracy obtained by including a growing number of fragments.

more than 60% of the norm, approaches STK whereas the latter requires less fragments. The plots also show the comparison against the same fragment mining algorithm and learning framework when using χ^2-based selection. This also provides similar good results, as far as the reduction of $||\vec{w}||$ is kept under control, i.e. as far as we select the components of the gradient that mostly affect its norm.

To concretely assess the benefits of our models for QC, Figure 3 plots the accuracy of OPT/Grad and OPT/Chi on the multiclass QC problem wrt the number of fragments employed. The results for the multi-class classifier are less biased by the binary Precision/Recall classifiers thus they are more stable and clearly show how, after selecting the optimal number of fragments (1,000-10,000 i.e. 60-65% of the norm), the accuracy of the OPT and CHI classifiers stabilize around levels of accuracy which are in line with STK.

	STK	OPT/Grad	
	F_1	F_1	Frags
QC	83.70	84.12	~2k
RE	67.53	66.31	~10k
SRL	87.56	88.17	~300k

Table 2: Multiclass classification accuracy on three benchmarks.

Finally, Table 2 shows the best results that we achieved on the three multi-class classification tasks, i.e. QC, RE[9] and SRL, and compares them against the STK [10]. For all the tasks OPT/Grad

[9]For RE, we show lower accuracy than in (Nguyen et al., 2009b) since, to have a closer comparison with STK, we do not combine structural features with manual designed features.

[10]We should point out that this models are only partially

produces the best results for all the tests, even though the difference with OPT/Chi is generally not statistically significant. Out of three tasks, OPT/Grad manages to slightly improve two of them, i.e. QC (84.12 vs. 83.7) and SRL (88.17 vs. 87.56), while STK is more accurate on RE, i.e. 67.53 vs. 66.31.

6.3 Comparison with P&M

The results on SRL can be compared against those that we presented in (Pighin and Moschitti, 2009a), where we measured an accuracy of 87.13 exactly on the same benchmark. As we can see in Table 2, our model improves the classification accuracy of about 1 point, i.e. 88.17. On the other hand, such comparison is not really fair since the algorithms rely on different parameter sets, and it is almost impossible to find matching configurations for the different versions of the algorithms that would result in exactly the same number of fragments. In a projected space with approximately 10^3 or 10^4 fragments, including a few hundred more features can produce noticeably different accuracy readings.

Generally speaking, the current model can achieve comparable accuracy with P&M while considering a smaller number of fragments. For example, in (Pighin and Moschitti, 2009b) the best model for the A1 binary classifier of the SRL benchmark was obtained by including 50,000 fragments, achieving an F_1 score of 89.04. With the new algorithm, using approximately half the fragments the accuracy of the linearized A1 classifier is 90.09. In P&M, the algorithm would only consider expansions of a fragment f where at most m nodes are expanded. Consequently, the set of mined fragments may include some small structures which can be less relevant than larger ones. Conversely, the new algorithm (see Alg. 5.1) may include larger but more relevant structures, thus accounting for a larger fraction of the gradient norm with a smaller number of fragments.

Concerning efficiency, the complexity of both mining algorithms is proportional to the number of fragments that they generate. Therefore, we can conclude that the new implementation is more efficient by considering that we can achieve the same accuracy with less fragments. As for the complexity of decoding, i.e. providing explicit vector representations of the input trees, in P&M, we used a very naive approach, i.e. the generation of all the fragments encoded in the tree and then look up each fragment in the index. This solution has exponential complexity with the number of nodes in the tree. Conversely, the new implementation has approximately linear complexity. The approach is based on the idea of an FST-like index, that we can *query* with a tree node. Every time the tree *matches* one of the fragments, the index increases the count of that fragment for the tree. The reduction in time complexity is made possible by encoding in the index the sequence of expansion operations that produced each indexed fragment, and by considering only those expansions at decoding time.

optimized, as we evaluated them by using the same threshold factor parameter L for all the classes. Better performances could be achieved by selecting an optimal value of L for individual classes when building the multi-class classifier.

7 Conclusions

Available feature selection frameworks for very high dimensional kernel families, such as tree kernels, suffer from the lack of a theory that could justify the very aggressive selection strategies necessary to cope with the exceptionally high dimensional feature space.

In this paper, we have provided a theoretical foundation in the context of margin classifiers by (i) linking the reduction of the gradient norm to the theoretical error bound and (ii) by proving that the norm is mostly concentrated in a relatively small number of features. The two properties suggest that we can apply an extremely aggressive feature selection by keeping the same accuracy. We described a very efficient algorithm to carry out such strategy in the fragment space. Our experiments empirically support our theoretical findings on three very different NLP tasks.

Acknowledgements

We would like to thank Truc-Vien T. Nguyen for providing us with the SVM learning and test files of the Relation Extraction dataset. Many thanks to the anonymous reviewers for their valuable suggestions.
This research has been partially supported by the EC project, EternalS: “Trustworthy Eternal Systems via Evolving Software, Data and Knowledge”, project number FP7 247758.

References

P. Bartlett and J. Shawe-Taylor, 1998. *Advances in Kernel Methods — Support Vector Learning*, chapter Generalization Performance of Support Vector Machines and other Pattern Classifiers. MIT Press.

Stephan Bloehdorn and Alessandro Moschitti. 2007. Structure and semantics for expressive text kernels. In *In Proceedings of CIKM '07*.

Stephan Bloehdorn, Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2006. Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of ICDM 06, Hong Kong, 2006*.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. 2003. Word sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL'05*.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of ACL'02*.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of ACL'04*.

Chad Cumby and Dan Roth. 2003. Kernel Methods for Relational Learning. In *Proceedings of ICML 2003*.

Hal Daumé III and Daniel Marcu. 2004. Np bracketing by maximum entropy tagging and SVM reranking. In *Proceedings of EMNLP'04*.

G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. 2004. The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation. *Proceedings of LREC 2004*, pages 837–840.

Ana-Maria Giuglea and Alessandro Moschitti. 2004. Knowledge Discovering using FrameNet, VerbNet and PropBank. In *In Proceedings of the Workshop on Ontology and Knowledge Discovering at ECML 2004, Pisa, Italy*.

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

David Haussler. 1999. Convolution kernels on discrete structures. Technical report, Dept. of Computer Science, University of California at Santa Cruz.

T. Joachims. 2000. Estimating the generalization performance of a SVM efficiently. In *Proceedings of ICML'00*.

Hisashi Kashima and Teruo Koyanagi. 2002. Kernels for semi-structured data. In *Proceedings of ICML'02*.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st ICCL and 44th Annual Meeting of the ACL*, pages 913–920, Sydney, Australia, July. Association for Computational Linguistics.

Jun'ichi Kazama and Kentaro Torisawa. 2005. Speeding up training with tree kernels for node relation labeling. In *Proceedings of HLT-EMNLP'05*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL'03*, pages 423–430.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL'03*.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*.

Xin Li and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2009. Nonextensive information theoretic kernels on measures. *J. Mach. Learn. Res.*, 10:935–975.

Alessandro Moschitti and Silvia Quarteroni. 2008. Kernels on linguistic structures for answer extraction. In *Proceedings of ACL-08: HLT, Short Papers*, Columbus, Ohio.

Alessandro Moschitti and Fabio Massimo Zanzotto. 2007. Fast and effective kernels for relational learning from texts. In Zoubin Ghahramani, editor, *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of ECML'06*, pages 318–329.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *Proceeding of CIKM '08*, NY, USA.

Julia Neumann, Christoph Schnorr, and Gabriele Steidl. 2005. Combined SVM-Based Feature Selection and Classification. *Machine Learning*, 61(1-3):129–150.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009a. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of EMNLP*.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009b. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1378–1387, Morristown, NJ, USA. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, 31(1):71–106.

J. Pei, J. Han, Mortazavi B. Asl, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu. 2001. PrefixSpan Mining Sequential Patterns Efficiently by Prefix Projected Pattern Growth. In *Proceedings of ICDE'01*.

Daniele Pighin and Alessandro Moschitti. 2009a. Efficient linearization of tree kernel functions. In *Proceedings of CoNLL'09*.

Daniele Pighin and Alessandro Moschitti. 2009b. Reverse engineering of tree kernel feature spaces. In *Proceedings of EMNLP*, pages 111–120, Singapore, August. Association for Computational Linguistics.

Libin Shen, Anoop Sarkar, and Aravind k. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In *Proceedings of EMNLP'06*.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to rank answers on large online QA collections. In *Proceedings of ACL-08: HLT*, Columbus, Ohio.

Jun Suzuki and Hideki Isozaki. 2005. Sequence and Tree Kernels with Statistical Feature Mining. In *Proceedings of NIPS'05*.

Ivan Titov and James Henderson. 2006. Porting statistical parsers with data-defined kernels. In *Proceedings of CoNLL-X*.

Kristina Toutanova, Penka Markova, and Christopher Manning. 2004. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In *Proceedings of EMNLP 2004*.

Ellen M. Voorhees. 2001. Overview of the trec 2001 question answering track. In *In Proceedings of the Tenth Text REtrieval Conference (TREC*, pages 42–51.

Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. 2001. Feature Selection for SVMs. In *Proceedings of NIPS'01*.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US. Morgan Kaufmann Publishers, San Francisco, US.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of SIGIR'03*, pages 26–32.

Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring Syntactic Features for Relation Extraction using a Convolution tree kernel. In *Proceedings of NAACL*.